A Probabilistic Model for Exteriors of Residential Buildings

LUBIN FAN and PETER WONKA
King Abdullah University of Science and Technology

We propose a new framework to model the exterior of residential buildings. The main goal of our work is to design a model that can be learned from data that is observable from the outside of a building and that can be trained with widely available data such as aerial images and street view images. First, we propose a parametric model to describe the exterior of a building (with a varying number of parameters) and propose a set of attributes as a building representation with fixed dimensionality. Second, we propose a hierarchical graphical model with hidden variables to encode the relationships between building attributes and learn both the structure and parameters of the model from the database. Third, we propose optimization algorithms to generate three-dimensional models based on building attributes sampled from the graphical model. Finally, we demonstrate our framework by synthesizing new building models and completing partially observed building models from photographs.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Modeling Packages*

General Terms: Algorithms

Additional Key Words and Phrases: Urban modeling, graphical models, urban reconstruction

ACM Reference Format:

Fan, L. and Wonka, P. 2016. A Probabilistic Model for Exteriors of Residential Buildings. ACM Trans. Graph. XX, X, Article XXX (Month 2016), 13 pages.
DOI = 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

1. INTRODUCTION

In the last two decades, we could observe a drastic improvement of remote sensing technology and Internet-based mapping. As a result, large amounts of urban data are available and accessible via the most popular mapping sites Google maps and Bing maps. In this context, a major challenge in urban modeling is to learn the structures and geometries of urban environments from available data.

Fig. 1: An application of our building model. Starting from a single image (Input), a user can specify parts of a building mass model and mark shapes (windows and doors) on the observed part of a building facade in a rectified image (Element Selection). Our framework can complete the missing parts of the building (mass model and facades). A 3D rendering of the completed building is shown on the bottom.

In this paper, we tackle a part of this grand challenge and we propose a probabilistic model for building exteriors of residential buildings. To motivate our design choices, we first elaborate on the design goals of such a computational building model:

(1) *Learning:* It should be possible to learn the model from available remote sensing data. We would like to rely exclusively on orthographic aerial images and street-view images that are widely available for most parts of the world. This restricts the data that we can consider to things that are observable from the outside and we cannot rely on interior information such as floor plans [Merrell et al. 2010].
(2) *Sampling:* The model should be generative. After the model is learned from the data, it should be possible to sample a large variety of buildings from this model that reflects the variety in the available data.
(3) *Hard Constraints:* The model should be able to handle hard geometric constraints. While modeling buildings from scratch is the major application of a computational building model,

This publication is based upon work supported by the Office of Sponsored Research (OSR) under Award No. OCRF-2014-CRG3-62140401 and the KAUST Visual Computing Center.
Authors' addresses: L. Fan, KAUST, email: lubinfan@gmail.com; P. Wonka, KAUST, email: pwonka@gmail.com.

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

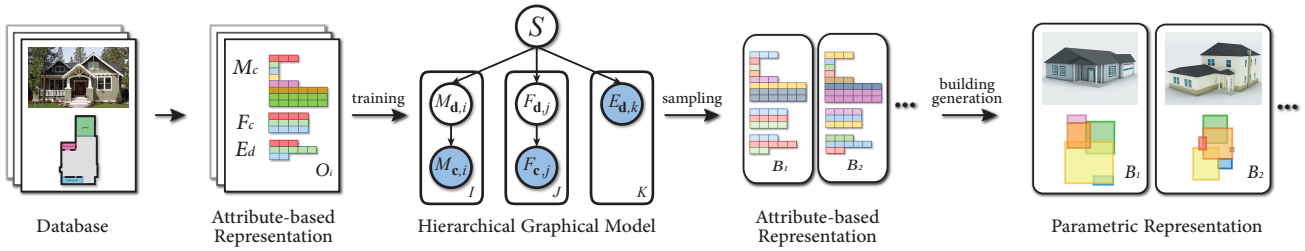

Fig. 2: Overview of our framework.

another important application is to complete partial buildings. Often a user might only have a single photograph or a building can be only partially observed because of occlusions due to vegetation. In a completion application, we would like to reconstruct the visible parts of a building from photographs and then to complete the rest of the building using the partial reconstruction as hard constraints.

Our model can fulfill these three design goals. It is inspired by previous work on assembly-based shape synthesis by Kalogerakis et al. [2012]. Their model is designed for shapes that mainly have components with unique labels. This is not the case for building models, where many components share the same label, e.g., windows. To solve this problem, we propose a set of attributes to describe a building and design a suitable hierarchical structure of the graphical model to represent the joint probability distribution of these attributes. Given a set of building attributes, we also propose optimization algorithms to generate three-dimensional building models. The contribution of our work are the design of the probabilistic model and the optimization methods for building generation. The learning algorithm for the graphical model is adapted from previous work [Koller and Friedman 2009; Kalogerakis et al. 2012]. We train our model on a database of conventional contemporary houses, and we demonstrate our model by synthesizing new building models and completing partially observed building models from photographs. In the next section, we will discuss related work and explain why current works do not handle all three design goals at the same time.

2. RELATED WORK

Procedural modeling. Generative building models have been studied in the context of procedural modeling, e.g. using shape grammars [Wonka et al. 2003; Müller et al. 2006; Schwarz and Müller 2015] or optimization [Lin et al. 2011; Bao et al. 2013]. We refer the reader to recent surveys on procedural modeling [Vanegas et al. 2010; Smelik et al. 2014] and urban reconstruction [Musialski et al. 2013] for discussions on a variety of models. In general, existing procedural models adequately fulfill design goal 2 (sampling), but they are not sufficiently developed to deal with design goals 1 (learning) and 3 (hard constraints). One open challenge in procedural modeling is how to derive a model that fulfills given hard constraints. Talton et al. [2011] propose an algorithm that makes grammars conform to high-level goals and soft constraints by MCMC. The extension to hard constraints (goal 3) and finding an efficient algorithm are still open questions. An additional challenge is inverse procedural modeling discussed next.

Inverse procedural modeling. One challenge in inverse procedural modeling is to find a procedural description for a single input model. Bokeloh et al. [2010] propose a solution for low-level shape understanding of a single 3D model to find a set of replaceable parts by partial symmetry detection. Zhang et al. [2013] and Wu et al. [2014] work on segmented facades to extract a compact model by optimizing a symmetry heuristic or the description length of a context-free grammar, respectively. An earlier effort in this direction was presented in the context of vector graphics [Šťava et al. 2010] using L-systems. The more difficult challenge in inverse procedural modeling is to find a procedural description that can encode a shape space given by a set of representative models. There are some initial ideas in this direction, e.g. [Talton et al. 2012; Martinovic and Van Gool 2013], that adapt the Bayesian model merging algorithm that was originally presented in the context of natural language processing. Still, there are many obstacles in making this approach work on interesting examples. While Bayesian model merging is elegant and simple to implement, the learned grammar creates many undesirable models. A better solution might need orders of magnitude more training examples or a more expressive procedural model than context-free grammars. We therefore conclude that design goal 1 (learning) is still an open question in procedural modeling.

Shape space interpolation. Our work is also related to shape space interpolation, which aims to generate novel shapes by interpolating examples from a given shape space. A key problem in shape space methods is to find a joint parameterization. This works well for shape collections with similar elements, e.g., fonts, but it is harder for shape collections with few distinct and labeled components, e.g., airplanes and chairs. Kim et al. [2013] learn shape templates from a large collection of shapes automatically. These templates can be used to explain the input model collection and synthesize new shapes. This idea is then extended by Averkiou et al. [2014] by embedding the feature vector of the templates into 2D subspaces. Users can explore the parameterized space and synthesize novel models by deforming and combining parts from input shapes. Yang et al. [2011] explore the shape space of constrained meshes. Bao et al. [2013] build on this idea to explore and interpolate novel building layouts from a set of layouts by optimizing building constraints. Our work represents a building as unions of boxes, similar to [Bao et al. 2013]. Complementary to this previous work, we focus on learning variations from a database and on completing partial buildings with sampled constraints. The idea of shape space exploration is also applied to fonts [Campbell and Kautz 2014]. They learn a generative manifold of standard fonts to explore and synthesize new fonts. The challenge in parameterizing building models is that many shapes have the same label. Additional work on shape space interpolation methods would be needed to satisfy the design goals (1-3) in the context of building modeling.

Probabilistic models. Probabilistic models seem to be the most promising direction for our work because they can be used to learn building attributes from observations and then to generate

new building models (goal 1). Merrel et al. [2010] learn a directed graphical model that encodes the relationships between rooms for interior building modeling. We would like to create a complementary model for the exterior of a building. What makes our task more challenging is the fact that we have many elements in the model with the same label. Merrell et al. also have to deal with label ambiguity for bedroom labels, but this challenge and its solution is not described in the text. However, with limited ambiguity, one can imagine a brute-force solution to this problem. Rather than using a simple directed graphical model, we propose to use a hierarchical graphical model with hidden variables. Our probabilistic model is inspired by the generative models for assembly-based modeling [Chaudhuri et al. 2011; Kalogerakis et al. 2012]. Chaudhuri et al. [2011] develop a probabilistic representation of shape structures that can be used to suggest relevant components during an interactive assembly-based modeling session. Kalogerakis et al. [2012] learn a distribution over a part-based model encoding multiple object styles, part cardinalities, part shapes, and part placements. They use this model for shape synthesis. Similar to the graphical model employed in this paper, they also use hidden variables in their model. Chaudhuri et al. [2013] improve the model by learning relative attributes for design components that reflect the high-level intent people may have for creating content in a domain.

3. OVERVIEW

Our framework consists of the following components:

—First, we introduce a parametric building model that can describe a building by a varying number of parameters. To make the model suitable for learning, we also present a fixed-dimensional, attributes-based building representation (see Sec. 4).

—Second, we describe how to structure the attributes of the attribute-based building representation in a hierarchical graphical model with hidden variables. We also describe how the graphical model can be learned from data and our database (see Sec. 5).

—Third, we propose an optimization-based approach to convert a sampled attribute-based building representation into a parametric building representation. We use a separate optimization for the mass model and of the roofs and the facades (see Sec. 6).

—Fourth, we evaluate our graphical model and show applications in building synthesis and model completion for partially reconstructed buildings (see Sec. 7).

Figure 2 illustrates how these components work together. Starting from a database of buildings, we extract an attribute-based representation of each building. Then, we learn a graphical model from these building representations. To synthesize new buildings or complete partial buildings, we sample an attribute-based representation from the graphical model. Finally, we derive a parametric representation by optimization.

4. A PARAMETRIC BUILDING MODEL

4.1 A Parametric Building Representation

A building consists of its *mass model*, *facades*, and the *roof*. The mass model of the building is defined as the union of a set of (overlapping) boxes, $\bigcup b_i$, (see Figure 3 top). We parameterize each box, b_i, using its position attributes (center (x_i, y_i)) and size attributes (width w_i, depth d_i, and height h_i). A box also has a style parameter to distinguish between the garage, porch, and general building box. Additionally, each box has the following roof parameters: a label describing the style of the roof, an angle encoding the pitch, and a flag encoding the orientation. The facades of a building are labeled by their orientation, i.e., front, back, left, and right (see Figure 3 bottom). Each building side consists of a set of facade pieces, $\{f_{i,j}\}$. The facade elements (e.g., doors, windows) are represented by rectangles on the facade. Each element is defined by its width, height, location inside the facade, and a label describing its style.

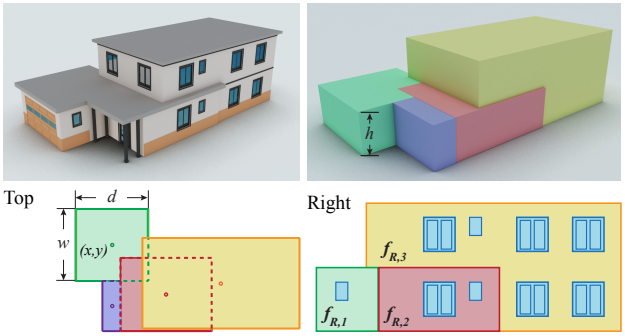

Fig. 3: We represent a building (top left) as the union of a set of boxes (top right), where each box is parametrized by its position attributes (x, y) and size attributes (w, d, h). The facades are labeled by their orientation, i.e., front, back, left, and right. Each building side consists of a set of facade pieces $\{f_{i,j}\}$. The right building side is shown on the bottom right.

The parametric building model describes a building by a list of parameters. The challenge of this parametric model in the context of machine learning is that the length of the list of parameters for each building model can vary. Further, there is no clear correspondence between the individual parameters, because most of the boxes in the mass model and the windows on the facades cannot be assigned a unique label based on their position, function, or other semantic meaning. Therefore, we propose a fixed set of attributes that can describe a building model in the next subsection. Note that the lack of distinct labels is the main reason that finding good representations for the exterior of a building is more challenging than describing its interior [Merrell et al. 2010].

4.2 An Attribute-based Building Representation

Finding a fixed set of attributes to describe a building is essentially a modeling task. There are four things we look for in a good set of attributes. First, the attributes should be descriptive enough such that mapping a parametric building description to a set of attributes and back gives a similar looking building. Second, the attributes need to be distinct (directly comparable). Third, it has to be possible to construct the attribute-based representation directly from easily available building descriptions without the need of manually reconstructing a large number of 3D buildings. Fourth, the model should be as simple as possible because the complexity determines how much data is needed during the learning step. In practice, we used an iterative design process to obtain the proposed set of attributes. For example, we initially used a building volume attribute that was not descriptive enough to encode information about the geometry of the mass model. We therefore replaced this attribute with two attributes in our final model: floor area ratio and area ratio of floors. Another example is an initial attempt to encode the complexity of the boundary with a discrete variable. This resulted in a value space that was too large and would have required too much training data. The attributes can be classified into three categories:

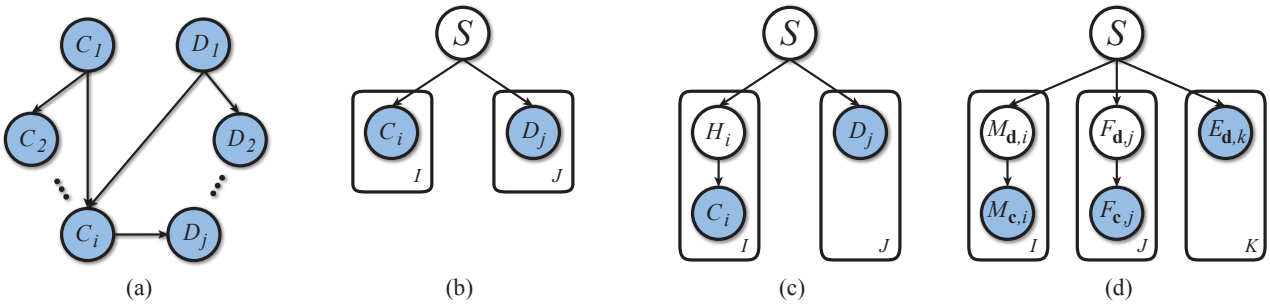

Fig. 4: Design choices for the graphical model. (a) A simple directed graphical model with continuous variables $\{C_i\}$ and discrete variables $\{D_j\}$. (b) A hidden variable S is introduced into the model to reduce the learned lateral edges. (c) Additional hidden variables $\{H_i\}$ are introduced. (d) To endow the model with more semantics, we separate mass model and facade variables to arrive at the final model.

—Mass model attributes ($\mathbf{M_c}$). We define five attributes to describe the basic configuration of a building. They are size of the bounding box ($M_{\mathbf{c},bbox}$), building coverage ratio ($M_{\mathbf{c},bcr}$), boundary complexity ($M_{\mathbf{c},bdry}$), floor area ratio ($M_{\mathbf{c},far}$), and area ratio of floors $M_{\mathbf{c},arf}$. Since some of the building parts (i.e., garage, porch) can be easily recognized from outside, and they also play an important role in the mass model, we add two feature vectors ($M_{\mathbf{c},gar}$ and $M_{\mathbf{c},por}$), which consist of position, size, and exposure ratio of the garage and porch, respectively.

—Facade attributes ($\mathbf{F_c}$). Each side of the building is described by a set of facade attributes, i.e., window-to-wall ratio ($F_{\mathbf{c},wwr}$), alignment of facade elements ($F_{\mathbf{c},align}$), and symmetry of facade elements ($F_{\mathbf{c},sym}$). They describe the number and size of each facade element type and their layouts on each building side.

—Element attributes ($\mathbf{E_d}$). We observed that the presence of specific types of building elements makes the building distinctive. For example buildings in Victorian style widely use bay windows and usually have a tower structure in the corner. We use three attributes to describe the roof style (E_{roof}), window style (E_{wnd}), and special building elements (E_{spe}).

Then the building exterior can be parameterized by a vector of the above attributes $[\mathbf{M_c}, \mathbf{F_c}, \mathbf{E_d}]$. The definition of each attribute is described in Appendix A. In total, we have seven mass model attributes, three facade attributes, and three element attributes. This results in a 49-dimensional representation.

5. PROBABILISTIC MODELING

We design a probabilistic graphical model with a hierarchy of hidden variables to represent the joint probability distribution of building attributes. In the following, we describe the motivation of our design choices in Sec. 5.1, the hierarchical graphical model in Sec. 5.2, the learning algorithm in Sec. 5.3, and the database used for learning in Sec. 5.4.

5.1 Motivation

A very popular model for encoding the relationship between variables is a directed graphical model without hidden variables and without a given hierarchy. Such a model directly encodes conditional dependencies between continuous variables $\{C_i\}$ and discrete variables $\{D_j\}$ (see Figure 4(a)). For example, such a simple directed graphical model was used by Merrell et al. [2010] to encode the topology of floor plans. However, it cannot learn different building styles. As a result, the complexity of the model will be higher than necessary requiring more training data. Therefore, it seems better to introduce a hidden variable, S, that will reduce the learned directed edges (see Figure 4(b)). In our experiments with this model, we noted difficulties in learning a style because S is dependent on both continuous and discrete variables. Specifically, defining a distance metric that considers discrete as well as continuous variables is difficult in the learning step. Based on this observation, we decided to include additional hidden variables, one for each continuous variable. Then S only depends on discrete variables, and it could easily be expressed by a conditional probability table (see Figure 4(c)). To endow the model with more semantics, we separate mass model and facade variables to arrive at the final model (see Figure 4(d)). The advantages of this model are that it is easy to learn and does not require a large amount of training data. Additionally, it is easily extendable and all the derived equations can be reused when adding additional attributes. We also considered additional refinements, by grouping multiple attributes under the same hidden variable. However, this requires extensive prior knowledge and makes it harder to extend the model. By contrast, having label information makes it easy to group variables under the same hidden variable, e.g., for encoding airplanes with labeled parts [Kalogerakis et al. 2012]. We therefore decided against such refinements. The main differences between our graphical model and the model in [Kalogerakis et al. 2012] are as follows. First, the variables in the graphical model of Kalogerakis et al. are directly related to the labeled components of a 3D model. Due to the aforementioned problems of labeling building exteriors, we need to resort to a different solution and work with building attributes as an intermediate representation. Second, we separate the continuous and discrete variables and place them on two different levels rather than placing the continuous and discrete variables on the same level.

5.2 Hierarchical Graphical Model

Random variables. Our model is a hierarchical mixture of distributions over attributes of buildings with a set of hidden variables, i.e., S, $\{M_{\mathbf{d},i}\}$, and $\{F_{\mathbf{d},j}\}$. At the root of the model, S can be interpreted as the overall style of a building. For each mass model and facade attribute, the corresponding hidden variables, $\{M_{\mathbf{d},i}\}$ and $\{F_{\mathbf{d},j}\}$, represent the styles of each attribute, respectively. The hidden variables are learned from training data as described in Sec. 5.3. Observed random variables describe attributes extracted from the data. These variables are mass model attributes $\{M_{\mathbf{c},i}\}$, facade attributes $\{F_{\mathbf{c},j}\}$, and element attributes $\{E_{\mathbf{d},k}\}$. All of them are

from the attribute-based model of the building exterior described in Sec. 4.2.

Model structure. We organize the random variables hierarchically as shown in Figure 4(d). There are three levels in our model. Random variables on the bottom level are continuous. They correspond to the mass model attributes and facade attributes introduced in the previous section. Random variables on the middle level are discrete. $\{E_{\mathbf{d},k}\}$ represents the different element styles and is expressed by binary vectors to encode the presence of specific element styles, e.g., hip roof, bay window, etc. $\{M_{\mathbf{d},i}\}$ and $\{F_{\mathbf{d},j}\}$ are hidden variables whose values represent clusters of similar buildings in the corresponding building attribute. The hidden variable, S, on the top of the model represents clusters of similar buildings in terms of the mass model, facade, and elements. To encode the conditional dependencies between the observed random variables, the model also includes lateral edges, which are not shown in Figure 4(d). For example, variables between $\{M_{\mathbf{d},i}\}$ can be connected by edges to represent strong relationships between different attributes. The lateral edges are also learned from the training data.

Probability distribution. We use different methods to encode the probability distributions of discrete and continuous random variables. The discrete random variables, $\mathbf{D} = \{S, \mathbf{M_d}, \mathbf{F_d}, \mathbf{E_d}\}$, are expressed by conditional probability tables (CPTs) or conditional probability distributions (CPDs), while the continuous random variables, $\mathbf{C} = \{\mathbf{M_c}, \mathbf{F_c}\}$, are always expressed by CPDs. As CPDs we use distributions based on sigmoid functions for discrete variables and conditional multi-variate Gaussian distributions for continuous variables.

Based on the structure of the model, the probability distribution $P(\mathbf{B})$ over all random variables $\mathbf{B} = \{S, \mathbf{M_d}, \mathbf{M_c}, \mathbf{F_d}, \mathbf{F_c}, \mathbf{E_d}\}$ can be factored as a product of conditional probability distributions:

$$P(\mathbf{B}) = P(S)\prod_i P(M_{\mathbf{d},i}|S)P(M_{\mathbf{c},i}|M_{\mathbf{d},i}, \pi(M_{\mathbf{c},i}))\cdot \prod_j P(F_{\mathbf{d},j}|S)P(F_{\mathbf{c},j}|F_{\mathbf{d},j}, \pi(F_{\mathbf{c},j})) \cdot \prod_k P(E_{\mathbf{d},k}|S), \quad (1)$$

where $\pi(M_{\mathbf{c},i})$ and $\pi(F_{\mathbf{c},j})$ are the learned parents of the corresponding random variables.

5.3 Learning

We derive a learning framework based on the description by Koller and Friedman [2009]. Given a set of N buildings, $\mathbf{O} = \{O_1, O_2, \ldots, O_N\}$, we learn the structure, G (i.e., the cardinality of the values of the hidden variables and the lateral edges) and the parameters, Θ (i.e., parameters of CPTs and CPDs). Each training sample is encoded as a feature vector $O_i = [\mathbf{M_c^i}, \mathbf{F_c^i}, \mathbf{E_d^i}]$.

Structure learning. We learn the structure, G, of the model using maximum a posteriori (MAP) estimation. To make the computation tractable, we use a scoring function to estimate $Score(G|\mathbf{O})$. Following Bayes' rule, the straightforward choice of a scoring function is the BIC score [Heckerman 1998]. However, we try to learn a model with hidden variables using a relatively small dataset. In this case, the BIC score is not accurate enough. Therefore, we use the Cheeseman-Stutz score [Cheeseman and Stutz 1996] which provides a good tradeoff between accuracy and computational speed. Given a structure, G, and the training data, $\mathbf{O}$, the score of the structure is defined as follows:

$$Score(G|\mathbf{O}) = \log P(G) + \log P(\mathbf{O}^*|G) + \log P(\mathbf{O}|G, \Theta_G) - \log P(\mathbf{O}^*|G, \Theta_G), \quad (2)$$

Fig. 5: An example building from our database. Left: the attributes of the building mass model are automatically extracted from the contour of the building footprint. The garage and the porches are labeled manually. Middle: the facade region, windows and their styles, and doors are annotated. The facade attributes are computed based on the annotations. Right: assigned element attributes according to the corresponding building.

where $\mathbf{O}^*$ is a fictitious complete dataset that consists of the training data $\mathbf{O}$, estimated values for the hidden variables and estimated statistics for all variables, and Θ_G are the parameters estimated for a given structure, G. To facilitate learning, we also assume a uniform prior $P(G)$ over possible structures.

Structure learning algorithm. Since our model contains hidden variables, it is still difficult to maximize the Cheeseman-Stutz score. Due to the complexity of structure learning, most existing algorithms are heuristic in nature. We therefore apply an iterative search method to explore possible graph structures with different value spaces for the hidden variables. This method is adapted from previous works [Koller and Friedman 2009; Kalogerakis et al. 2012]. The search procedure is initialized as follows: we start by setting the cardinality of all the value spaces of all hidden variables to 2. Then, we gradually increase the cardinality of the value space of the first hidden variable on the middle level, evaluating the score at each iteration. If the score increases, we keep the change and try the next higher cardinality. Otherwise, we keep the previous cardinality and move to the next hidden variable on the middle level. After iterating over all variables on the middle level, we increase the cardinality of the value space of S, reinitialize the cardinality of all hidden variables on the middle level to 2, and then repeat the procedure. We do this by trying all values in $[2, 25]$ for the cardinality of the value space of S. In our implementation we also cap the cardinality of other value spaces to 25. At the end, we keep the configuration with the highest score. For each configuration of value spaces that is explored in the search algorithm, we then need to search over the possible set of lateral edges between observed random variables. We employ a greedy search method that proposes new candidate structures by adding edges, testing the topological validity, and then scoring the structure candidate. We retain the graph structure with the highest score.

Parameter learning. For a given structure G, we estimate the parameters using MAP. We apply an expectation-maximization (EM) framework to optimize the parameters of our model. The EM algorithm iterates until all parameters change by less than 10^{-6}. In order to achieve a better generalization, we follow common conventions and assume Dirichlet priors and normal-Wishart priors for the discrete random variables and the continuous random variables, respectively.

5.4 Building Dataset

We generated a building database of 200 buildings consisting of building footprints and photographs. Thumbnails of all buildings in our dataset are shown in the additional materials. Figure 5 shows

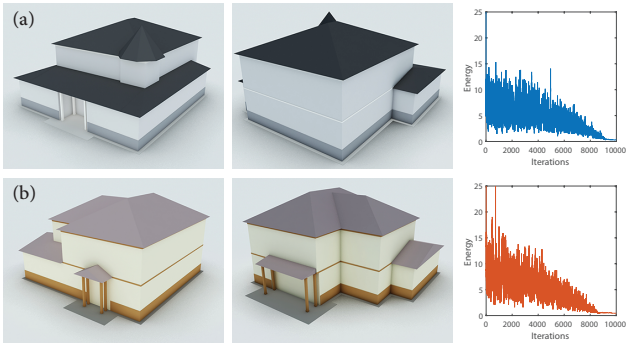

Fig. 6: Two mass models generated by the given sets of attributes (shown in the additional materials). The evolution of the energy of each mass model during the optimization process is shown on the right.

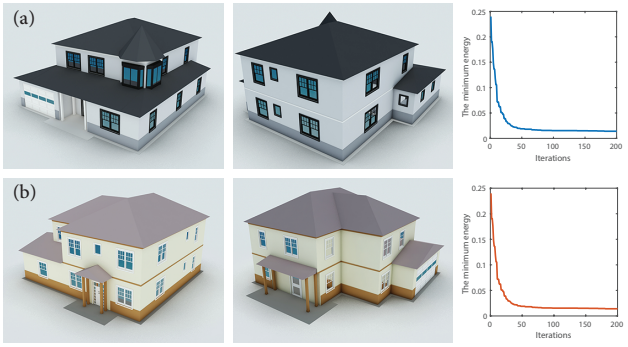

Fig. 7: Given the mass models in Figure 6 and the target attributes (shown in the additional materials), we generate facade layouts using our algorithm by reusing the elements in the database. The minimal energy of each iteration is shown on the right.

an example of how we obtain data from a building. We developed a tool to analyze the attributes of the building mass model from images and to process the facade images semi-automatically. The user needs to annotate only the garages and the porches on the footprints and the observed facade elements on the facade images. Then, all the attributes are computed automatically. Potentially, automatic image parsing methods could be used, but this is currently not implemented. A larger database could be built using aerial images and corresponding street view images from an Internet-based mapping site such as Google Maps or Bing Maps. The building database is used to learn the probabilistic model described in this section.

6. BUILDING GENERATION

Given an instance of the building parametric model described in Sec. 4.2, i.e., a vector of building attributes, $[\mathbf{M_c}, \mathbf{F_c}, \mathbf{E_d}]$, we would like to compute a corresponding parametric building representation (Sec. 4.1) that takes the attributes as constraints. In this section, we describe the following components: mass model and roof generation (Sec. 6.1) and facade generation (Sec. 6.2). Mass model generation is a fairly easy problem and we observed good results with a straightforward simulated annealing algorithm. Roof generation is a simple geometric construction. Facade generation is more difficult. Therefore, we propose a two-step approach. We use a stochastic algorithm to propose candidate layouts with varying topologies and quadratic programming to optimize size and spacing parameters. The colors are assigned based on pre-designed templates extracted from the training data.

6.1 Mass Model and Roof Generation

Mass model energy. We define the mass model energy, E, for M by summing up the normalized squared error of each attribute:

$$E(M) = \sum_i \frac{\|M_{\mathbf{c},i} - M^*_{\mathbf{c},i}\|^2}{\|M^*_{\mathbf{c},i}\|^2} + E_{topo}(M) + E_{thick}(M), \quad (3)$$

where $\{M^*_{\mathbf{c},i}\}$ is the desired attribute set. To generate more realistic mass models, we add two additional constraints, i.e., a topology constraint (E_{topo}) and a thickness constraint (E_{thick}). We require that all boxes in the mass model are connected, and we penalize the mass model with more than one connected part by returning a large constant (10^6), otherwise 0. We also would like to avoid mass models that are too narrow. We find all pairs of parallel edges and ensure the distance between each pair of edges, t, is constrained as $t \geq t_{min}$. The energy, E_{thick}, is defined as $(\frac{t-t_{min}}{t_{min}})^2$ if $t < t_{min}$ and 0 otherwise, where t_{min} is the minimum width of the box. t_{min} could be learned from the database; in practice, we simply set it to 0.5 m.

Simulated annealing. We employ simulated annealing (SA) to generate the mass model, M, according to the mass model attributes, $\mathbf{M_c}$. We start with an empty layout, i.e., $M = \emptyset$. At each iteration, a new mass model, M', is proposed by applying one of the following operations randomly: 1) adding a box, b, i.e., $M' = M \cup b$; 2) removing an existing box, b, at random, i.e., $M' = M \setminus b$; 3) moving an existing box, b, by randomly perturbing $(b.x, b.y)$; 4) resizing an existing box, b, by changing its size randomly $(b.w, b.d, b.h)$. To get more plausible alignments, edges of any pair of boxes that are closer than 10 cm are snapped. Then, we compute the energy for the new mass model as E'. If $E' < E$, we always accept M'; otherwise, we accept M' with probability of $\exp(-(E' - E)/t)$, where t is the temperature. We gradually reduce the temperature, t, in each iteration. The algorithm terminates either when E is smaller than a threshold (10^{-3}), or when the maximum number of iterations (i.e., 10^4) has been reached. Figure 6 shows that given a set of mass model attributes, SA can generate a desired mass model. The target attributes and results for each mass model are shown in the additional materials.

Roof generation. The main roof style of the mass model is based on the roof style in element attributes, $\mathbf{E_d}$. In our current implementation, we can generate five types of roofs, i.e., flat, gable, hip, shed and cone roof. Gable roofs and hip roofs are generated by applying the straight skeleton method [Aichholzer et al. 1996] over the top polygons of the mass model.

6.2 Facade Generation

Given a mass model, M, the next step is to place elements on the facades of M. Our facade generation problem is more challenging than in previous works, since our goal is to obtain the facades for the whole building, which requires the facades of each building side to be consistent. Fan et al. [2014] use grids to encode a facade structure, but the proposed approach only works on a single facade. We propose a facade generation method that can generate facades for the whole building from data. Moreover our method considers the consistency between different facade pieces and the reusability of the elements (i.e., windows and doors).

Facade layout energy. We formulate the facade layout problem as an optimization problem to minimize the following energy of the

facades on all building sides:

$$E(F) = \sum_s \sum_j \left(\frac{F_{\mathbf{c},j,s} - F^*_{\mathbf{c},j,s}}{F^*_{\mathbf{c},j,s}} \right)^2, \quad (4)$$

where s is the index of the building side, and $\{F^*_{\mathbf{c},j,s}\}$ denote the desired facade attributes for the s-th building side.

Genetic algorithm. There are two main challenges in solving the above problem. One challenge is how to obtain the combination of facade elements on each facade piece; the other is how to obtain the new size for each facade element. We propose a stochastic optimization method combined with a continuous facade layout optimization to solve these two problems jointly. Before that, in our building database, we label elements according to their main style (e.g., sash window, sliding window, etc.) and then generate five sub-styles of the elements for each style by k-means clustering according to their sizes. We use the style attributes of the building to find the building with the most similar style attributes in the database and reuse its facade elements. Given a set of facade element candidates, we employ a genetic algorithm to solve the problem. The elements layout of each facade piece can be represented by a set of genes. Then the facades of the building can be represented as a genome, that contains four sections corresponding to four building sides, and each section consists of a set of genes describing its corresponding facade piece. We set the population size to 100. A crossover is computed by exchanging the layouts of the same facade piece from two parents. A mutation is computed by sampling the layout for a randomly selected facade piece. By using this method, we can explore the combination of facade elements efficiently. Once a combination of facade elements is obtained, we apply the facade layout optimization to adapt them to each facade piece. Then the obtained facade layouts are evaluated by Equation 4.

Facade layout optimization. Since the sizes of the assets need to be adapted to each specific building, we employ a quadratic programming approach similar to Bao et al.'s work [2013] by computing

$$\arg\min_{w_i} \sum_i (w_i - w_i^*)^2, \quad (5)$$

where w_i^* is the width suggested by the genetic algorithm. The optimization tries to find a new size for each element sub-style and to make it as close as possible to its original size. The constraints for this problem are: 1) the region-size constraint ensures that the width of each element is in the range of prescribed widths $\underline{w}_i \leq w_i \leq \overline{w}_i$, where $\underline{w}_i$ and $\overline{w}_i$ are the minimum and maximum of the allowed width for element e_i. In practice, we set them to be 0.5 times and 1.5 times the average size of the corresponding element respectively. 2) The total-size constraint enforces that the elements in each facade piece equal to the current region's size, w_{total}. It takes form $\sum_i w_i = w_{total}$. Figure 7 shows the generated facade layouts according to the given mass models in Figure 6 and the facade attributes.

7. RESULTS

We implemented the proposed framework in C++ using the CGAL library for geometric computations and the GAlib library for the genetic algorithm. We report running times for a computer with two 2.53 GHz Intel core i5 processors and 8 GB main memory.

The parametric building model. First, we evaluate the parametric building model and the building generation. We reconstruct a ground-truth building based on a photograph using our parametric building model. Then, we encode this parametric building model using the attribute-based building representation and use the optimization algorithms from Sec. 6 to regenerate the parametric building model. In Figure 8, we show that the regenerated building is similar to the ground-truth building. Second, we evaluate the effect of each attribute by performing leave-one-out tests. In the top two rows of Figure 9, we show the effect of each mass model attribute. We can see that the building size is unreasonable if $M_{\mathbf{c},bbox}$ is excluded. Without $M_{\mathbf{c},bcr}$, the coverage of the building is much larger than the ground-truth building. There is an unexpected courtyard in the regenerated building without $M_{\mathbf{c},bdry}$. Excluding $M_{\mathbf{c},far}$ or $M_{\mathbf{c},arf}$ leads to buildings with quite different second floors to that of the ground-truth building. $M_{\mathbf{c},gar}$ and $M_{\mathbf{c},por}$ contribute to the position and size of the garage and porches, respectively. The effect of each facade attribute is shown in the bottom row of Figure 9. The facades are synthesized on the mass model shown in (b). We can see that the number of windows and their sizes are determined by $F_{\mathbf{c},wwr}$, the types and arrangement of the windows are determined by both $F_{\mathbf{c},sym}$ and $F_{\mathbf{c},align}$, e.g., the windows' types and positions are quite different than (j). In summary, we find that each attribute contributes to the regenerated result and that leaving out any attribute reduces the overall quality.

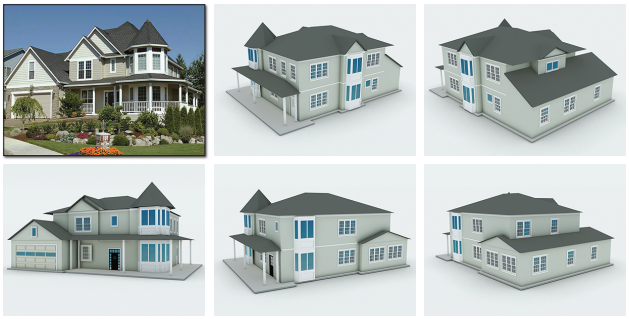

Fig. 8: We regenerate the building according to the attributes from the real building model. Top: photograph of the real building and two views of the reconstructed building model. Bottom: three views of the regenerated building. The attributes of each building are shown in the additional materials.

Structure of the model. Our model uses hidden variables to compactly parameterize the structural variability of building models. The cardinality of the value spaces of these variables are learned from data and different values represent different underlying styles of buildings. In Figure 10, we show the seven styles learned from our dataset for the hidden variable, S. The buildings are sampled by fixing S in the learned model. We can observe that the seven styles impose a reasonable partition on the space of all buildings. For example, styles 1-3 are simpler residential buildings that are predominantly one floor high. We can also see that the discrete hidden variables are very useful to encode the existence of a porch or a garage. For example, if multiple models within the same style are found without a garage (encoded by setting all garage attributes to 0), the discrete hidden variable will form a cluster of these models. During synthesis, if the garage attributes are 0, we do not generate a garage. Lateral edges encode strong relationships between different attributes. We learn nine lateral edges in our graphical model (see Fig 11 left). For example, one of the edges connects the size of the bounding box with the garage attributes. This implies that a large building usually has a large garage as well.

Evaluation of synthesized results. We sample high-probability, attribute-based building models. To avoid similar buildings, we re-

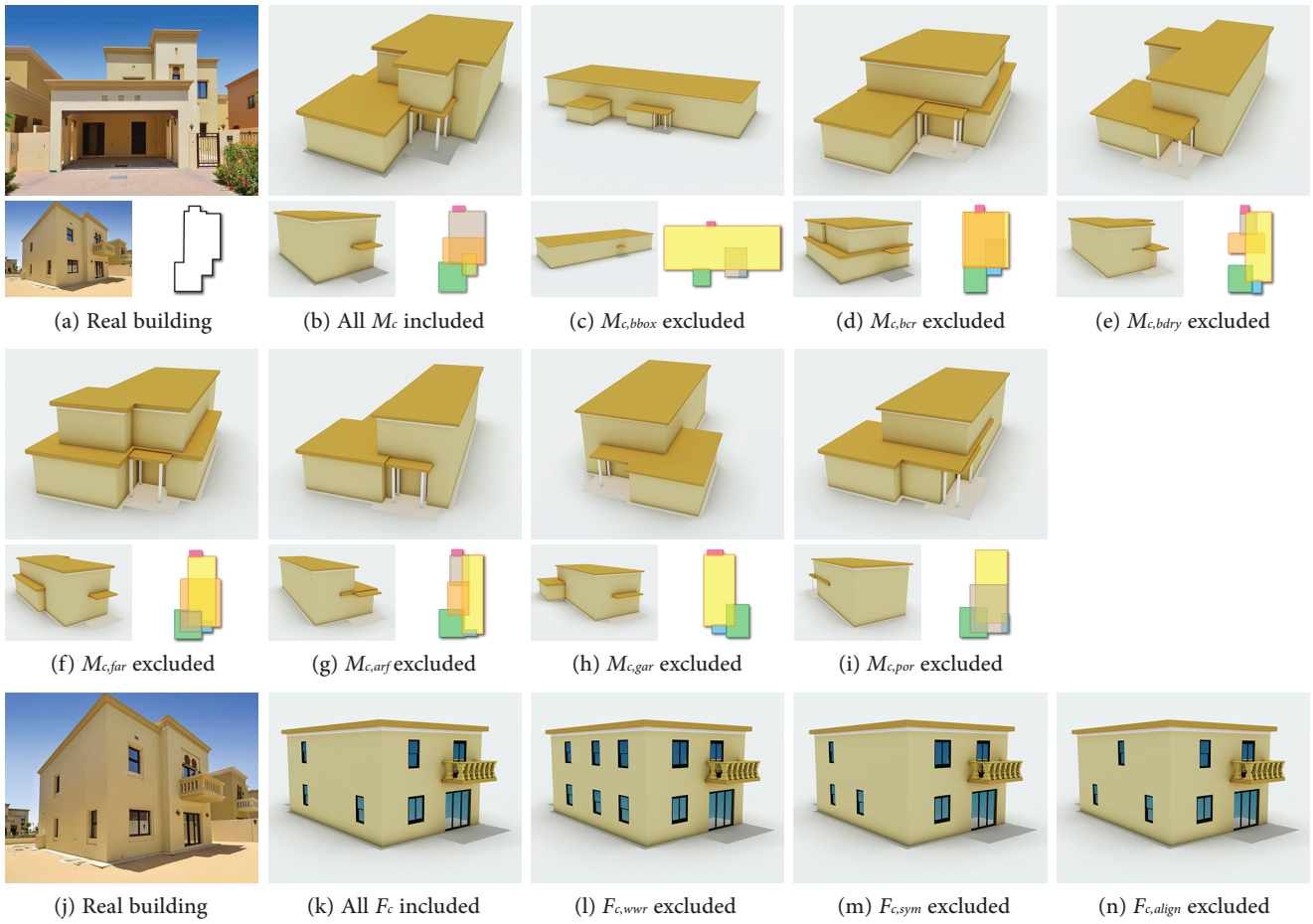

Fig. 9: The effect of each attribute. The top two rows show the effect of each mass model attribute. We extract the attributes from a real building (a). (b) A mass model generated with all attributes selected from the five-highest probability results. (c-i) Ablation of individual attributes and the effect on the result. For each mass model, two views of the mass model and its box representation from a top view are shown. The third row shows the effect of each facade attribute. All facades are generated on the same mass model shown in (b). (k) A facade model generated with all attributes selected from the five highest-probability results. The attributes of each building are shown in the additional materials.

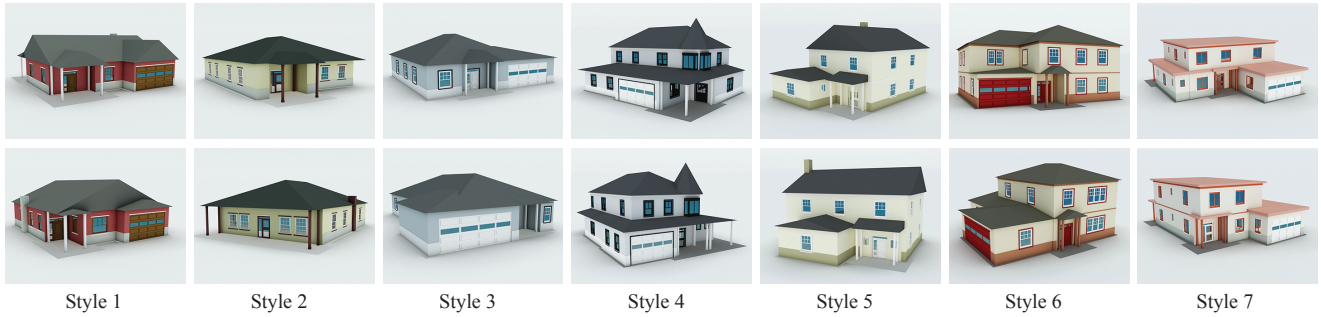

Fig. 10: Seven building styles are encoded in the hidden variable, S, of our model. We show two high-probability buildings of each style.

ject samples that have similar attributes to a model in the input dataset or to a previous sample. To evaluate the compatibility of attributes from different source buildings, we sample a large number of building models and collect the top 1000 models with high-probability. By computing the most similar building model from the database per attribute, we can observe that on average, each sample consists of the attributes from 6 different buildings in the

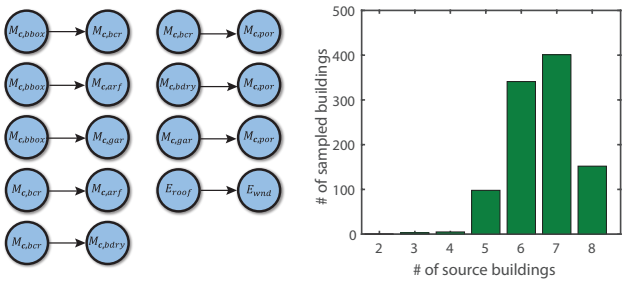

Fig. 11: Left: nine learned lateral edges of our graphical model. Right: histogram of the number of different buildings in the training dataset contributing to the attributes to sampled buildings.

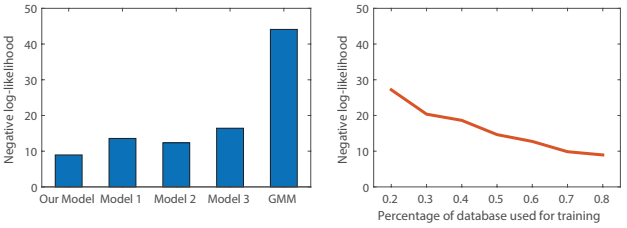

Fig. 12: Evaluations of generalization performance. Left: negative log-likelihood of our model compared with other versions of the model. Lower negative log-likelihood indicates better generalization performance. Model 1 is similar to our model but without learned edges between variables. Model 2 does not use hidden variables on the second level (see Figure 4(b)). Model 3 uses a directed graphical model without hidden variables (see Figure 4(a)). The performance of a Gaussian mixture model is visualized as the right bar. Our model achieves the best generalization performance across the test set. Right: The impact of training set size on generalization performance. Performance increases as the training set becomes larger.

training dataset (see Fig 11 right), which means that our graphical model can sample buildings with high compatibility of attributes.

Comparison with other graphical models. Similar to a classification algorithm that can be evaluated by predicting the label of instances in a test dataset, a generative model can be evaluated by computing the probability of the instances in a test dataset. We expect that a good generative model assigns high probability to the instances in the test dataset, which means that the model can synthesize novel and reasonable instances. We apply the holdout validation method to evaluate the generalization performance of our model. Firstly, we randomly split out dataset into a training set (80%) and a test set (20%). Then, we train our model using only the training dataset and compute the likelihood for all test data. Higher likelihood on the test dataset corresponds to better generalization performance. We repeat the procedure five times. In Fig 12 left, we show the generalization performance of our model and compare it with alternative graphical models discussed in Sec. 5.1. Each bar in the figure denotes the negative logarithm of the geometric mean of the likelihoods, and a lower bar corresponds to better generalization performance. We can see that our model has the best performance.

Comparison with the baseline method. The purpose of this test is to investigate what happens if no graphical model is used to encode the joint probability distribution of the building attributes. The result of this test will be useful to understand the benefit of a graphical model in general in the context of our application. For this purpose, we set up a baseline method for comparison using a

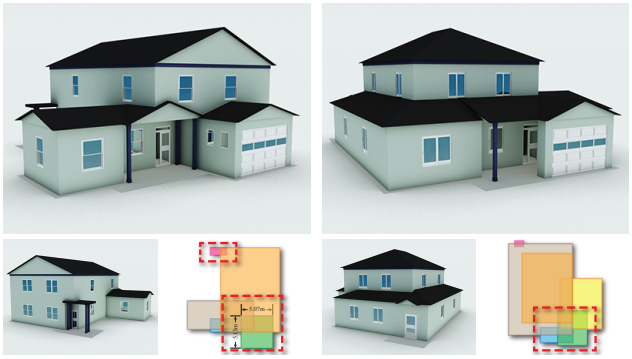

Fig. 13: Two buildings are reconstructed by the mean vectors of a Gaussian mixture model. Two views of the building and the parametric representation are shown. The problems of the building are highlighted in the box representation by red rectangles. The front porch is in blue, the back porch is in red, and the garage is in green.

Gaussian mixture model (GMM), which does not encode the relationship between building attributes. Each building in the dataset is represented as a high-dimensional feature vector by concatenating the attributes. We build a GMM for the feature vectors to encode the building distribution and estimate the number of components in the mixture model using the MDL criterion [Rissanen 1983]. Finally, we obtain a mixture model with two components and extract the mean of each Gaussian as building attributes. The reconstructed buildings of the extracted attributes are shown in Figure 13. We can find that GMM cannot encode and learn the building distribution well. For example, in the left building, the position of the back porch is unreasonable, and the size of the garage is neither reasonable for a one-car garage nor for a two-car garage. In the right building, the front porch and garage intersect. The causes of the problems are the high-dimensionality of the feature vector and the comparatively small dataset. More importantly, GMM cannot encode the relationship between different attributes well. To compare our model with the baseline method, we sample the building attributes with given bounding box and garage attributes. The reconstructed buildings are shown in Fig 14. We can see that the front porch and garage intersect in the building sampled from the GMM. We also evaluate the generalization performance of the GMM, which is much worse than our model (see Figure 12 left).

Training set size. In Figure 12 right, we evaluate the performance of our model using an increasingly larger training set. Using the same test protocol as in the previous test, we can observe that our model improves with more training data. However, based on the slope of the curve, we can also speculate that better performance could be achieved with a larger database.

Applications. We demonstrate two major applications of our work. The first application is building synthesis. We sample the building attributes from the trained probabilistic graphical model and generate the building model as discussed in Sec. 6. Several synthesis results are shown in Figs. 10 and 15. The second application is building completion. Starting from a single image, a user can specify parts of a building mass model and mark facade elements (i.e., windows and doors) on the observed part of a building facade in a rectified image. Our framework will extract the observed attributes from the annotations and sample the remaining building attributes with high-probability automatically. Then, taking both user's annotations (e.g., window sizes and locations) and the building attributes as constraints, our framework can re-

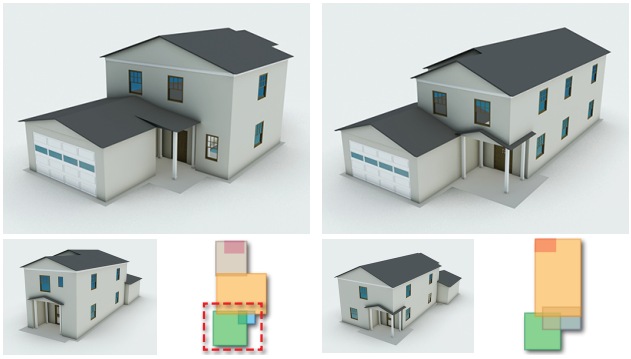

Fig. 14: The left and the right buildings are reconstructed by the attributes sampled from a GMM and our model, respectively. $M_{\mathbf{c},bbox}$ and $M_{\mathbf{c},gar}$ are given. Two views of the building and the box representation are shown. The front porch and garage intersect in the building sampled from the GMM.

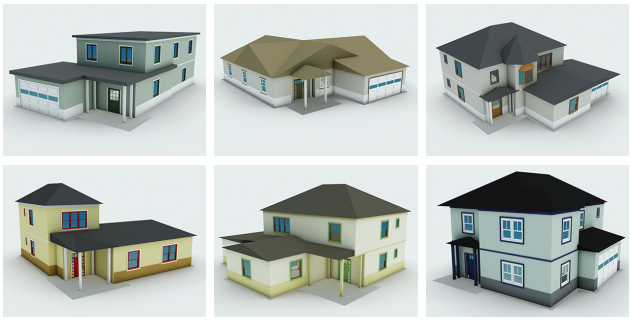

Fig. 15: Six buildings synthesized using our algorithm.

construct the whole building. The whole process is shown in Figure 1 and the accompanying video. Figure 17 shows three buildings that are completed from images. Our completion framework can also take a partial 3D building as input. Since we can sample multiple attribute-based representations and since the mapping from attributes to the parametric representation is not unique, we can generate many possible completion results. For evaluation purposes, we generate five possible reconstructions and select the best fitting candidate among them. Figure 16 shows that given a partial building, our work can complete the whole building which is similar to the ground truth. More results are shown in the additional materials. Another example application is a suggestion system for residential building design. A user can specify partial attributes of a building and ask the system for a complete building design. Then the user can edit the completed building design by adding, deleting, and moving building boxes. We show such an editing sequence in Figure 19. The main bottleneck of this application is the performance of the building generation algorithm, which we leave to future work.

Performance. In our implementation, learning takes about 40 minutes for 200 buildings. Given a set of attributes, the algorithm takes about 12 minutes to obtain a set of building mass models and 1 minutes to get facade layouts on a mass model.

Discussions and limitations. In the probabilistic model, we deal with two special building structures, i.e., garages and porches. The main reason is that these two structures can be recognized from the outside of the building without ambiguity. In contrast, it is not

Fig. 16: Building completion compared to ground truth. Top row: photograph of the building, the right side and the back side of the incomplete building. Middle row: the ground truth shown from the front, back, and right side respectively. Bottom row: one possible completed building selected from the five highest probability reconstructions.

Fig. 18: Given an incomplete building model, our algorithm can generate multiple completions. Top row: photograph of the building, and two views of the incomplete building. Middle and bottom rows show three possible completions. For each completion, the front and back of the building are shown.

easily possible to recognize other rooms, such as a living room or a kitchen from the outside. If a building is without a garage or a porch, we set the corresponding attributes to zero. Note that our model can be easily extended by adding other building structures, such as patios and decks, to describe more details of a building. In our work, the mass model generation and the facade generation are formulated as non-convex optimization problems that have many local minima in the general case. A disadvantage of our solution is that we cannot guarantee finding a global minimum for either of these two problems. In the mass model generation, our graphical model samples a set of high-probability attributes. In gen-

Fig. 17: Three buildings are completed using our method. Each completed building is shown from three views. The observed parts are highlighted in yellow.

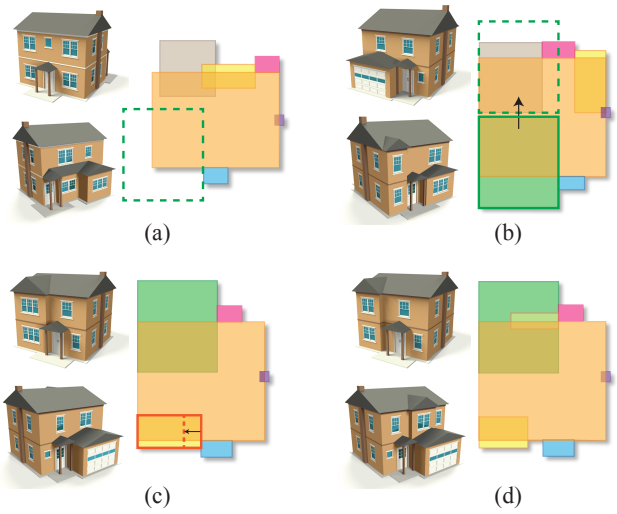

Fig. 19: An application for building design. For each stage, two views of the current 3D model are shown, and the new modifications are illustrated on the box representation. (a) The initial model. In this edit, the user adds a garage in the left front corner. The garage position is shown as a green dashed box. The system suggests a new building (b), and the user moves the garage to the back of the building yielding result (c). Then, the user changes the size of one building box (shown by a red dashed line). The final building model is shown in (d).

eral, the building attributes might be conflicting, but since we start with high-probability attributes, the conflicts are typically minor. Therefore, simulated annealing can find good local minima, i.e., a configuration where the parametric description and the attribute description of the building are in agreement. In our implementation, we restart simulated annealing three times and take the best result. However, like in most other applications of simulated annealing, no theoretical guarantees for convergence can be given, not even for finding a local minimum. In the facade generation, we opted for a genetic algorithm instead of simulated annealing, because the exploration of the search space is more difficult. In this context, it is important to have a proposed mechanism for large changes. Here, we found the crossover operation of genetic algorithms to be more suitable than the restart operation in simulated annealing. While our genetic algorithm can successfully find low energy states in the search space, we cannot guarantee that a local or global minimum has been found. While our building generation has high-quality output, it takes several minutes to obtain a good result. To generate very large environments, it might be essential to optimize this step first. Our model does not consider appearance (e.g., color and texture) of the building. We believe that appearance can be integrated into our model, but this requires a larger database for training, thus we leave appearance to future work. Also, our model does not consider the context of a building, such as the parcel shape, parcel slope, vegetation, or the relative location of the street. Although one of our applications is building completion, our model does not focus on reconstruction of an exact building from a single image. We consider this problem complementary to our work.

8. CONCLUSIONS

In this paper, we proposed a probabilistic model for building exteriors of residential buildings. Starting from a variable-sized parametric building description, we identify a fixed dimensional list of attributes to describe a building. The joint probability distribution of these attributes is then encoded in a hierarchical graphical model with hidden variables. Finally, an optimization algorithm can convert a set of building attributes to a three-dimensional geometric model. This framework is useful for learning building designs from training data and automatically synthesizing new buildings. Additionally, it can be used for completing buildings that are partially reconstructed from photographs. The main motivation for designing our model was to consider only information that can be observed from the outside. In this way, we can make use of widely available data from Internet-based mapping sites such as Google maps and Bing maps to build a training dataset. Our framework relies on a conversion from an attribute-based representation to a parametric representation. It is a promising research direction to find a model that would support both learning and building generation without the conversion step. In future work, we would also like to extend our probabilistic model to include information of garden layouts. Further, we would like to extend the model to include building interiors that can conform to a given building mass model.

ACKNOWLEDGMENTS

We would like to acknowledge the help of Yoshihiro Kobayashi and Christopher Grasso for rendering most of the images in this paper, the help of Tina Smith for video narration, and the help of Virginia Unkefer for proofreading.

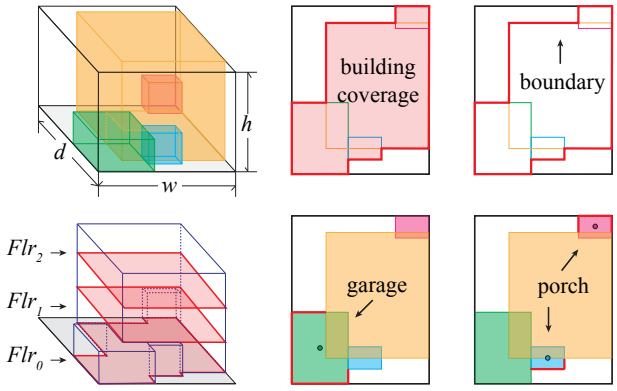

Fig. 20: Mass model attributes used in our framework.

APPENDIX

A. BUILDING PARAMETRIC MODEL

A.1 Mass Model Attributes $\mathbf{M_c}$

We use seven attributes to describe a building mass model (See Figure 20):

Size of bounding box $M_{\mathbf{c},bbox}$**.** The size of the bounding box of a mass model: $M_{\mathbf{c},bbox} = (w, d, h)$, where w, d and h are the width, depth and height of the bounding box of the building mass model.

Building coverage ratio $M_{\mathbf{c},bcr}$**.** We borrow the concept of building coverage ratio from architecture to describe the coverage ratio of the ground floor. Since we do not know the real parcel of the building, we replace the building parcel by the bounding box of the ground floor (Flr_0). Then, $M_{\mathbf{c},bcr} = \frac{Area(Flr_0)}{Area(BBox(Flr_0))}$.

Boundary complexity $M_{\mathbf{c},bdry}$**.** We use an attribute to describe the boundary complexity of the building. It is defined as $M_{\mathbf{c},bdry} = \frac{Perimeter(Flr_0)}{\sqrt{Area(Flr_0)}}$.

Floor area ratio $M_{\mathbf{c},far}$**.** We also borrow the concept of floor area ratio from architecture to describe the availability of space. Similar to $M_{\mathbf{c},bcr}$, we use the bounding box of the ground floor instead of the building parcel. Then, $M_{\mathbf{c},far} = \frac{\sum_i Area(Flr_i)}{Area(BBox(Flr_0))}$.

Area ratio of floors $M_{\mathbf{c},arf}$**.** We use this attribute to describe the area of each floor. In our current problem, each residential building has less than three floors, thus we use a 3-dimensional vector to encode it, $M_{\mathbf{c},arf} = [ar_0, ar_1, ar_2]$, where $ar_i = \frac{Area(Flr_i)}{\sum_j Area(Flr_j)}$, $i = 0, 1, 2$.

Garage attributes $M_{\mathbf{c},gar}$**.** We define a 6-dimensional vector to encode a description of the garage. We concatenate three features, i.e., $M_{\mathbf{c},gar} = [GarPos, GarSize, GarBdry]$, where $GarPos$ denotes the garage's 2D position in the bounding box of the ground floor, $GarSize$ denotes the 3D size of the garage, $GarBdry = \frac{Length(Visible(garage))}{Perimeter(garage)}$. $Visible(garage)$ computes the visible part of the garage from the outside of the building (shown in red in Figure 20). If the building does not have a garage, we set all the elements of this attribute to 0.

Porch attributes $M_{\mathbf{c},por}$**.** We describe the front and back porches using a 12-dimensional feature vector, $M_{\mathbf{c},por}$. It is concatenated by two feature vectors from the front porch and the back porch. The feature vector is defined in the same manner as $M_{\mathbf{c},gar}$.

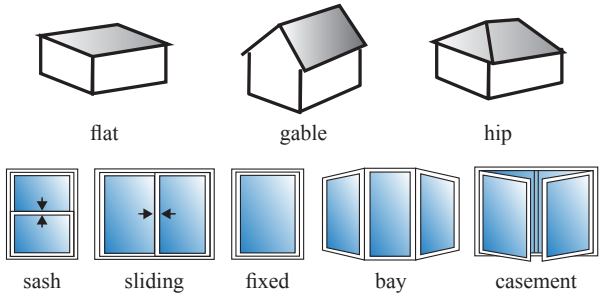

Fig. 21: An illustration of roof styles and window styles used as element attributes.

A.2 Facade Attributes $\mathbf{F_c}$

As discussed in Sec. 4.2, each building side consists of a set of facade pieces $\{f_j\}$ and each of the facade pieces contains a set of facade elements $\{e_k\}$. We use the following attributes:

Window-to-wall ratio $F_{\mathbf{c},wwr}$**.** We use the window-to-wall ratio to describe how much of the facade is covered by facade elements. This attribute helps us to decide the size and number of facade elements. It is defined as $F_{\mathbf{c},wwr} = \frac{\sum_k Area(e_k)}{\sum_j Area(f_j)}$.

Symmetry $F_{\mathbf{c},sym}$**.** Symmetry is a very important attribute observed in facades. To compute the symmetry attribute, we sum up the area of all symmetric facade pieces and divide it by the total facade area: $F_{\mathbf{c},sym} = \frac{\sum_j Area(f_j) \cdot IsSymmetric(f_j)}{\sum_j Area(f_j)}$.

Alignment between facade pieces $F_{\mathbf{c},align}$**.** We also consider the vertical alignment of elements between facade pieces. We first find all pairs of facade pieces that can be aligned. The number of candidate pairs is N_{candi}. Then, we count the number of aligned pairs, N_{align}. Finally, $F_{\mathbf{c},align} = \frac{N_{align}}{N_{candi}}$.

A.3 Element attributes $\mathbf{E_d}$

For each element attribute, we use a binary vector to encode the presence of specific element styles. We consider three types of elements, i.e., roof, window, and special building elements. We currently do not encode the door style and assume that there is at least one door.

Roof styles E_{roof}**.** We use a 3D binary vector to encode the presence of a particular roof style. The three roof styles in our dataset are flat, gable, and hip.

Window styles E_{wnd}**.** Window styles are encoded by a 5D binary vector. In our database, we use sash window, sliding window, casement window, bay window, and fixed window.

Special building elements E_{spe}**.** We also encode the presence of two special building elements, i.e., chimney and tower, as a 2-dimensional binary vector.

REFERENCES

AICHHOLZER, O., AURENHAMMER, F., ALBERTS, D., AND GÄRTNER, B. 1996. *A novel type of skeleton for polygons.* Springer.

AVERKIOU, M., KIM, V., ZHENG, Y., AND MITRA, N. J. 2014. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum 33,* 2, 125–134.

BAO, F., SCHWARZ, M., AND WONKA, P. 2013. Procedural facade variations from a single layout. *ACM Trans. Graph. 32,* 1 (Feb.), 8:1–8:13.

BAO, F., YAN, D.-M., MITRA, N. J., AND WONKA, P. 2013. Generating and exploring good building layouts. *ACM Trans. Graph. 32,* 4 (July), 122:1–122:10.

BOKELOH, M., WAND, M., AND SEIDEL, H.-P. 2010. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29,* 4 (July), 104:1–104:10.

CAMPBELL, N. D. F. AND KAUTZ, J. 2014. Learning a manifold of fonts. *ACM Trans. Graph. 33,* 4 (July), 91:1–91:11.

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3D modeling. *ACM Trans. Graph. 30,* 4 (July), 35:1–35:10.

CHEESEMAN, P. AND STUTZ, J. 1996. Bayesian classification (autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*. 153–180.

FAN, L., MUSIALSKI, P., LIU, L., AND WONKA, P. 2014. Structure completion for facade layouts. *ACM Trans. Graph. 33,* 6 (Nov.), 210:1–210:11.

HECKERMAN, D. 1998. *A tutorial on learning with Bayesian networks*. Springer.

KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph. 31,* 4 (July), 55:1–55:11.

KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. 2013. Learning part-based templates from large collections of 3d shapes. *ACM Trans. Graph. 32,* 4 (July), 70:1–70:12.

KOLLER, D. AND FRIEDMAN, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

LIN, J., COHEN-OR, D., ZHANG, H., LIANG, C., SHARF, A., DEUSSEN, O., AND CHEN, B. 2011. Structure-preserving retargeting of irregular 3D architecture. *ACM Trans. Graph. 30,* 6 (Dec.), 183:1–183:10.

MARTINOVIC, A. AND VAN GOOL, L. 2013. Bayesian grammar learning for inverse procedural modeling. In *CVPR*. 201–208.

MERRELL, P., SCHKUFZA, E., AND KOLTUN, V. 2010. Computer-generated residential building layouts. *ACM Trans. Graph. 29,* 6 (July), 181:1–181:12.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND GOOL, L. V. 2006. Procedural modeling of buildings. *ACM Trans. Graph. 25,* 3 (July), 614–623.

MUSIALSKI, P., WONKA, P., ALIAGA, D. G., WIMMER, M., VAN GOOL, L., AND PURGATHOFER, W. 2013. A survey of urban reconstruction. *Computer Graphics Forum 32,* 6, 146–177.

RISSANEN, J. 1983. A universal prior for integers and estimation by minimum description length. *The Annals of statistics*, 416–431.

SCHWARZ, M. AND MÜLLER, P. 2015. Advanced procedural modeling of architecture. *ACM Trans. Graph. 34,* 4 (July), to appear.

SMELIK, R. M., TUTENEL, T., BIDARRA, R., AND BENES, B. 2014. A survey on procedural modelling for virtual worlds. *Computer Graphics Forum 33,* 6, 31–50.

ŠT'AVA, O., BENEŠ, B., MĚCH, R., ALIAGA, D. G., AND KRIŠTOF, P. 2010. Inverse procedural modeling by automatic generation of l-systems. *Computer Graphics Forum 29,* 2, 665–674.

TALTON, J., YANG, L., KUMAR, R., LIM, M., GOODMAN, N., AND MĚCH, R. 2012. Learning design patterns with bayesian grammar induction. In *Proc. of UIST*. 63–74.

TALTON, J. O., LOU, Y., LESSER, S., DUKE, J., MĚCH, R., AND KOLTUN, V. 2011. Metropolis procedural modeling. *ACM Trans. Graph. 30,* 2 (Apr.), 11:1–11:14.

VANEGAS, C. A., ALIAGA, D. G., WONKA, P., MÜLLER, P., WADDELL, P., AND WATSON, B. 2010. Modelling the appearance and behaviour of urban spaces. *Computer Graphics Forum 29,* 1, 25–42.

WONKA, P., WIMMER, M., SILLION, F. X., AND RIBARSKY, W. 2003. Instant architecture. *ACM Trans. Graph. 22,* 3 (July), 669–677.

WU, F., YAN, D.-M., DONG, W., ZHANG, X., AND WONKA, P. 2014. Inverse procedural modeling of facade layouts. *ACM Trans. Graph. 33,* 4 (July), 121:1–121:10.

YANG, Y.-L., YANG, Y.-J., POTTMANN, H., AND MITRA, N. J. 2011. Shape space exploration of constrained meshes. *ACM Trans. Graph. 30,* 6 (Dec.), 124:1–124:12.

ZHANG, H., XU, K., JIANG, W., LIN, J., COHEN-OR, D., AND CHEN, B. 2013. Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph. 32,* 4 (July), 121:1–121:13.

Received Month YYYY; accepted Month YYYY