Proceduralization for Editing 3D Architectural Models Supplemental Material

Ilke Demir
Purdue University
idemir@purdue.edu

Daniel G. Aliaga
Purdue University
aliaga@purdue.edu

Bedrich Benes
Purdue University
bbenes@purdue.edu

A. Tree Construction

We didactically show how the split tree is constructed on a toy example (Figure 1). The boxes show the physical meaning of the nodes and the red points are the split points and they are stored on the edges.

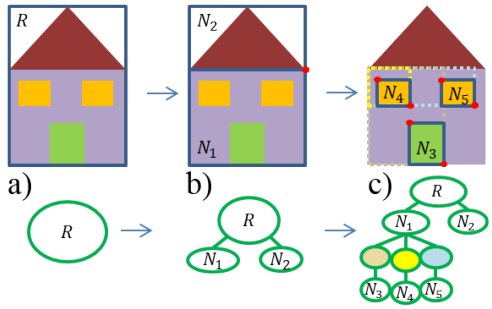

Figure 1. **Tree Construction.** (a) Building makes up the root, which is split into (b) next largest components, up to the (c) leaves.

B. Experiments: Input, Segmentation, and Beyond

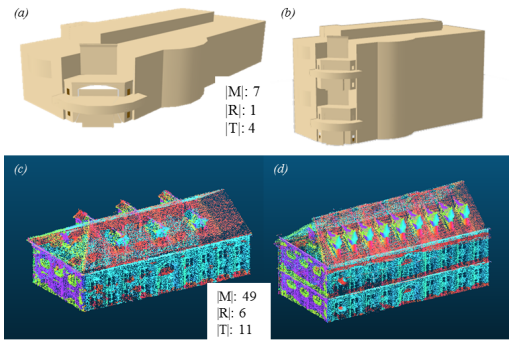

Figure 2. **Experiments.** (a) A trivial grammar and (b) editing result. (c) The inverse-modeled and (d) edited version of a building point cloud.

Supplemental Figure 2a shows a trivial example that is mentioned in the limitations part. If there is no repetition in the model, then the grammar is a basic one, still enabling the user to use the simple rules to generate new models.

We also applied our approach on a segmented point cloud of a building model (Supplemental Figure 2b). The segmentation of [2] was not applicable, so we used the semi-automatic segmentation approach of [3] for the building point cloud. We observed that our inverse procedural modeling is successful at generating the grammar, however the editing part needs additional resources to deal with the varying sampling density of the resized nodes.

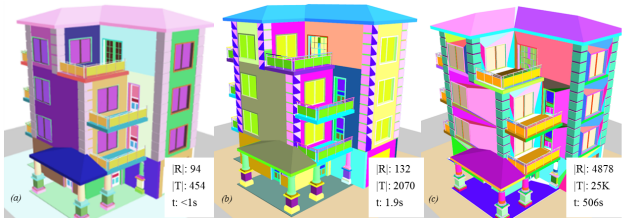

Figure 3. **Segmentation Sensitivity.** Components and grammar statistics of (a) a well-segmented building, (b) an over-segmented and labeled building, (c) an overly segmented building.

As mentioned in Section 3, our terminals and grammar expressivity depend on how the input elements of a building are segmented and labeled. To evaluate that sensitivity, we used three segmentations: a well-segmented and labeled building, a slightly over-segmented building with inconsistent labels, and an overly segmented and labeled building. As per the resulting terminals, non-terminals, and rules (Supplemental Figure 3); the terminal set in the most overly segmented building was not compact (25K terminals), however the rule-mechanism was able to find patterns proportional to terminals, even though $|\Sigma|$ in the worst case was greatly affected by over-segmentation. In all cases, our approach was able to convert the building into a procedural representation.

Note that we also tried several other known algorithms (fitting primitives, plumber and tailor, core extraction) on our dataset. However those algorithms hold some assumptions about models, and they cannot segment all buildings.

		Mesh/ PC	*Building/ Façade/ City*	*Automatic*	*Unspecified Grammar*	*Outputs Grammar*	*Provides Editing*	*Hierarchical Rules*	*Pattern Detection*
Editing	[Lip08]	Mesh	F	Manual	No	Yes	Yes	Yes	No
	[Lin11]	Mesh	B	Manual	NA	No	Yes	No	No
	[Kal12]	Mesh	B	Manual	NA	No	Yes	No	No
	[Zhe11]	Mesh	-	Semi	NA	No	Yes	No	No
IPM	[Wu14]	Mesh	F	Auto	Yes	Yes	No	No	Yes
	[Bok10]	Both	B, F	Semi	Yes	Yes	Yes	No	No
	[Bok11]	Mesh	B, F	Semi	NA	No	Yes	No	Yes
	[Bok12]	Mesh	B, F	Auto	NA	No	Yes	No	Yes
	[Tal11]	Mesh	B	Semi	No	Yes	No	No	No
	[Tal12]	Mesh	B, F	Manual	No	Yes	No	Yes	No
	[Hoh09]	PC	C, F	Manual	No	Yes	No	No	No
	[Tos10]	PC	B, C	Manual	Yes	No	No	No	No
	Ours	Mesh	B, C, F	Auto	Yes	Yes	Yes	Yes	Yes

Figure 4. **Comparison to Previous Work.** Blue rows: inverse procedural modeling approaches, green rows: structure-aware editing methods.

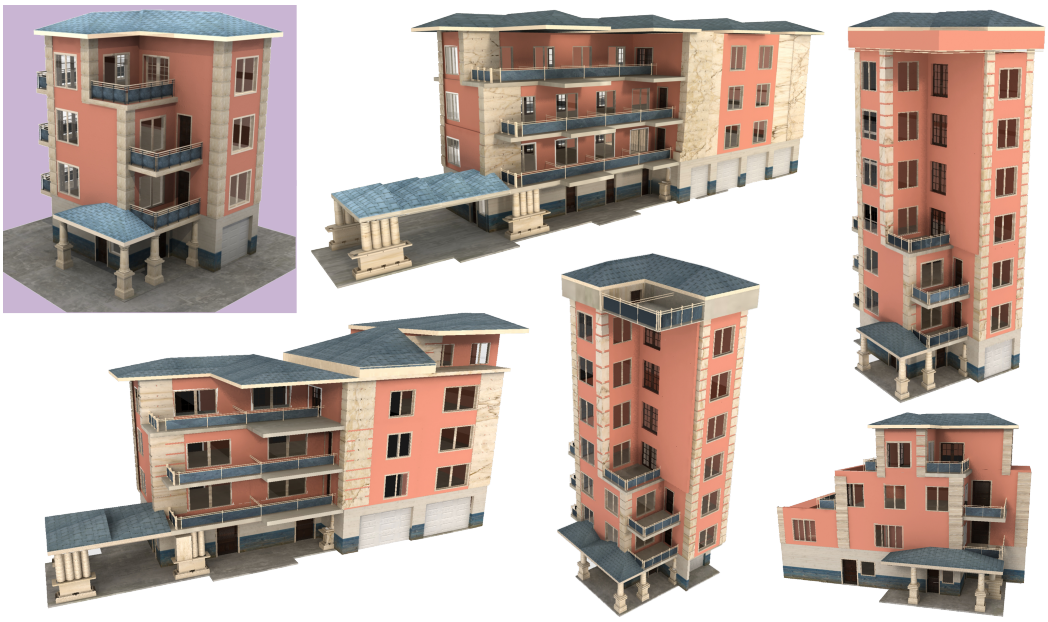

Figure 5. **Additional Results.** (purple) Original model, versus new buildings obtained by editing the extracted procedural representation.

C. Comparison to Previous Work

To start with, our approach does not employ any supervised, unsupervised, or deep learning. Supplemental Figure 4 summarizes our differences with some known papers in shape editing, and inverse procedural modeling. In general, most of them either do not produce a grammar, or start with a known grammar and focus on parameter estimation. Also, the support for hierarchical patterns, the editing control (Figure 7), not-needing labeled input and fully automaticity differentiates our method (Note that, if a method needs per-model parameter or threshold tuning, we listed it as "semi-automatic", considering that those numbers are user-dependent). In particular, our method can capture patterns (Figure 7, in contrast to Talton et al. [5]), and supports curved patches (Supplemental Figure 6, in contrast to Wu et al. [6]). We highlight that Figure 9, and Supplemental Figures 5 and 6 contain hierarchical and multi-dimensional regular patterns that are not all supported by Bokeloh et al. [1] and Lin et al. [4]. Finally, unlike Talton et al. [5], our method does not need training data or multiple instances of a model, just one model is enough to infer the instance.

D. Attachment Equations

For resize operation, we define a set of attachment equations as the following. A tree node box i is represented by two 3D vertices: the min-valued vertex $a_i = (a_{i_x}, a_{i_y}, a_{i_z})$ and the max-valued vertex $b_i = (b_{i_x}, b_{i_y}, b_{i_z})$ (i.e., for coordinates $c = x, y, z$, $a_{i_c} \leq b_{i_c}$). Further, we can organize

these six coordinates so as to define each of the eight box corner vertices $v_i k = (v_{ik_x}, v_{ik_y}, v_{ik_z})$ for $k \in [1, 8]$.

- **Ground attachment**: $a_{i_z} = 0$
- **Corner attachment**: $v_{ik_1} - v_{jk_2} = 0$where v_{ik_1} is the corner vertex k_1 of node i that is the same as v_{jk_2} of node j.
- **Edge attachment.** To define equations, assume $a_i \leq a_j$ and $b_i \leq b_j$ and they overlap in x. We define an edge ratio overlap function as:

$$e(s, t, u) = (t - s)/(u - s) \tag{1}$$

for $s \leq t \leq u$. The edge defined by v_{ik_1} and v_{ik_2} overlaps in the x coordinate with the edge in between v_{jk_3} and v_{jk_4} where $v_{ik_{1_x}} \leq v_{jk_{3_x}} \leq v_{ik_{2_x}} \leq v_{jk_4x}$. If $\hat{v}$ represents the initial (constant) vertex values, then edge attachment equations are:

$$\begin{aligned} v_{ik_{2_x}} - v_{jk_{3_x}} - v_{jk_{4_x}} - v_{jk_{3_x}} e(\hat{v}_{jk_{3_x}}, \hat{v}_{ik_{2_x}}, \hat{v}_{jk_{4_x}}) &= 0 \\ v_{jk_{4_x}} - v_{ik_{1_x}} - v_{ik_{2_x}} - v_{ik_{1_x}} e(\hat{v}_{ik_{1_x}}, \hat{v}_{jk_{4_x}}, \hat{v}_{ik_{2_x}}) &= 0 \\ b_{i_y} - a_{j_y} &= 0 \\ b_{i_z} - a_{j_z} &= 0 \end{aligned}$$

Similar equations are written for overlap in y and z.

- **Plane attachment.** Using the same notation as in edge attachments, the plane attachment equations for overlap in x and y are:

$$\begin{aligned} v_{ik_{2_x}} - v_{jk_{3_x}} - v_{jk_{4_x}} - v_{jk_{3_x}} e(\hat{v}_{jk_{3_x}}, \hat{v}_{ik_{2_x}}, \hat{v}_{jk_{4_x}}) &= 0 \\ v_{jk_{4_x}} - v_{ik_{1_x}} - v_{ik_{2_x}} - v_{ik_{1_x}} e(\hat{v}_{ik_{1_x}}, \hat{v}_{jk_{4_x}}, \hat{v}_{ik_{2_x}}) &= 0 \\ v_{ik_{2_y}} - v_{jk_{3_y}} - v_{jk_{4_y}} - v_{jk_{3_y}} e(\hat{v}_{jk_{3_y}}, \hat{v}_{ik_{2_y}}, \hat{v}_{jk_{4_y}}) &= 0 \\ v_{ik_{2_y}} - v_{jk_{3_y}} - v_{jk_{4_y}} - v_{jk_{3_y}} e(\hat{v}_{jk_{3_y}}, \hat{v}_{ik_{2_y}}, \hat{v}_{jk_{4_y}}) &= 0 \\ b_{i_z} - a_{j_z} &= 0. \end{aligned}$$

Same equations are written for overlap in yz and xz.

- **Volume attachment.** Volume equations are:

$$\begin{aligned} v_{ik_{2_x}} - v_{jk_{3_x}} - v_{jk_{4_x}} - v_{jk_{3_x}} e(\hat{v}_{jk_{3_x}}, \hat{v}_{ik_{2_x}}, \hat{v}_{jk_{4_x}}) &= 0 \\ v_{jk_{4_x}} - v_{ik_{1_x}} - v_{ik_{2_x}} - v_{ik_{1_x}} e(\hat{v}_{ik_{1_x}}, \hat{v}_{jk_{4_x}}, \hat{v}_{ik_{2_x}}) &= 0 \\ v_{ik_{2_y}} - v_{jk_{3_y}} - v_{jk_{4_y}} - v_{jk_{3_y}} e(\hat{v}_{jk_{3_y}}, \hat{v}_{ik_{2_y}}, \hat{v}_{jk_{4_y}}) &= 0 \end{aligned}$$

$$\begin{aligned} v_{ik_{2_y}} - v_{jk_{3_y}} - v_{jk_{4_y}} - v_{jk_{3_y}} e(\hat{v}_{jk_{3_y}}, \hat{v}_{ik_{2_y}}, \hat{v}_{jk_{4_y}}) &= 0 \\ v_{ik_{2_z}} - v_{jk_{3_z}} - v_{jk_{4_z}} - v_{jk_{3_z}} e(\hat{v}_{jk_{3_z}}, \hat{v}_{ik_{2_z}}, \hat{v}_{jk_{4_z}}) &= 0 \\ v_{ik_{2_z}} - v_{jk_{3_z}} - v_{jk_{4_z}} - v_{jk_{3_z}} e(\hat{v}_{jk_{3_z}}, \hat{v}_{ik_{2_z}}, \hat{v}_{jk_{4_z}}) &= 0 \end{aligned}$$

- **Size attachment.** For d equals one or more of (x, y, z),

$$b_{i_d} - a_{i_d} = |\hat{b}_{i_d} - \hat{a}_{i_d}| \tag{2}$$

where $\hat{a}_i$ and $\hat{b}_i$ are the initial (constant) vertex values.

- **Position attachment.** For d equals one or more of (x, y, z),

$$v_{ik_d} - \hat{v}_{ik_d} = 0 \tag{3}$$

E. Additional Results

In this section we show additional results that are obtained using our procedural editing engine after the procedural representation is extracted from the input models. In Supplemental Figure 5, the original model is colored with purple background, and all other models are synthesized from its grammar using our method, under 5 minutes. Note that users were not experts, thus interesting structures such as adding balconies to roofs are possible.

Supplemental Figure 6 shows other examples on different styles of architecture, such as St. Basil's drop shaped tops, Japanese style complicated roofs, neighborhoods such as Stanford Square, castles and towers as in Cindrella's castle, and domes as in Blue Mosque. Seeing the interesting grammars and editing results, we are hopeful that we have taken an important step towards automatic inverse procedural modeling.

References

[1] M. Bokeloh, M. Wand, and H.-P. Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29(4):104:1–104:10, 2010. 2

[2] I. Demir, D. G. Aliaga, and B. Benes. Coupled segmentation and similarity detection for architectural models. *ACM Trans. Graph.*, 34(4):104:1–104:11, July 2015. 1

[3] I. Demir, D. G. Aliaga, and B. Benes. Procedural editing of 3d building point clouds. *International Conference on Computer Vision (ICCV)*, Dec 2015. 1

[4] J. Lin, D. Cohen-Or, H. Zhang, C. Liang, A. Sharf, O. Deussen, and B. Chen. Structure-preserving retargeting of irregular 3d architecture. *ACM Trans. Graph.*, 30(6):183:1–183:10, Dec. 2011. 2

[5] J. Talton, L. Yang, R. Kumar, M. Lim, N. Goodman, and R. Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of the ACM UIST*, UIST '12, pages 63–74, New York, NY, USA, 2012. ACM. 2

[6] F. Wu, D.-M. Yan, W. Dong, X. Zhang, and P. Wonka. Inverse procedural modeling of facade layouts. *ACM Trans. Graph.*, 33(4):121:1–121:10, July 2014. 2

Figure 6. **Synthesis Examples.** (left) Original, and (right) re-modeled versions of real-world buildings. All models are taken from Google Warehouse.