A Probabilistic Model for Component-Based Shape Synthesis

Evangelos Kalogerakis   Siddhartha Chaudhuri   Daphne Koller   Vladlen Koltun

Stanford University

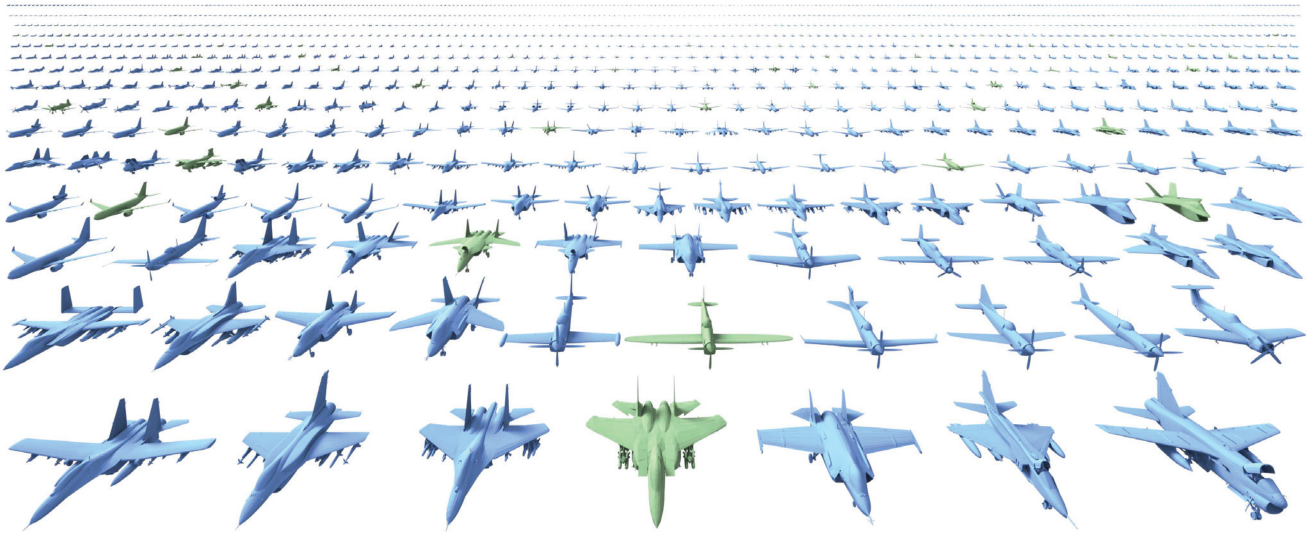


Figure 1: *Given* 100 *training airplanes (green), our probabilistic model synthesizes* 1267 *new airplanes (blue).*

Abstract

We present an approach to synthesizing shapes from complex domains, by identifying new plausible combinations of components from existing shapes. Our primary contribution is a new generative model of component-based shape structure. The model represents probabilistic relationships between properties of shape components, and relates them to learned underlying causes of structural variability within the domain. These causes are treated as latent variables, leading to a compact representation that can be effectively learned without supervision from a set of compatibly segmented shapes. We evaluate the model on a number of shape datasets with complex structural variability and demonstrate its application to amplification of shape databases and to interactive shape synthesis.

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

Keywords: shape synthesis, shape structure, probabilistic graphical models, machine learning, data-driven 3D modeling

Links: DL PDF

1 Introduction

The creation of compelling three-dimensional content is a central problem in computer graphics. Many applications such as games and virtual worlds require large collections of three-dimensional shapes for populating environments, and modeling each shape individually can be tedious even with the best interactive tools. This is particularly true for small development teams that lack 3D modeling expertise and resources. Such users can benefit from tools that automatically synthesize a variety of new, distinct shapes from a given domain.

Tools for automatic synthesis of shapes from complex real-world domains must understand what characterizes the structure of shapes within such domains. Developing formal models of this structure is challenging, since shapes in many real-world domains exhibit complex relationships between their components. Consider sailing ships. Sailing ships vary in the size and type of hull, keel and masts, as well as in the number and configuration of masts. Different types of sailing ships constrain these factors differently. For example, yawls are small crafts with a shallow hull that supports two masts with large, triangular sails. Caravels are small, highly maneuverable ships carrying two or three masts with triangular sails. Galleons are multi-decked vessels with much larger hulls and primarily square sails on three or more masts. Various geometric, stylistic and functional relationships influence the selection and placement of individual components to ensure that the final shape forms a coherent whole. Similarly complex networks of relationships characterize other domains such as airplanes, automobiles, furniture, and various biological forms.

The focus of our work is on designing a compact representation of these relationships that can be learned without supervision from a limited number of examples. Our primary contribution is a generative probabilistic model of shape structure that can be trained on a set of compatibly segmented shapes from a particular domain. The model compactly represents the structural variability within the domain, without manual tuning or any additional specification of the domain. Given a trained model, plausible new shapes from the domain can be automatically synthesized by combining existing components, subject to optional high-level constraints. The key idea in the design of the model is to relate probabilistic relationships between geometric and semantic properties of shape components to learned latent causes of structural variability, both at the level of individual component categories and at the level of the complete shape.

We demonstrate two applications of the presented model. First, it can be used to amplify an existing shape database. Given a limited number of example shapes, the model can synthesize a large number of new shapes, expanding the size of the database by an order of magnitude. For example, given a hundred airplanes, the model can automatically synthesize over a thousand new airplanes, each distinct from those in the input set (Figure 1). Second, the model enables interactive shape synthesis interfaces that allow rapid creation of plausible shapes subject to high-level constraints.

2 Related Work

Our work is closely related to research on assembly-based 3D modeling, which aims to facilitate interactive composition of shapes from components. The pioneering Modeling by Example system by Funkhouser et al. [2004] used a database of segmented shapes to enable interactive assembly of new shapes from retrieved components. A follow-up project extended this approach to sketch-based retrieval of components [Lee and Funkhouser 2008]. The Shuffler system of Kraevoy et al. [2007] allows interchanging components between shapes in order to interactively create new shapes. Chaudhuri and Koltun [2010] describe an approach to retrieving compatible components for incomplete shapes during assembly-based modeling. Xu et al. [2011] describe a system that fits components from a retrieved database shape to the silhouette of an object extracted from a photograph. Jain et al. [2012] describe a method that interpolates between two shapes by combining components from these shapes. None of these techniques allow automatic synthesis of plausible new shapes with novel structure from a complex domain described only by a set of examples.

The most related assembly-based modeling technique is by Chaudhuri et al. [2011], who develop a probabilistic representation of shape structure that can be used to suggest relevant components during an interactive assembly-based modeling session. While their probabilistic model can be used to assemble complete novel shapes, the plausibility of the synthesized shapes is severely limited. This is due to a number of factors, including the use of probability tables, the use of the Bayesian Information Criterion, and, most notably, the flat nature of the model, which does not account for latent causes of structural variability. The model is thus sufficient for suggesting individual components, but is unsatisfactory for synthesizing complete shapes. We evaluate the performance of this model against ours in Section 7.

Our work is also related to techniques that analyze a single input shape and generate larger shapes by exploiting adjacencies and repeated patterns within the input, akin to texture synthesis [Merrell 2007; Merrell and Manocha 2011; Bokeloh et al. 2010]. However, these previous techniques produce shapes that are only locally similar to the input and do not represent the global structure of shapes within a complex domain.

Prior works on learning models of variability in collections of shapes have primarily focused on continuous variability. These include SCAPE [Anguelov et al. 2005], a learned model of variation in human shape and pose, and the earlier works of Blanz and Vetter [1999] and Allen et al. [2003]. A related recent work by Ovsjanikov et al. [2011] enables exploration of continuous variability in collections of shapes by means of a deformable template. Our work can be seen as a limited generalization of SCAPE to domains in which shapes differ significantly in their component structure.

Our work can also be viewed as a generalization of inverse procedural modeling [Aliaga et al. 2007; Stava et al. 2010; Bokeloh et al. 2010], which aims to reconstruct a procedural representation from a given exemplar shape. Prior inverse procedural modeling techniques analyzed single example shapes in isolation. In contrast, we model structural variability in complex domains exemplified by a set of shapes.

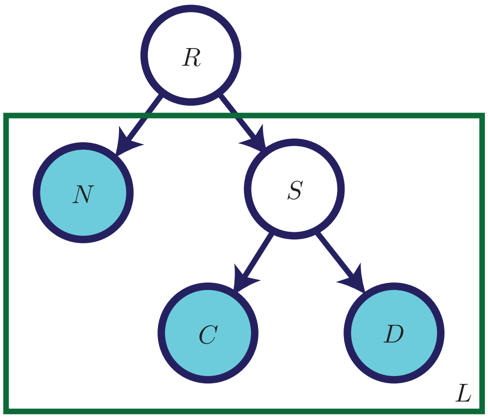
Our probabilistic model is related to a number of hierarchical generative models for object recognition in images [Bouchard and Triggs 2005; Tu et al. 2005; Jin and Geman 2006; Fidler and Leonardis 2007; Todorovic and Ahuja 2008; Zhu et al. 2008; Ommer and Buhmann 2010; Roux et al. 2011; Ranzato et al. 2011]. Like many of these models, we employ latent variables to represent higher-level concepts, and learn both the content of the latent variables and some of the structure of the model from data. However, our model operates not on image pixels or patches, but on geometric and semantic features of three-dimensional shape components. It is specifically designed to have a compact parameterization so as to synthesize complete plausible novel shapes after training on only a small number (of up to a hundred) examples.

3 Probabilistic Model

Our probabilistic model is designed to represent the component-based structure of shapes in complex domains such as furniture, aircraft, vehicles, etc. Our key observation is that the structural variability in such domains is often characterized by the presence of multiple underlying types of shapes and their components. For example, the expected set of components present in a chair—as well as their geometry—differs markedly between office chairs, dining chairs, and cantilever chairs. This observation allows us to design a compact hierarchical model that can be effectively trained on small datasets. Our model incorporates latent variables that parameterize the types of shapes in the domain as well as the styles of individual components. The latent variables and their probabilistic relationships to other variables in the model are learned from data.

The model is trained on a set of compatibly segmented shapes. We do not require that the set of components in the examples be consistent: some airplanes have horizontal stabilizers and some do not, some have landing gear while others don't, etc. Our requirement from the input segmentation is rather that compatible components be identified as such; thus horizontal stabilizers on all example airplanes should be labeled as belonging to the same category. In our implementation, we use the compatible segmentation and labeling technique of Kalogerakis et al. [2010], assisted by manual segmentation and labeling. The component categories were labeled with semantically meaningful labels, but this is not a requirement for our approach and an unsupervised compatible segmentation technique could have been used instead [Huang et al. 2011; Sidi et al. 2011].

Random variables. Our model is illustrated in Figure 2. It is a hierarchical mixture of distributions over attributes of shape components, with a single latent variable R at the root. This variable can be interpreted as the overall type of the shape. For each component category l, there is also a latent variable S_l that aims to represent the styles of components from l. The latent variables are not directly observed from the training data and are learned as described in Section 4.



notation	domain	interpretation
R	$R \in \mathbb{Z}^+$	shape style
$\mathbf{S} = \{S_l\}$	$S_l \in \{0\} \cup \mathbb{Z}^+$	component style per category l; value 0 means no components from l exist
$\mathbf{N} = \{N_l\}$	$N_l \in \{0\} \cup \mathbb{Z}^+$	number of components from category l
$\mathbf{C} = \{\mathbf{C}_l\}$	$\mathbf{C}_l \in \mathbb{R}^{p_l}$	continuous geometric feature vector for components from category l
$\mathbf{D} = \{\mathbf{D}_l\}$	$\mathbf{D}_l \in \mathbb{Z}^{p'_l}$	discrete geometric feature vector for components from category l

Figure 2: *Probabilistic model for component-based shape synthesis.* ***Top:*** *Visualization of the model's structure. Each node represents a random variable. Shaded nodes correspond to observed variables, non-shaded nodes correspond to latent variables. The visualization uses plate notation: the variables of the larger rectangle are replicated L times, where L is the number of component categories.* ***Bottom:*** *The random variables used in the model.*

Observed random variables describe attributes that can be unambiguously extracted from the data. These include N_l, the number of components from category l, $\mathbf{C}_l$, a vector of continuous geometric features of components from category l (with dimensionality p_l), and $\mathbf{D}_l$, a vector of discrete geometric features of components from category l (with dimensionality p'_l). In our implementation, the continuous features include curvature histograms, shape diameter histograms, scale parameters, spin images, PCA-based descriptors, and lightfield descriptors. These features are described further in Appendix A. The discrete features encode adjacency information. Specifically, these features specify the number of components from each category l' that are adjacent to components from category l. The discrete features help ensure that components selected for a synthesized shape have compatible numbers of adjacent components of each type, so that they can be assembled into a coherent shape using the optimization procedure described in Section 5.2.

Model structure. The random variables are organized hierarchically, as shown in Figure 2, so that the latent variables produce a hierarchical clustering effect: the values of the random variables S_l represent clusters of similar components in terms of their geometric and adjacency features, while the values of the root variable R represent clusters of similar shapes in terms of the style and numbers of their components. In addition, the model includes lateral conditional dependencies between the observed random variables, which are not shown in Figure 2. For example, variables $\mathbf{C}_l$ and $\mathbf{C}_{l'}$ can be connected by an edge. Such lateral connections represent strong relationships between attributes of different components.

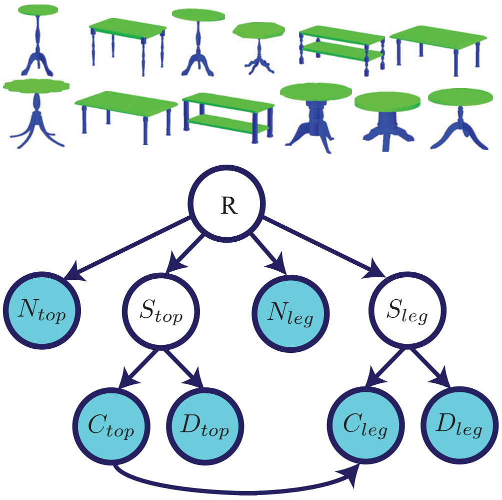


Figure 3: *Illustrative example. A small dataset of tables (top) and the probabilistic model learned for this dataset (bottom).*

Illustrative example. Figure 3 shows a small dataset of compatibly segmented tables and a probabilistic model learned for this dataset. The input shapes have two component categories: legs and tabletops. The random variable R represents the styles of tables in the dataset. Our model learned that there the two dominant styles: one-legged tables and four-legged tables. For each style, the model represents the conditional distribution over the number of components from each category: specifically, the number of tabletops (N_{top}) and the number of legs (N_{leg}). For example, in the four-legged table style, the number of tabletops is either one or, less commonly, two. The model also represents the styles of components from each category (S_{top} and S_{leg}). In this example, the model learned that there are two dominant tabletop styles: rectangular tabletops and roughly circular tabletops. The model also learned that there are two leg styles: narrow column-like legs and legs with a split base. The probability distributions in the model represent the tendency of rectangular tabletops and narrow column-like legs to be commonly associated with four-legged tables, and the tendency of roughly circular tabletops and split legs to be commonly associated with one-legged tables.

The variables C_{top} and C_{leg} represent continuous geometric features for the respective component categories, while D_{top} and D_{leg} represent discrete geometric features. For example, the conditional probability distribution associated with the variable D_{leg} encodes that legs in four-legged tables can be adjacent to either one or two tabletops. There is also a learned lateral edge that represents a strong relationship between the continuous geometric features C_{top} and C_{leg}. For one-legged tables, the conditional probability distribution associated with C_{leg} indicates that the horizontal extent of the base of the leg is positively correlated with the horizontal extent of the tabletop. This prevents the composition of narrow bases with wide tabletops: such shapes were never observed in the data and would be unstable and visually implausible.

Probability distribution represented by the model. The model represents a joint probability distribution $P(\mathbf{X})$ over all random variables $\mathbf{X} = \{R, \mathbf{S}, \mathbf{N}, \mathbf{C}, \mathbf{D}\}$. This distribution is factorized as a product of conditional probability distributions (CPDs) as follows:

$$P(\mathbf{X}) = P(R) \prod_{l \in \mathcal{L}} \left[P(S_l \mid R) P(N_l \mid R, \pi(N_l)) P(\mathbf{C}_l \mid S_l, \pi(\mathbf{C}_l)) \right.$$
$$\left. P(\mathbf{D}_l \mid S_l, \pi(\mathbf{D}_l)) \right],$$

where $\pi(N_l), \pi(\mathbf{C}_l), \pi(\mathbf{D}_l)$ are the sets of observed random variables that are linked to N_l, $\mathbf{C}_l$ and $\mathbf{D}_l$ by lateral edges.

Parametrization of CPDs for discrete variables. The CPDs for the discrete random variables $\mathbf{T} = \{S_l, N_l, D_l\}$ of the model can be represented as conditional probability tables (CPTs). Consider a discrete random variable T with a single parent discrete variable U. For every assignment t to T and u to U, the CPT at T stores the entry

$$P(T = t \mid U = u) = q_{t|u}.$$

The values $\mathbf{Q} = \{q_{t|u}\}$ comprise the parameters of the CPT. For the random variable R, which has no parents, we simply store the probability table $P(R = r) = q_r$.

When a discrete random variable has a set of multiple parents $\mathbf{U} = \{U_1, U_2, \ldots, U_m\}$, we use sigmoid functions instead of a CPT to parametrize its CPD. Sigmoid functions reduce the complexity of the model and improve generalization, since the number of parameters in a sigmoid CPD increases linearly with the number and domain size of the parent random variables, while the number of parameters of CPTs increases exponentially. Models with a large number of parameters are more prone to overfitting the training data. The sigmoid CPD is expressed as follows:

$$P(T = t \mid \mathbf{U} = \mathbf{u}) = \frac{\exp(w_{t,0} + \sum_{j=1}^{m} \mathbf{w}_{t,j} \cdot \mathbf{I}_j(u_j))}{\sum_{t' \in \mathcal{T}} \exp(w_{t',0} + \sum_{j=1}^{m} \mathbf{w}_{t',j} \cdot \mathbf{I}_j(u_j))},$$

where $\mathbf{u} = \{u_1, u_2, \ldots, u_m\}$ is the assignment to the parent variables, $\mathbf{W} = \{w_{t,0}, \mathbf{w}_{t,*}\}_{t \in \mathcal{T}}$ are the parameters of the sigmoids, and $\mathbf{I}_j(u_j) = \{I(U_j = u_j)\}$ is a vector-valued binary indicator function for each of the parent variables U_j.

Parametrization of CPDs for continuous variables. The CPD for each continuous random variable $\mathbf{C}$ is expressed as a conditional linear multivariate Gaussian. Let $\mathbf{U} = \{U_1, U_2, \ldots, U_m\}$ be the discrete and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ the continuous parents of $\mathbf{C}$. The conditional linear Gaussian for $\mathbf{C}$ is defined as

$$P(\mathbf{C} \mid \mathbf{u}, \mathbf{v}) = \mathcal{N}\left(\boldsymbol{\phi}_{\mathbf{u},0} + \sum_{j=1}^{n} \boldsymbol{\phi}_{\mathbf{u},j} \cdot \mathbf{v}_j ; \Sigma_{\mathbf{u}}\right),$$

where $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_{\mathbf{u},*}\}$ and $\boldsymbol{\Sigma} = \{\Sigma_{\mathbf{u}}\}$ are the parameters of the conditional Gaussian for each $\mathbf{u}$ in the value space $\mathcal{U}$ of $\mathbf{U}$, and $\mathbf{v}$ is the vector of assignments to the continuous parents $\mathbf{Z}$. Specifically, the parameters $\boldsymbol{\phi}_{\mathbf{u},0}$ and $\Sigma_{\mathbf{u}}$ are the conditional mean and the covariance matrix, respectively, while the remaining parameters $\boldsymbol{\Phi}$ are regression coefficients. If $\mathbf{C}$ has no continuous parents, the CPD becomes

$$P(\mathbf{C} \mid \mathbf{u}) = \mathcal{N}(\boldsymbol{\phi}_{\mathbf{u},0}; \Sigma_{\mathbf{u}}) \quad \text{for each } \mathbf{u} \in \mathcal{U}.$$

The parameters $\Theta = \{\boldsymbol{\Phi}, \boldsymbol{\Sigma}, \mathbf{Q}, \mathbf{W}\}$, the value spaces of the hidden random variables, and the edges between the observed random variables are learned as described in the next section.

4 Learning

We now describe the offline procedure for learning the structure and parameters of the probabilistic model. The input is a set of K compatibly segmented shapes. For each component, we compute its geometric attributes as described in Appendix A. Our training data is thus a set of feature vectors $\mathbf{O} = \{O_1, O_2, \ldots, O_K\}$, where $O_k = \{\mathbf{N}_k, \mathbf{D}_k, \mathbf{C}_k\}$. Our goal is to learn the structure of the model (domain sizes of latent variables and lateral edges between observed variables) and the parameters of all CPDs in the model.

The desired structure G is the one that has highest probability given input data $\mathbf{O}$ [Koller and Friedman 2009]. By Bayes' rule, this probability can be expressed as

$$P(G \mid \mathbf{O}) = \frac{P(\mathbf{O} \mid G)P(G)}{P(\mathbf{O})},$$

where the denominator is a normalizing factor that does not distinguish between different structures. Assuming a uniform prior $P(G)$ over possible structures, maximizing $P(G \mid \mathbf{O})$ reduces to maximizing the marginal likelihood $P(\mathbf{O} \mid G)$. In order to avoid overfitting and achieve better generalization, we assume prior distributions over the parameters Θ of the model. Integrating over the parameters, the marginal likelihood can be expressed as

$$P(\mathbf{O} \mid G) = \sum_{R,\mathbf{S}} \int P(\mathbf{O}, R, \mathbf{S} \mid \Theta, G) P(\Theta \mid G) \, \mathrm{d}\Theta,$$

where $P(\Theta \mid G)$ are the parameter priors. Our choice of priors is discussed in the supplementary material. In the above expression, the marginal likelihood involves summing over all possible assignments to the latent variables R and $\mathbf{S}$, thus the number of integrals is exponentially large. To make the learning procedure computationally tractable, we use an effective approximation of the marginal likelihood known as the Cheeseman-Stutz score [Cheeseman and Stutz 1996]:

$$P(\mathbf{O} \mid G) \approx P(\mathbf{O}^* \mid G) \cdot \frac{P(\mathbf{O} \mid G, \tilde{\Theta}_G)}{P(\mathbf{O}^* \mid G, \tilde{\Theta}_G)}. \tag{1}$$

Here $\tilde{\Theta}_G$ are the parameters estimated for a given G, and $\mathbf{O}^*$ is a fictitious dataset that comprises the training data $\mathbf{O}$ and approximate statistics for the values of the latent variables. The computation of this score for a given structure is described in detail in supplementary material.

Structure search. The Cheeseman-Stutz score is maximized by greedily searching over different structures G. We resort to greedy search in order to decrease the computational costs associated with the score evaluation for each candidate structure. The search proceeds as follows: we start with a domain size of 1 for R, corresponding to a single shape style. Then, for each component style S_l in each category l, we evaluate the score with a domain size of 2. This corresponds to a single component style, since the value 0 for S_l denotes the absence of components from this category. We then gradually increase the domain size of S_l, evaluating the score at each step. If the score decreases, the previous value (a local maximum) is retained as the domain size for S_l and the search moves to the next component category. After the search iterates over all variables in $\mathbf{S}$, we increase the domain size of R and repeat the procedure. The search terminates when the score reaches a local maximum that does not improve over 10 subsequent iterations; the domain size for R is set to the value that yielded the highest score, and the domain sizes for all variables in $\mathbf{S}$ are set to the corresponding locally maximal values.

Once the domain sizes for the latent variables have been determined, we search over possible sets of lateral edges between observed random variables, by locally adding, removing, and flipping possible edges, and evaluating the Cheeseman-Stutz score for each attempted structure. We retain the graph structure that yields the highest score, along with the corresponding parameters for all the CPDs in the model, computed as described below.

Parameter estimation. For a given structure G, we perform maximum a posteriori (MAP) estimation of the parameters. The parameters cannot be optimized in closed form, since the content of the latent random variables is unknown. Thus, MAP estimates are found with the expectation-maximization (EM) algorithm. The details of the EM algorithm are discussed in the supplementary material, along with the computation of the three terms in (1). All computations involving probabilities are performed in log-space to avoid numerical errors.

5 Shape Synthesis

A model trained on a set of shapes as described in Section 4 can be used to synthesize new shapes. The synthesis proceeds in two stages. In the first stage, we enumerate high-probability instantiations of the model. Each instantiation specifies a set of components. In the second stage, we optimize the placement of these components to produce a cohesive shape.

5.1 Synthesizing a set of components

Instantiations of the model, corresponding to sets of components, can be found through forward sampling. However, this random sampling process is biased towards higher-probability assignments to the random variables. As a result, valid lower-probability assignments can take an exponentially long time to be discovered. Forward sampling is thus unsuitable for efficiently enumerating all instantiations that have non-negligible probability.

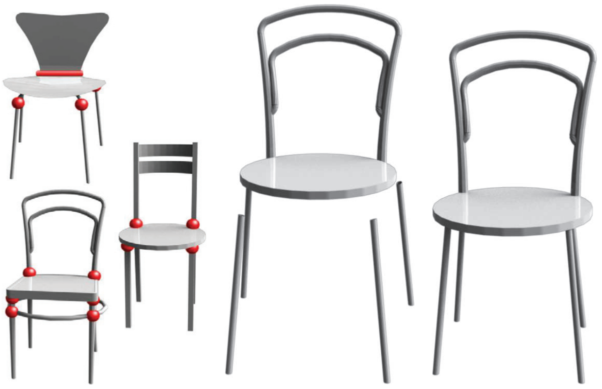
Instead, we use a simple deterministic procedure. First, we topologically sort the nodes in the model. Note that the style variables R and $\mathbf{S}$ always appear before other variables, but the complete ordering depends on the learned graph structure. Next, we create a tree whose nodes correspond to partial assignments to the random variables. The tree is initialized with an empty root node. The children of the root are all possible assignments to R. The tree continues to expand by creating nodes for the next partial assignment based on the values of the next random variable in the sorted list.

When the algorithm reaches the continuous variables $\mathbf{C}_l$, the partial assignments could in principle take any value in $\mathbb{R}^{\dim(\mathbf{C}_l)}$, which would make the search infeasible. However, only specific values of these variables correspond to geometric features of components extracted from the training set. Therefore, we expand the partial assignments only to those values of $\mathbf{C}_l$ that correspond to existing components from category l. Branches that contain assignments that have extremely low probability density (less than 10^{-12} in our implementation) are pruned from the tree. Each complete assignment obtained in this way corresponds to a set of components that can be combined to form a new shape.

5.2 Optimizing component placement

Given a set of components selected by the model, we need to place them relative to each other to form a cohesive shape. To assist this placement, certain regions on each component are marked as "slots" that specify where this component can be attached to other components. These slots, shown in red in Figure 4(a), are extracted automatically from the training data, as described in Appendix A. The discrete features D in the model encode the number of adjacent components for each component category, which ensures that the set of components synthesized by the model has compatible sets of slots.

Each slot stores the category label of components it can be attached to. It also stores simple automatically extracted symmetry relationships that allow correct relative placement of symmetric groups of adjacent components. In the example shown in Figure 4, the chair seat has two slots for front legs and two slots for back legs. The symmetry information stored in the seat slots specifies that the leg placed in the front right slot of the seat is a symmetric counterpart of the leg placed in the front left slot, reflected by one of the symmetry planes of the seat. Note that these relationships are a property of the seat and not of the legs, thus different legs in a new chair can be placed consistently around the same seat. This gives us a deterministic procedure for placing each component vis-a-vis its adjacent components by matching slots and applying the stored symmetry transforms. If a component has multiple adjacent components, its placement at this initial stage is determined by the largest adjacent component.



(a) Source shapes (b) Unoptimized (c) Optimized

Figure 4: *Optimizing component placement. (a) Three source shapes from the chair database. "Slots" are highlighted in red. (b) A set of components for a new chair synthesized by the probabilistic model. The components are drawn from the source shapes shown on the left. Initially, the legs and the back are placed relative to the seat according to transformations stored in the seat's slots. (c) The final synthesized chair, with components placed and scaled by the least-squares optimization that aligns all pairs of adjacent components at their corresponding slots.*

This initial placement is further refined by an optimization step that aligns all pairs of adjacent components at their points of contact. The optimization minimizes a squared error term over the slots, which penalizes discrepancies of position and relative size between each pair of adjacent slots, expressed as a function of the translation and scaling parameters of the corresponding components. To ensure that components are not drastically distorted in scale, the optimization penalizes deviation from the original component scales, weighting the error in each parameter by a learned variance in scale. Finally, the error term also penalizes deviations from symmetry and ground contact. (Ground contact points are extracted as described in Appendix A.) We used the same objective term weights for all domains in our evaluation. A linear least-squares solver, with non-negativity constraints for the scaling parameters, is used to minimize the error. To enhance the visual appearance of synthesized shapes, we glue adjacent components of organic shapes by matching adjacent edge loops, shifting local neighborhoods with a smooth falloff, and applying a final Laplacian smoothing filter. A more sophisticated implementation could benefit from more advanced gluing techniques [Sharf et al. 2006].

	planes	c. vehicles	chairs	ships	creatures
# of training shapes	100	22	88	42	69
# of categories	14	7	11	30	11
# of components	881	122	504	639	593
# of synth. shapes	1267	253	870	199	563

Table 1: *Datasets used in the evaluation.*

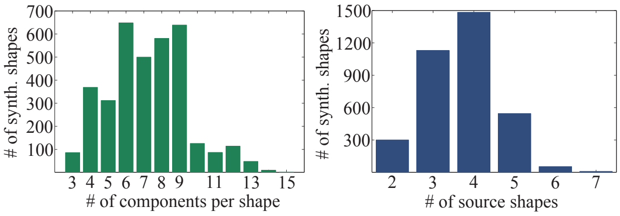


Figure 5: *Left: histogram of number of components used per synthesized shape. Right: histogram of number of source shapes contributing components per synthesized shape.*

6 Applications

We describe two applications of the presented model. The first is amplification of an input shape database and the second is constrained shape synthesis based on high-level specifications provided interactively by a user.

Shape database amplification. The first application is a direct result of applying the learning and inference procedures described in the preceding sections. Given an input dataset of compatibly segmented and labeled shapes, we train the model as described in Section 4. We then synthesize all instantiations of the model that have non-negligible probability and optimize the resulting shapes, as described in Section 5. We identify and reject instantiations that are very similar to shapes in the input dataset or to previous instantiations. This is achieved by summing over the distances of the geometric feature vectors of the corresponding components in the respective shapes, weighted by the sum of the component areas, and rejecting new instantiations that have a below-threshold distance to an input shape or to previous instantiations. Note that this pruning is optional, since these instantiations still correspond to plausible shapes in our experiments; we simply seek to avoid visually redundant shapes generated by shuffling very similar components around. We also reject synthesized shapes for which the component placement optimization fails.

Constrained shape synthesis. Our model can also be used to synthesize shapes subject to interactively specified constraints. For example, the user may want to synthesize shapes that have specific components, or components from particular categories, or components from some of the learned latent styles; or she may want to synthesize shapes that belong to some of the learned latent shape styles. To this end, we have created an interactive interface for visually specifying such constraints. The interface allows combining multiple types of constraints and is demonstrated in the accompanying video. To synthesize shapes subject to the provided constraints, we perform the deterministic search procedure described in Section 5.1 with the modification that partial assignments to constrained random variables assume values only from the range corresponding to the specified constraints. For example, if the user wishes to synthesize animal shapes with torsos from particular styles, the deterministic search will consider only the corresponding values for the torso style variable during the tree expansion.

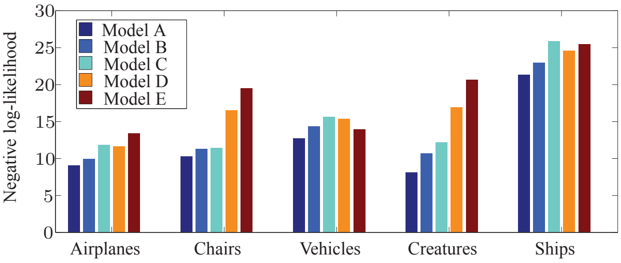


Figure 6: *Quantitative evaluation of generalization performance. Negative log-likelihood of our model (Model A) compared to weaker versions of the model. Lower negative log-likelihood indicates better generalization performance. Model B uses CPTs instead of sigmoid functions. Model C is trained with maximum likelihood instead of MAP. Model D does not use lateral edges between observed variables. Model E does not use latent variables, akin to Chaudhuri et al. [2011]. Our model achieves the best generalization across all datasets.*

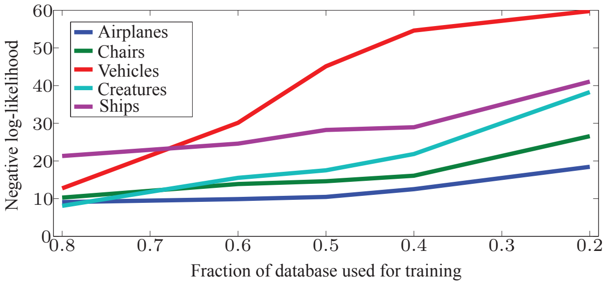


Figure 7: *Generalization performance with impoverished training sets. Performance decreases as the training set becomes less representative of the domain.*

7 Evaluation

We evaluated the presented model on five shape datasets obtained from publicly available 3D model libraries (Digimation Model Bank, Dosch 3D, and the furniture database of Wessel et al. [2009]). The datasets were compatibly segmented and labeled using the technique of Kalogerakis et al. [2010], assisted by manual segmentation and labeling. Table 1 gives the number of shapes in each dataset, the number of component categories, the number of individual components extracted from each dataset, and the number of new shapes synthesized for each dataset by our model. These synthesized shapes are shown alongside the training shapes in Figures 1, 14, 15, 16, and 17, as well as in the accompanying video. Figure 5 shows a histogram of the number of components used per synthesized shape and a histogram of the number of source shapes that contributed components per synthesized shape.

Generalization performance. A key question in the evaluation of a probabilistic model is how well it generalizes from the training data. A successful generative model will be able to not only reproduce instances from the training data, but synthesize genuinely novel plausible instances from the domain exemplified by the dataset. A standard technique for evaluating generalization performance is holdout validation, where the dataset is randomly split into a training set and a test set. The test set is withheld and only the training set is used for training the model. The trained model is then evaluated on the test set, by computing the probability assigned by the model to instances in the test set. Higher probability on the test set corresponds to better generalization performance. We

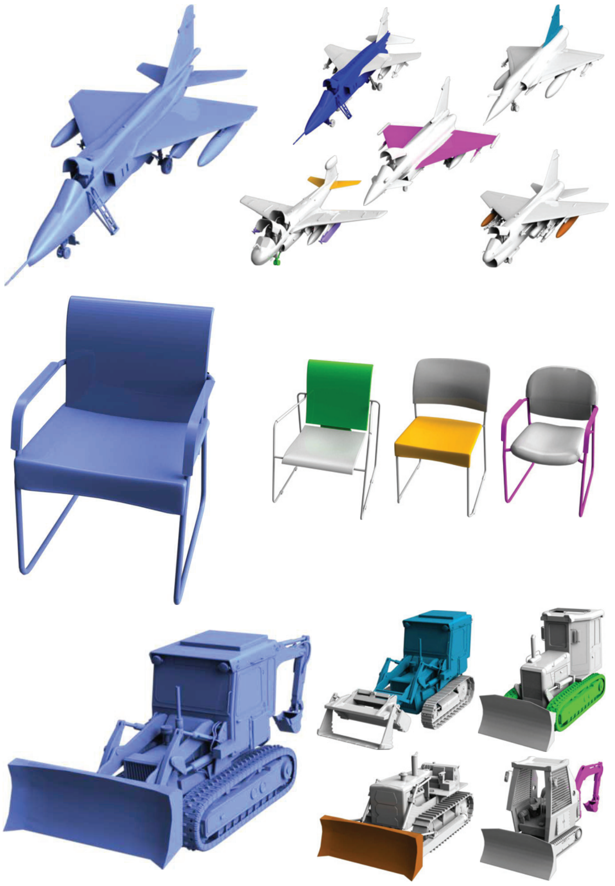
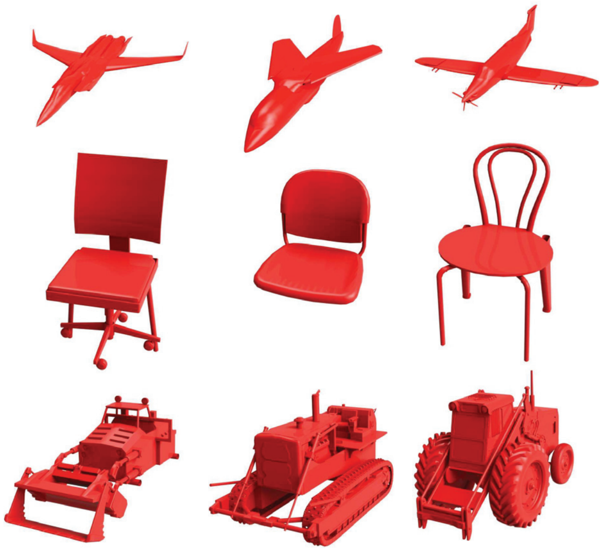


Figure 8: *Qualitative demonstration of generalization. Components from multiple source shapes (right) are combined by the probabilistic model to yield plausible new shapes (left, blue). Utilized components are highlighted in color in the source shapes.*

repeat the procedure with three random 80-20 training-test splits of the dataset, and take the geometric mean of the resulting probabilities. We compare the presented model to weaker models in which some of the components of the presented model are disabled. The results are shown in Figure 6. Each bar in the figure corresponds to the negative logarithm of the probability assigned to the test data by our model or a weaker variant; a lower value corresponds to better generalization performance.

We have also evaluated the performance of the model with impoverished datasets. To this end, we gradually changed the split ratios from 80-20 (train on 80% of the data, test on the remaining 20%) to 20-80 (train on 20% of the data, test on the remaining 80%). The results are plotted in Figure 7. Generalization performance degraded when the dataset made available for training became less representative of the overall domain. The rapid degradation for construction vehicles is due to the small size of the dataset: 20% of the data in this case corresponds to having only four examples.

Comparison to prior work. The probabilistic model developed by Chaudhuri et al. [2011] can also be used to synthesize complete novel shapes, although it was not designed for this purpose and in our experiments generally produced shapes of low plausibility (Figure 9). For quantitative evaluation against this prior model, we could not use holdout validation, since the model of Chaudhuri et al. has a different parameterization from ours and the log-probabilities of their model and ours are not directly comparable. Instead, we conducted an informal perceptual evaluation with 107 student volunteers recruited through a university mailing list. Each volunteer performed 30 pairwise comparisons in a Web-based survey. Each comparison was between images of two shapes from the same randomly chosen domain. The shapes were randomly sampled from three sets: original training shapes, shapes synthesized by our model and optimized by the procedure described in Section 5.2, and shapes synthesized by the model of Chaudhuri et al. and optimized by the same procedure. Each comparison involved images of shapes from two of these three sets. The images were sampled from the complete set of 321 images of training shapes, the complete set of 3152 images of shapes synthesized by our model, and 672 images of shapes synthesized by the model of Chaudhuri et al. The participants were asked to choose which of the two presented objects was more plausible, or indicate lack of preference. A total of 3210 pairwise comparisons were performed. The results are visualized in Figure 10. Shapes produced by our model were seen as more plausible than shapes produced by the prior model, with strong statistical significance.



(a) Chaudhuri et al. (b) No latent variables (c) No lateral edges

Figure 9: *Examples of shapes synthesized with alternative probabilistic models. (a) Shapes generated by the model of Chaudhuri et al. (b) Shapes generated by a variant of our model that has the same observed random variables and learning procedure but no latent variables. (c) Shapes generated by a variant of our model that has no lateral edges between observed random variables. The shapes have missing components or implausible combinations of components.*

Content of learned latent variables. Our model is distinguished by its use of latent variables to compactly parameterize the underlying causes of structural variability in complex domains. The domains of these variables are learned from data and different values of the variables are intended to represent different underlying styles of shapes and shape components. We visualize the styles learned by two of the variables for the set of chairs in Figures 12 and 13. Figure 12 shows high-probability shapes sampled by fixing the root variable in the learned model to each of its possible values. Figure 13 shows high-probability components sampled by fixing one of the lower-level latent variables (corresponding to backs of chairs) to each of its values.

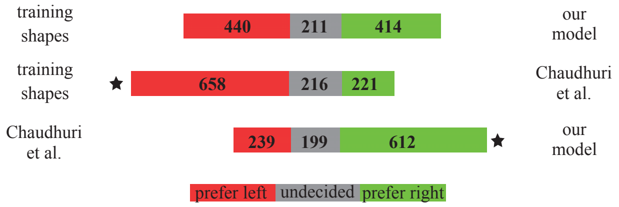


Figure 10: *Evaluation against prior work. 107 volunteers performed pairwise comparisons to evaluate the plausibility of shapes produced by our model against original training shapes and the shapes produced by the prior model of Chaudhuri et al. Results marked with a $\star$ are strongly statistically significant ($p < 10^{-7}$), according to a two-tailed single sample t-test.*

Lateral edges. Lateral edges capture strong correlations between features of different component categories. For example, in the case of construction vehicles, one of the learned lateral edges connects geometric features of the front tool with geometric features of the cabin. The construction vehicle shown in Figure 9(c) was synthesized by a model without lateral edges and has a bumper instead of a front scoop or another appropriate front tool. Likewise, for the chairs dataset, a learned lateral edge connects geometric features of the front legs with geometric features of the back legs. The chair in Figure 9(c) was synthesized by a model without lateral edges and has incompatible front and back legs. Overall, the number of lateral edges learned by the full model correlates with the number of component categories and the complexity of the domain. There are 9 learned edges for vehicles, 24 for creatures, 35 for chairs, 36 for planes, and 89 for ships.

Computational complexity and running times. The score evaluation (including parameter estimation) has complexity $O(LK)$, where L is the number of component categories and K is the number of input shapes. The total complexity of learning is $O(L^3K)$, taking into account the greedy search for the domain sizes of the hidden variables and the lateral edges. Our implementation is not parallelized, and was executed on a single core of an Intel i7-740 CPU. Learning took about 0.5 hours for construction vehicles, 3 hours for creatures, 8 hours for chairs, 20 hours for planes, and 70 hours for ships. For shape synthesis, enumerating all possible instantiations of a learned model takes less than an hour in all cases, and final assembly of each shape takes a few seconds.

8 Discussion

We presented a probabilistic model of component-based shape structure that can be used to synthesize new shapes from a domain demonstrated by a set of example shapes. Our processing pipeline assumes that the training shapes are compatibly segmented. Furthermore, the extraction of geometric features used for training assumes that the shapes are upright-oriented and front-facing. We employed semi-automatic procedures to segment and orient shapes, but the preprocessing stage still required manual effort. Advances upon current compatible shape segmentation and orientation techniques would be broadly beneficial [Fu et al. 2008; Kalogerakis et al. 2010; Huang et al. 2011; Sidi et al. 2011].

Our model also uses the simplifying assumption that the geometric features of components are normally distributed. This simplifies the learning procedure, but does not capture more complex variability of geometric features. Further, the model only learns linear correlations between component features. In addition, the component placement approach described in Section 5.2 is heuristic and can fail to produce visually pleasing results. Specifically, it does not optimize the orientation of components and does not prevent intersections between components, as shown in Figure 11. The development of more sophisticated approaches to component placement is thus an interesting avenue for future work. Analysis of the function of shapes is likewise an interesting direction that can enhance shape synthesis.

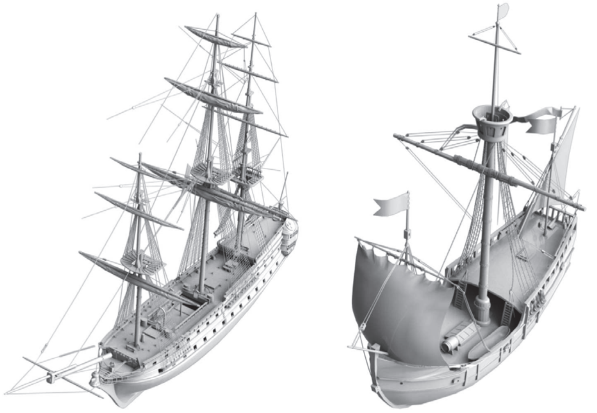


Figure 11: *The optimization procedure described in Section 5.2 may fail to yield plausible configurations of components. Some ropes on the left ship are misaligned because they are treated as a single component, and some components on the right ship intersect inappropriately.*

Finally, a significant avenue for future work is joint modeling of discrete structural variability together with continuous variability at the level of individual components. This can lead to learned models of continuous shape variability in increasingly complex real-world domains, which can enable new capabilities for shape reconstruction.

Acknowledgements

We are grateful to Aaron Hertzmann, Sergey Levine, and Philipp Krähenbühl for their comments on this paper, and to Tom Funkhouser for helpful discussions. This research was conducted in conjunction with the Intel Science and Technology Center for Visual Computing, and was supported in part by KAUST Global Collaborative Research and by NSF grants SES-0835601 and CCF-0641402.

References

ALIAGA, D. G., ROSEN, P. A., AND BEKINS, D. R. 2007. Style grammars for interactive visualization of architecture. *IEEE Transactions on Visualization and Computer Graphics 13*, 4.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics 22*, 3.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: shape completion and animation of people. *ACM Transactions on Graphics 24*, 3.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH*, ACM.

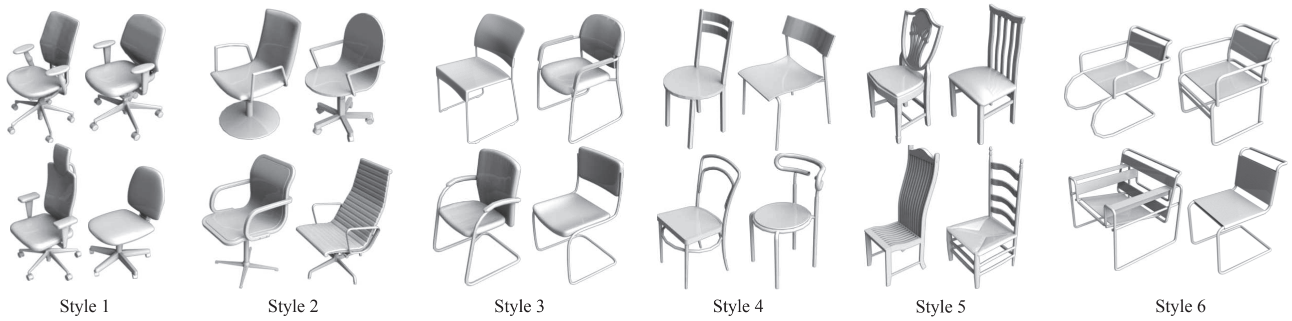


Figure 12: *Content of root latent variable for a model trained on the chair dataset. Different values of the variable correspond to learned styles of chair shapes. Four high-probability synthesized shapes are shown for each value.*

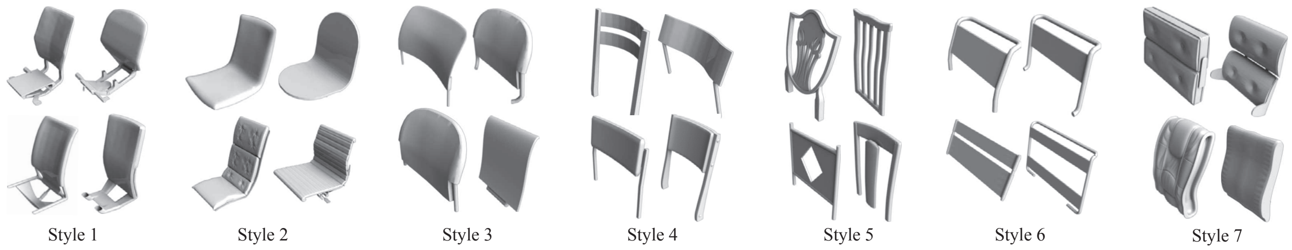


Figure 13: *Content of learned latent style variable for a specific component category (chair backs) in the chair dataset. Four high-probability components are shown for each value of the variable.*

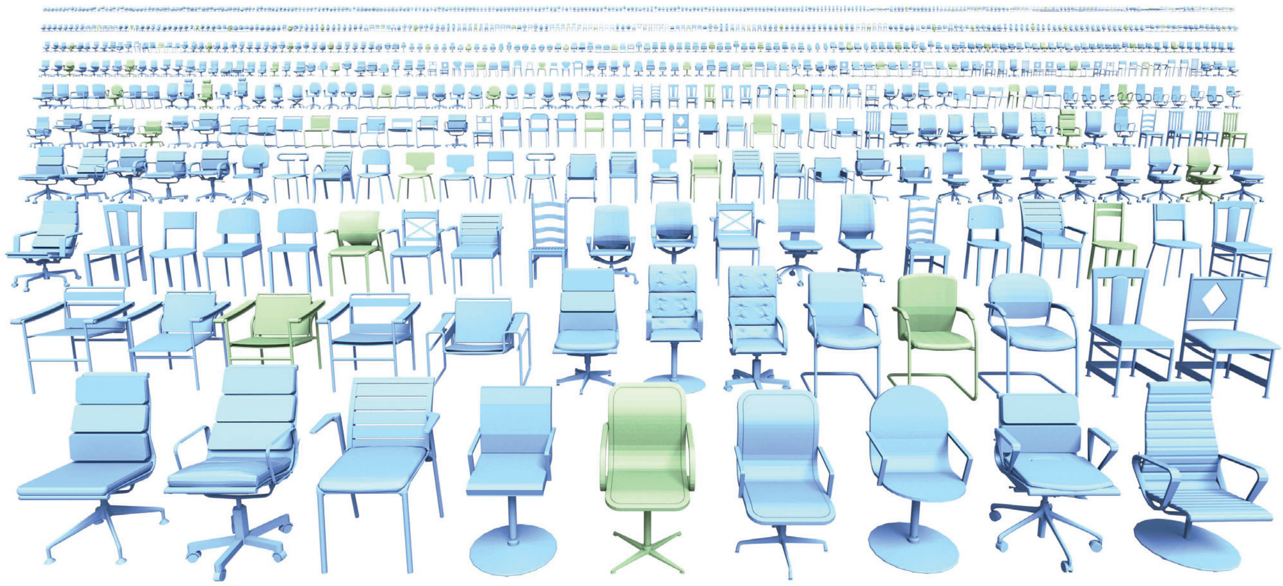


Figure 14: *Given 88 training chairs (green), our probabilistic model synthesizes 870 new chairs (blue).*

BOKELOH, M., WAND, M., AND SEIDEL, H.-P. 2010. A connection between partial symmetry and inverse procedural modeling. *ACM Transactions on Graphics 29*, 4.

BOUCHARD, G., AND TRIGGS, B. 2005. Hierarchical part-based visual object categorization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

CHAUDHURI, S., AND KOLTUN, V. 2010. Data-driven suggestions for creativity support in 3D modeling. *ACM Transactions on Graphics 29*, 6.

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3D modeling. *ACM Transactions on Graphics 30*, 4.

CHEESEMAN, P., AND STUTZ, J. 1996. Bayesian classification (autoclass): Theory and results. *Advances in Knowledge Discovery and Data Mining*.

CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. 2003. On visual similarity based 3D model retrieval. *Computer Graphics Forum 22*, 3.

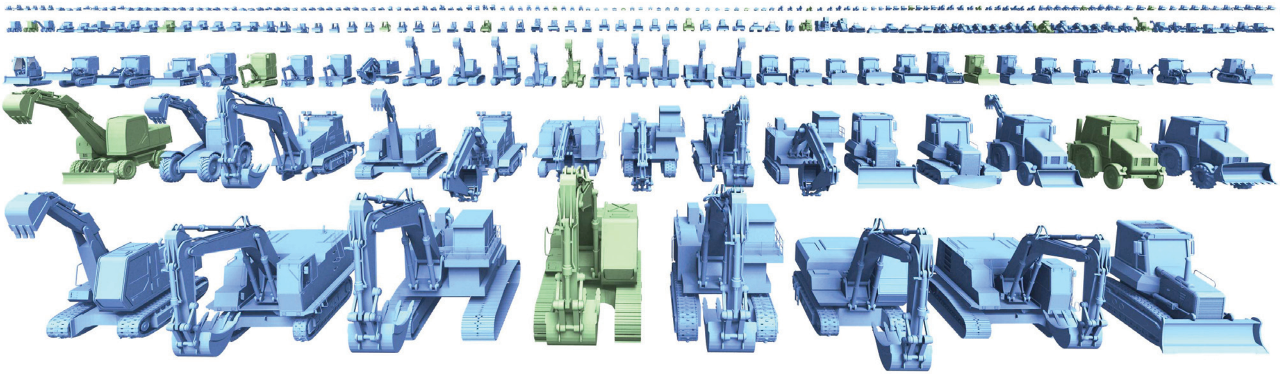


Figure 15: *Given 22 construction vehicles (green), our probabilistic model synthesizes 253 new vehicles (blue).*

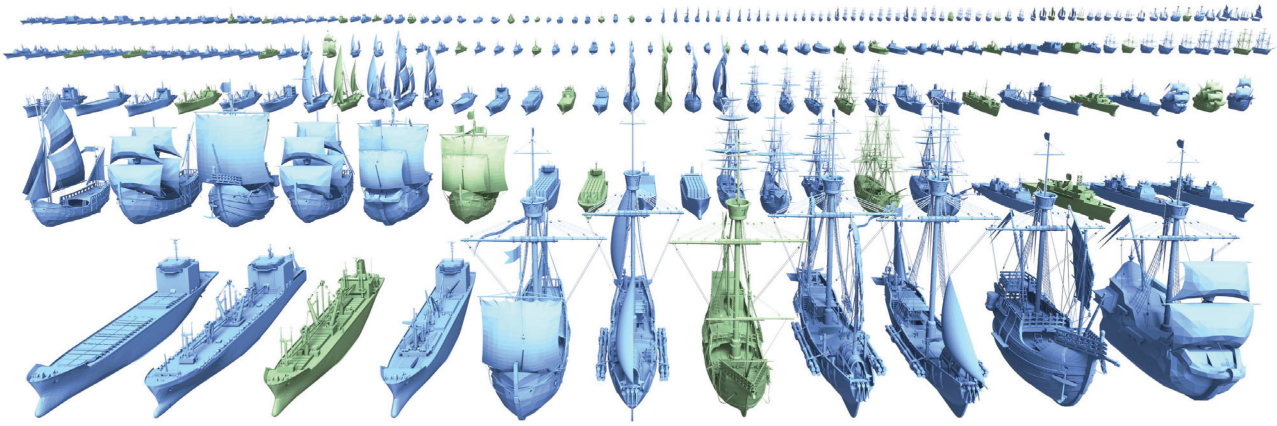


Figure 16: *Given 42 training ships (green), our probabilistic model synthesizes 199 new ships (blue).*

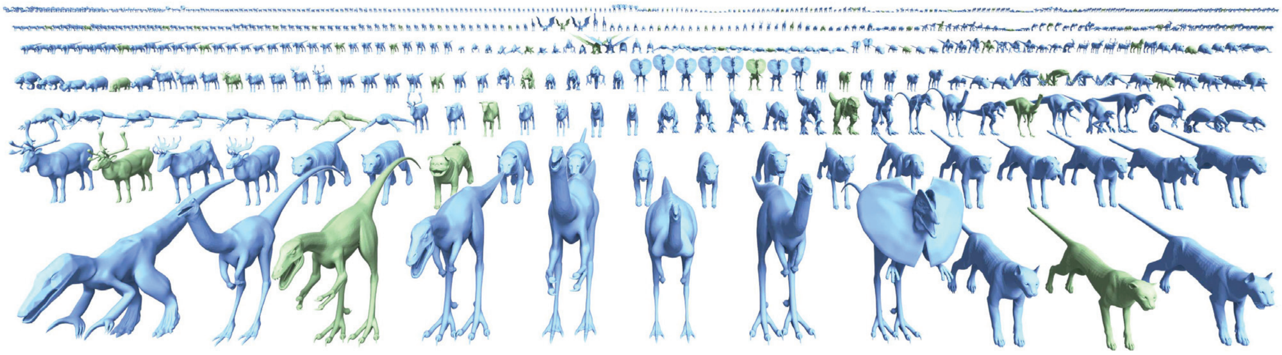


Figure 17: *Given 69 training creatures (green), our probabilistic model synthesizes 563 new creatures (blue).*

CHENNUBHOTLA, C., AND JEPSON, A. 2001. S-PCA: Extracting multi-scale structure from data. In *Proc. International Conference on Computer Vision*.

FIDLER, S., AND LEONARDIS, A. 2007. Towards scalable representations of object categories: Learning a hierarchy of parts. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

FU, H., COHEN-OR, D., DROR, G., AND SHEFFER, A. 2008. Upright orientation of man-made objects. *ACM Transactions on Graphics 27*, 3.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Transactions on Graphics 23*, 3.

HUANG, Q., KOLTUN, V., AND GUIBAS, L. 2011. Joint shape segmentation with linear programming. *ACM Transactions on Graphics 30*, 6.

JAIN, A., THORMAHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Exploring shape variations by 3D-model decomposition and part-based recombination. *Computer Graphics Forum 31*, 2.

Jin, Y., and Geman, S. 2006. Context and hierarchy in a probabilistic image model. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Kalogerakis, E., Hertzmann, A., and Singh, K. 2010. Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics 29*, 4.

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*.

Merrell, P., and Manocha, D. 2011. Model synthesis: A general procedural modeling algorithm. *IEEE Transactions on Visualization and Computer Graphics 17*, 6.

Merrell, P. 2007. Example-based model synthesis. In *Proc. Symposium on Interactive 3D Graphics*, ACM.

Ommer, B., and Buhmann, J. 2010. Learning the compositional nature of visual object categories for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 3.

Ovsjanikov, M., Li, W., Guibas, L., and Mitra, N. J. 2011. Exploration of continuous variability in collections of 3D shapes. *ACM Transactions on Graphics 30*, 4.

Ranzato, M. A., Susskind, J., Mnih, V., and Hinton, G. 2011. On deep generative models with applications to recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

Roux, N. L., Heess, N., Shotton, J., and Winn, J. 2011. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23.

Sharf, A., Blumenkrants, M., Shamir, A., and Cohen-Or, D. 2006. SnapPaste: an interactive technique for easy mesh composition. *Visual Computer 22*, 9.

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

Stava, O., Beneš, B., Měch, R., Aliaga, D., and Kristof, P. 2010. Inverse procedural modeling by automatic generation of L-systems. *Computer Graphics Forum 29*, 2.

Todorovic, S., and Ahuja, N. 2008. Unsupervised category modeling, recognition, and segmentation in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 12.

Tu, Z., Chen, X., Yuille, A. L., and Zhu, S.-C. 2005. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision 63*, 2.

Wessel, R., Blümel, I., and Klein, R. 2009. A 3d shape benchmark for retrieval and automatic classification of architectural data. In *Eurographics 2009 Workshop on 3D Object Retrieval*.

Xu, K., Zheng, H., Zhang, H., Cohen-Or, D., Liu, L., and Xiong, Y. 2011. Photo-inspired model-driven 3D object modeling. *ACM Transactions on Graphics 30*, 4.

Zhu, L. L., Lin, C., Huang, H., Chen, Y., and Yuille, A. L. 2008. Unsupervised structure learning: Hierarchical recursive composition, suspicious coincidence and competitive exclusion. In *Proc. European Conference on Computer Vision*.

A Geometric Preprocessing for Components

First, the meshes are oriented so that $+Z$ is the upward direction and $+Y$ is the front-facing direction. Sparse-PCA [Chennubhotla and Jepson 2001] on surface samples of a specified component is used to extract principal axes of each mesh. Sparse-PCA tends to choose axes that align with the latent XYZ frame, which is appropriate since the meshes are usually already oriented to some permutation of the latent axes. Specified principal axes are aligned to $+Z$ and $+Y$. For example, the upward direction of chairs is determined by the SPCA axis that corresponds to the smallest variance in the points of the seats and for which the center of mass of backs has positive y-axis value. If this process fails, the meshes are oriented manually.

Then, for each shape component c from each source mesh m in the repository, we extract a high-dimensional feature vector containing: *a*) the 3D scale vector of the oriented bounding box of the component; *b*) histograms of 4, 8 and 16 uniform bins for principal curvatures κ_1 and κ_2 (the curvatures are 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); *c*) histograms of 4, 8 and 16 uniform bins of the shape diameter over the surface of c, and of its logarithmized versions w.r.t. normalizing parameters 1, 2, 4 and 8; *d*) 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, $s_1/s_2 + s_1/s_3$, $s_1/s_2 + s_2/s_3$, $s_1/s_3 + s_2/s_3$; *e*) lightfield descriptor values computed as in [Chen et al. 2003] (since the meshes are oriented consistently, these descriptors can be compared without searching over aligning transforms).

We take the average of the above feature vectors of the shape components belonging to the same category. We perform PCA on the matrix of lightfield features, and also on the matrix containing the descriptors (b-d) to reduce the overall dimensionality of the features (retaining 75% of the variance in the data). The final feature vector $\mathbf{C}_l$ contains the 3D scale vector and the projected low-dimensional features.

Finally, for each component we detect slots that connect them to other components (Figure 4(a)). For components obtained by cutting topologically manifold meshes along edge loops, the slots are simply these edge loops. For all other components, we mark vertices close to a component of a different category as slot vertices, using a threshold equal to $1/64$ of the radius of the bounding sphere of the mesh. If no such vertices are found, the threshold is doubled until at least ten slot vertices are found. We also automatically extract ground contacts for each component, if these exist. These contacts are extracted by finding the vertices whose distance to the ground plane is below the threshold used to detect slot vertices.

A Probabilistic Model for Component-Based Shape Synthesis

Supplementary Material

Evangelos Kalogerakis Siddhartha Chaudhuri Daphne Koller Vladlen Koltun

Stanford University

Likelihood evaluation and parameter estimation. The first task in evaluating the score for a test structure G is to estimate the MAP parameters $\tilde{\Theta}_G$ by maximizing the product

$$\tilde{\Theta}_G = \arg\max_{\Theta} P(\mathbf{O} \mid G, \Theta) P(\Theta \mid G).$$

Here, the first term is the likelihood function and the second term is the parameter prior. Unfortunately, the product cannot be optimized in closed form, because the likelihood is a function of the unknown values of the hidden random variables:

$$P(\mathbf{O} \mid G, \Theta) = \prod_k \sum_{R_k, \mathbf{S}_k} P(O_k, R_k, \mathbf{S}_k \mid \Theta).$$

Therefore, we use the expectation-maximization (EM) algorithm to optimize the parameters iteratively. The algorithm starts with an initial assignment to the values of the shape style R and component style S_l for each category label l. The initial assignment is obtained by k-means clustering on the feature space $\{\mathbf{C}_l, \mathbf{D}_l\}$ under the Euclidean metric to obtain initial values of S_l for each training example. Then we perform k-medoids on the feature space $\{\mathbf{S}, \mathbf{N}\}$ under the Hamming metric to get initial values for the shape style R for each training example. In both cases, we repeat the clustering with random starting points until we find the assignments that minimize the sum of distances of the data points to their closest cluster centers.

The EM algorithm alternates between two steps: the M-step in which the parameters $\tilde{\Theta}_G$ are re-estimated based on the current assignments to the hidden random variables, and the E-step in which the algorithm performs inference to find probabilistic assignments to the hidden variables for each training example. In the M-step, the MAP estimates are computed using a Dirichlet prior distribution for the parameters of the CPDs of the discrete random variables and a normal-Wishart distribution for the parameters of the CPDs of the continuous random variables. The updates to the parameters are computed as follows:

Given the probabilities estimated in the previous E-step, we compute for each shape k:

$$\begin{aligned}
M[R = r] &= \sum_k P(R_k = r \mid O_k), \\
M[S_l = s] &= \sum_k P(S_{l,k} = s \mid O_k), \\
M[S_l = s, R = r] &= \sum_k P(S_{l,k} = s, R_k = r \mid O_k).
\end{aligned}$$

The parameters for the probability table for R are estimated as:

$$q_r = \frac{M[R = r] + \alpha}{K + \alpha |\mathcal{R}|},$$

where the hyperparameter α is set to 0.1. Then for each remaining discrete random variable T with a single parent U and every value u of U, the corresponding CPT parameters are estimated:

$$q_{t|u} = \frac{M[T = t, U = u] + \alpha}{M[U = u] + \alpha |\mathcal{T}|}.$$

For discrete random variables with multiple parents $\mathbf{U}$, we fit sigmoid CPDs using iterative reweighted least-squares [Bishop 2006]. Then we compute for every value $\mathbf{u}$ in the value space $\mathcal{U}$ of $\mathbf{U}$:

$$q_{t|\mathbf{u}} = \frac{P(T = t \mid \mathbf{U} = \mathbf{u}) + \alpha / K}{1 + \alpha |\mathcal{T}| / K},$$

where $P(T = t \mid \mathbf{U} = \mathbf{u})$ is the output of the estimated sigmoid functions.

For the case of a continuous random variable $\mathbf{C}_l$ with no continuous parents, the parameters of its conditional linear Gaussians are updated as follows [Gauvain and Lee 1994; Geiger and Heckerman 1994; Koller and Friedman 2009] (we omit the conditioning on discrete parents for notational clarity):

$$\begin{aligned}
\boldsymbol{\phi}_l &= \frac{\boldsymbol{\mu}_l + \sum\limits_{k=1}^{K} P(S_{l,k} = s) \mathbf{C}_{l,k}}{1 + M[S_l = s]}, \\
\Sigma_l \equiv \Sigma_{ll} &= \frac{\Omega + \sum\limits_{k=1}^{K} P(S_{l,k} = s)(\mathbf{C}_{l,k} - \boldsymbol{\phi}_l)(\mathbf{C}_{l,k} - \boldsymbol{\phi}_l)^{\mathrm{T}}}{1 + M[S_l = s]} \\
&\quad + \frac{(\boldsymbol{\mu}_l - \boldsymbol{\phi}_l)(\boldsymbol{\mu}_l - \boldsymbol{\phi}_l)^{\mathrm{T}}}{1 + M[S_l = s]}.
\end{aligned}$$

where $\Omega = 10^{-4}\mathbf{I}$ is a regularization parameter. If the number of the above parameters is larger than the number of training instances, we only consider diagonal covariance matrices. The parameter $\boldsymbol{\mu}_l$ is set to be the mean of the features in the component category l. If $\mathbf{C}_l$ has continuous parents $\{\mathbf{C}_{l'}\}$, then the parameters are updated as follows:

$$\begin{aligned}
\Sigma_{ll'} &= \frac{\sum\limits_{k=1}^{K} P(S_{l,k} = s)(\mathbf{C}_{l,k} - \boldsymbol{\phi}_l)(\mathbf{C}_{l',k} - \boldsymbol{\phi}_{l'})^{\mathrm{T}}}{M[S_l = s]}, \\
\boldsymbol{\phi}_{l,0} &= \boldsymbol{\phi}_l - \Sigma_{ll'} \Sigma_{l'l'}^{-1} \boldsymbol{\phi}_{l'}, \\
\boldsymbol{\phi}_{l,l'} &= \Sigma_{ll'} \Sigma_{l'l'}^{-1} \\
\Sigma_l &= \Sigma_{ll} - \Sigma_{ll'} \Sigma_{l'l'}^{-1} \Sigma_{l'l}.
\end{aligned}$$

If the number of the parameters in $\boldsymbol{\phi}_{l,l'}$ is larger than the number of training instances, we use only the three first scale features in $\mathbf{C}_l$ for estimating $\boldsymbol{\phi}_{l,l'}$.

In the E-step, inference is performed to estimate probabilities of assignments $P(R_k \mid O_k)$ and $P(S_{l,k} \mid O_k)$ for each training data instance. Inference for the hidden random variables can be performed using variable elimination. Given observed data O_k for a source shape k, we compute $P(R, S_l \mid O_k)$ for a label $l \in \mathcal{L}$ using the following formula:

$$P(R, S_l \mid O_k) = \frac{P(R, S_l, O_k)}{P(O_k)},$$

where

$$\begin{aligned}P(R, S_l, O_k) = \sum_{\mathbf{S}\backslash\{S_l\}} P(R) \prod_{l'\in\mathcal{L}} P(S_{l'} \mid R) P\left(N_{l',k} \mid R, \pi(N_{l'})\right)\\ P\left(\mathbf{C}_{l',k} \mid S_{l'}, \pi(\mathbf{C}_{l'})\right)\\ \prod_{l^* \text{ adj } l'} P\left(\mathbf{D}_{l',l^*,k} \mid S_{l'}, \pi(\mathbf{D}_{l',l^*})\right)\\ = P(R)\, P(S_l \mid R)\, P(N_{l,k} \mid R, \pi(N_l))\\ P(\mathbf{C}_{l,k} \mid S_l, \pi(\mathbf{C}_l)) \prod_{l' \text{ adj } l} P(\mathbf{D}_{l,l',k} \mid S_l, \pi(\mathbf{D}_{l,l'}))\\ \prod_{l^*\in\mathcal{L}, l^*\neq l} \sum_{S_{l^*}} P(S_{l^*} \mid R) P(N_{l^*,k} \mid R, \pi(N_{l^*}))\\ P(\mathbf{C}_{l^*,k} \mid S_{l^*}, \pi(\mathbf{C}_{l^*}))\\ \prod_{l^{**} adj. l^*} P(\mathbf{D}_{l^*,l^{**},k} \mid S_{l^*}, \pi(\mathbf{D}_{l^*,l^{**}})).\end{aligned}$$

and

$$P(O_k) \;=\; \sum_{R,S_l} P(R, S_l, O_k).$$

In the above formulas $\pi(.)$ denotes the parents of a variable and $\mathbf{D}_l = \{D_{l,l*}\}$ represents the discrete features for each label l - in our case, these discrete features store the number of adjacent components for each label l^* adjacent to label l.

The EM algorithm iterates until convergence: it is stopped when the parameters differ less than 0.001% at maximum compared to the previous iteration.

Likelihood of the fictitious dataset $\mathbf{O}^*$. The term $P(\mathbf{O}^* \mid G, \tilde{\Theta}_G)$ is the likelihood of the estimated MAP parameters assuming the training dataset was completed with probabilistic assignments to the hidden random variables. Given these probabilistic assignments found in the last step of the EM algorithm, the likelihood is computed as follows:

$$\begin{aligned}P(O^* \mid G, \tilde{\Theta}_G) = \prod_R P(R)^{M[R]}\\ \prod_{l\in\mathcal{L}} \Bigg[\prod_R \prod_{S_l} P(S_l \mid R)^{M[S_l,R]}\\ \prod_{R,\pi(N_l)} \prod_{N_l} P(N_l \mid R, \pi(N_l))^{M[N_l,R,\pi(N_l)]}\\ \prod_{l' \text{ adj } l} \prod_{S_l,\pi(\mathbf{D}_{l,l'})} \prod_{\mathbf{D}_{l,l'}} P(\mathbf{D}_{l,l'} \mid S_l, \pi(\mathbf{D}_{l,l'}))^{M[\mathbf{D}_{l,l'},S_l,\pi(\mathbf{D}_{l,l'})]}\\ \prod_{S_l,\pi(\mathbf{C}_l)} \prod_k P(\mathbf{C}_{l,k} \mid S_l, \pi(\mathbf{C}_{l,k}))^{P(S_{l,k}|O_k)} \Bigg],\end{aligned}$$

where $M[\cdot]$ measures the number of times one or more discrete random variables take particular values. For the case of hidden random variables, this is replaced by the expected counts, e.g. $M[R = r] = \sum_k P(R_k = r \mid O_k)$.

Marginal likelihood of the fictitious dataset $\mathbf{O}^*$. The marginal likelihood $P(\mathbf{O}^* \mid G)$ is the likelihood times the parameter prior integrated over the parameter space, assuming the training dataset was completed with probabilistic assignments to the hidden random variables. For complete datasets, the marginal likelihood can be evaluated analytically in the case of Dirichlet priors for the discrete random variables and normal-Wishart priors for the conditional Gaussian random variables [Geiger and Heckerman 1994]:

$$\begin{aligned}P(\mathbf{O}^* \mid G) = \int_\Theta P(\mathbf{O}^* \mid \Theta, G) P(\Theta \mid G)\, \mathrm{d}\Theta\\ = \frac{\Gamma(|R|\alpha)}{\Gamma(|R|\alpha + K)} \prod_R \frac{\Gamma(\alpha + M[R])}{\Gamma(\alpha)}\\ \prod_{l\in\mathcal{L}} \Bigg[\prod_R \frac{\Gamma(|\mathcal{S}_l|\alpha)}{\Gamma(|\mathcal{S}_l|\alpha + M[R])} \prod_{S_l} \frac{\Gamma(\alpha + M[S_l, R])}{\Gamma(\alpha)}\\ \Bigg(\prod_{R,\pi(N_l)} \frac{\Gamma(|\mathcal{N}_l|\alpha)}{\Gamma(|\mathcal{N}_l|\alpha + M[R, \pi(N_l)])}\\ \prod_{N_l} \frac{\Gamma(\alpha + M[N_l, R, \pi(N_l)])}{\Gamma(\alpha)} \Bigg)\\ \Bigg(\prod_{l' \text{ adj } l} \prod_{S_l,\pi(\mathbf{D}_{l,l'})} \frac{\Gamma(|\mathcal{D}_{l,l'}|\alpha)}{\Gamma(|\mathcal{D}_{l,l'}|\alpha + M[S_l, \pi(N_l)])}\\ \prod_{\mathbf{D}_{l,l'}} \frac{\Gamma(\alpha + M[\mathbf{D}_{l,l'}, S_l, \pi(N_l)])}{\Gamma(\alpha)} \Bigg)\\ \prod_{S_l,\pi(\mathbf{C}_l)} \frac{T_{\mathbf{C}_l}(\mathbf{C}_l, S_l, \pi(\mathbf{C}_l))}{T_{\mathbf{C}_l}(\pi(\mathbf{C}_l), S_l, \pi(\mathbf{C}_l))} \Bigg],\end{aligned}$$

where $\Gamma(\cdot)$ is the gamma function, and

$$\begin{aligned}T_{\mathbf{C}}(\mathbf{u}, S_l, \mathbf{v}) = 2\pi^{(-pM[S_l,\pi(\mathbf{C})=\mathbf{v}]/2)} \left(\frac{1}{1 + M[S_l, \pi(\mathbf{C}) = \mathbf{v}]} \right)^{\frac{p}{2}}\\ \cdot \frac{c(p,p)}{c(p, p + M[S_l, \pi(\mathbf{C}) = \mathbf{v}])} |\Omega|^{\frac{p}{2}} |\Sigma_{\mathbf{u}}|^{-(p+M[S_l,\pi(\mathbf{C})=\mathbf{v}])/2}.\end{aligned}$$

Here p is the number of dimensions of $\mathbf{u}$, and

$$c(p,t) = \left[2^{\frac{dt}{2}} \pi^{\frac{p(p-1)}{4}} \prod_{i=1}^{p} \Gamma\left(\frac{t+1-i}{2} \right) \right]^{-1}.$$

References

BISHOP, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag.

GAUVAIN, J., AND LEE, C. 1994. Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *IEEE Transactions on Speech and Audio Processing 2*, 291–298.

GEIGER, D., AND HECKERMAN, D. 1994. Learning Gaussian Networks. Tech. Rep. MSR-TR-94-10.

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