

# Neural Procedural Reconstruction for Residential Buildings

Huayi Zeng<sup>1</sup>, Jiaye Wu<sup>1</sup> and Yasutaka Furukawa<sup>2</sup>

<sup>1</sup> Washington University in St. Louis, USA

{zengh, jiaye.wu}@wustl.edu

<sup>2</sup> Simon Fraser University, Canada

furukawa@sfu.ca

**Abstract.** This paper proposes a novel 3D reconstruction approach, dubbed Neural Procedural Reconstruction (NPR). NPR infers a sequence of shape grammar rule applications and reconstructs CAD-quality models with procedural structure from 3D points. While most existing methods rely on low-level geometry analysis to extract primitive structures, our approach conducts global analysis of entire building structures by deep neural networks (DNNs), enabling the reconstruction even from incomplete and sparse input data. We demonstrate the proposed system for residential buildings with aerial LiDAR as the input. Our 3D models boast compact geometry and semantically segmented architectural components. Qualitative and quantitative evaluations on hundreds of houses demonstrate that the proposed approach makes significant improvements over the existing state-of-the-art.

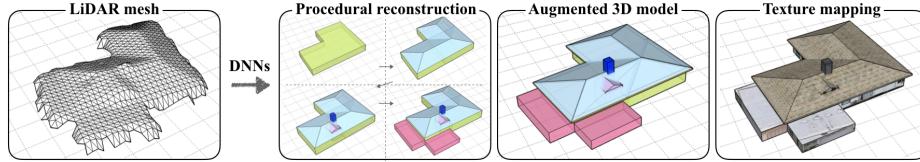
**Keywords:** 3D reconstruction, CAD, deep learning, procedural modeling

## 1 Introduction

Procedural modeling (PM) has revolutionized the practice of urban planning, architecture, and entertainment. PM procedurally applies shape transformation rules in a shape grammar to synthesize realistic 3D models, which have CAD quality geometries with procedural structures [9, 26, 8].

Discovering such procedural structure and reconstructing CAD quality geometry from raw sensor data, such as images or 3D point-clouds, is a similar but completely different problem [27], which we call procedural reconstruction (PR). A successful PR system could turn city-scale LiDAR scans into high-quality 3D city models with procedural structures, opening doors for novel applications in digital mapping, urban study, civil engineering, and entertainment. Unfortunately, most existing PR algorithms start from low-level geometry analysis in a bottom-up process (e.g., RANSAC for plane detection), requiring dense and near complete 3D points.

This paper proposes a novel approach, dubbed Neural Procedural Reconstruction (NPR), which trains deep neural networks (DNNs) to procedurally



**Fig. 1.** Neural procedural reconstruction learns to procedurally apply shape grammar rules to reconstruct CAD-quality models from raw 3D points. The procedural representation allows easy geometry augmentation (e.g., roof thickening) and texture mapping.

apply shape grammar rules and reconstruct CAD-quality geometry models from 3D points (See Fig. 1). DNNs detect primitive structures via global analysis of entire buildings, making the reconstruction possible even from incomplete and sparse 3D data. We demonstrated the framework for a shape grammar of residential buildings in England, where LiDAR point-clouds are publicly accessible. Qualitative and quantitative evaluations over hundreds of houses demonstrate that our approach makes significant improvements over the state-of-the-art. We will publicly share code and data to promote further research.

## 2 Related work

This paper makes contributions at the intersection of architectural reconstruction, procedural modeling, and procedural reconstruction. We focus the description to automated techniques in these fields.

**Reconstruction with geometric regularities:** Geometric regularities, such as planarity or orthogonality, have been effective for architecture reconstruction [10, 33, 39, 3, 38, 16, 37, 28]. Global geometric regularities can further improve the model quality [41]. Their 3D models are clean and compact, but miss procedural structure, limiting the range of applications.

**Procedural modeling and shape analysis:** Procedural modeling of architectural buildings has been a big success with many commercial products [9, 26]. A binary image or a volume guides the procedural modeling process of buildings by Markov Chain Monte Carlo (MCMC) [34]. However, their goal is to synthesize virtual scenes as opposed to faithfully reconstructing existing ones from sensor data. The analysis of man-made shapes through a structured representation has also been presented for objects [18, 19], buildings [6], indoor floorplans [24], and raw 3D point-clouds [5]. The analysis further enables model manipulation for interactive modeling [5] or even the discovery of a grammar [21]. Our problem is different: turning noisy sensor data into 3D models with procedural structure.

**Procedural reconstruction:** Procedural reconstruction has been an active research topic for building facades [27, 22], plants [31], and trees [35]. For building architecture, a seminal work by Dick et al. [7] employs MCMC to reconstruct structured building models from multiple images. MCMC is also used for the reconstruction of roof structures from laser scanned 3D points [14]. Recent techniques rely on machine learning to incorporate semantics [36, 20]. However, these

approaches critically rely on low-level geometry analysis for primitive extraction (e.g., RANSAC for plane detection), which is vulnerable to noisy or incomplete input data. Poor data quality is the key challenge for our problem, where the data comes from a country-scale survey [4] with much lower resolution than what has been commonly used [36, 20]. Our approach employs DNNs and conducts global analysis of entire buildings to detect primitive structures.

Integrating primitive detection with shape-rule inference robustifies the process [23], but the approach has been only demonstrated on one special building type: Greek temples. Top-down procedural reconstruction without primitive detection was proposed for indoor scenes [15], but the system requires many heuristics and hand-coded algorithms. A Support Vector Machine (SVM) performs roof type classification to generate CAD-quality building models [13], but their method requires rectangular building footprints as input. DNNs are used to guide the reconstruction of stroke-graphics [32]. They have a simple grammar for 2D strokes, while we handle 3D architectural buildings with more complex grammar.

The closest work to ours is the one by Nishida et al. [30], which utilizes DNNs to predict geometric primitives from user-strokes. While the fundamental role of DNNs is the same (i.e., classification of a rule-branch and regression of geometric parameters), our problem is substantially more challenging, requiring a different algorithmic solution. First, their input is clean feature-curves, while ours is raw sensor data. Second, their rules are limited to the generation of a single primitive at a time as an interactive system, while our rules need to generate an arbitrary number of primitives (e.g., multiple foundational blocks, dormers or chimneys). Third, their system infers a single rule application given user strokes, while our system needs to infer a complete sequence of rule applications as an automated reconstruction algorithm. To our knowledge, this work is one of the first<sup>3</sup> to demonstrate the use of DNNs for procedural reconstruction of architectural buildings from raw sensor data.

### 3 Shape grammar for residential buildings

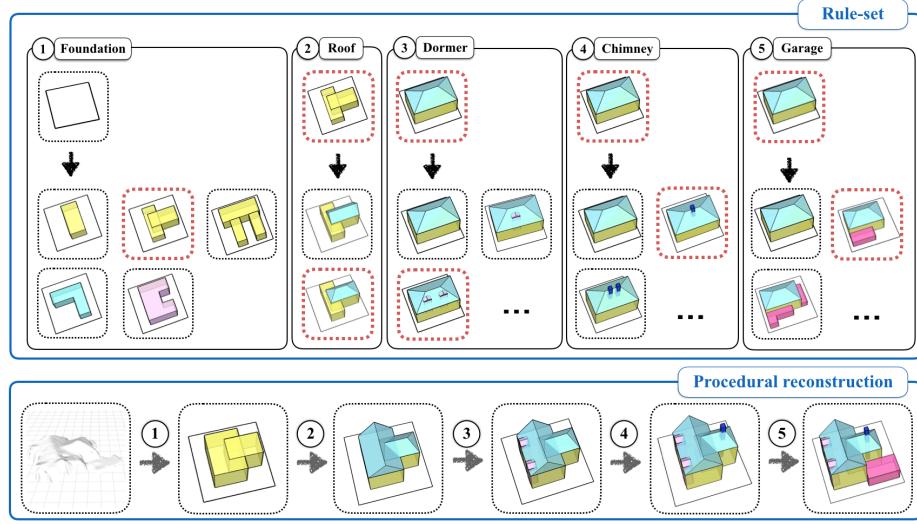
Our shape-grammar has seven rules, forming five reconstruction phases (See Fig. 2 ). We modified a default grammar in CityEngine [9] to focus on houses in England. Rules are applied in a fixed sequential order over the phases.

Each rule is associated with several branches and each branch has its own geometric parameters. We defer the full specification of our parameterization to the supplementary material, as the grammar definition is not our contribution.

- The “foundation-rule” first determines the 2D shape of a house from six types (i.e., rule-branches): I, II, III, L, U, or C. Houses consisting of one, two, and three

---

<sup>3</sup> About one week before the deadline, we encountered a future publication [29], which utilizes DNNs to reconstruct procedural building models from a RGB image. DNNs are used to parse each rectified facade image. This is not considered to be a prior work, but we cite the paper here for a reference.



**Fig. 2.** Our NPR system for residential buildings has 5 stages, consisting of seven shape-grammar rules. The application order of the rules are fixed while DNNs are trained to 1) select a rule-branch and 2) regress geometric parameters for each rule-application.

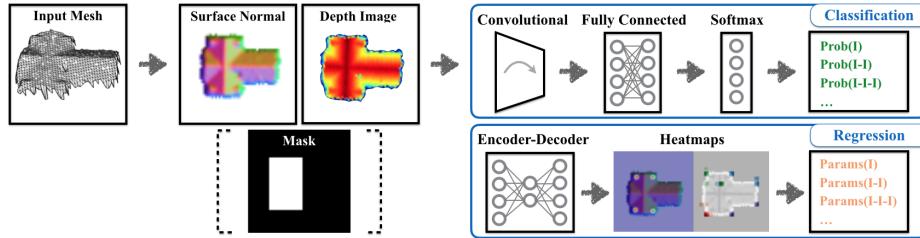
rectangular blocks have I, II, and III types, respectively. L (or U) types refer to L-shaped (or U-shaped) buildings.

The last type  $\mathcal{C}$  is for houses with complex shapes beyond our grammar or non-architectural structures such as trees, where our system would stop the reconstruction process. Geometric parameters are the position, the shape, and the height of each foundational block.

- Three “roof-rules” determine the roof structures for each foundational block, namely, I-, L-, or U-component. Hip and gable are the two popular types in England, and each rule has two branches. Geometric parameters determine the hip roof shape (i.e. hip ratio) and the roof height.
- The “dormer-rule” adds arbitrary number of dormers to each foundation component. We model dormers as a block primitive plus a gable roof. Therefore, the geometric parameters are the position and the shape of a block plus the roof height. The “chimney-rule” is the same as the dormer except that we assume its roof to be flat and its shape to be square.
- The “garage-rule” adds arbitrary number of surrounding sub-structures such as garages, balconies, or shelters as I- or L-polygons. The roof is fixed to flat.

## 4 Neural Procedural Reconstruction

Neural procedural reconstruction (NPR) applies shape-grammar rules in a fixed sequential order to reconstruct 3D models. NPR solves two fundamental tasks at each rule application: 1) classifying a rule-branch, and 2) regressing geometric



**Fig. 3.** Typical rule branch classifier and geometric parameter regressor. A variant of the ResNet [12] performs the classification. A standard encoder-decoder performs the regression. The regressors estimate the corners of building foundations for example. Our input is a 4-channel (surface normal + depth) image of resolution  $64 \times 64$ . The binary mask optionally specifies the region of interests.

parameters associated with the branch. DNNs perform all the branch classifications. For the parameter regression, DNNs play important roles, while we also use standard heuristics for some parameters, whose training data collection would require excessive manual work (e.g., 3D model manipulation). For DNNs, our input image has 4 channels encoding the surface normal and the depth, whose values are normalized to the intensity range  $[0, 255]$ . The resolution is  $64 \times 64$ . As part of pre-processing, we apply the orientation rectification DNN introduced in [25] and assume that building structures are axis-aligned.

#### 4.1 Foundation-rule

A variant of the ResNet [12] performs the rule branch classification (i.e., I, II, III, L, U, or  $\mathcal{C}$ ) with one modification: We add one more fully-connected layer between the global pooling and the output layers with softmax. A simple one-hot encoding is used with a cross-entropy loss (See Fig. 3). At test time, we simply pick a branch corresponding to the maximum probability.

I-, L-, or U-shaped 2D polygons are the regression targets. Directly regressing the parameters did not work even with DNNs. We borrow a standard encoder-decoder network to detect corner points in the output activation image [2]. We then enumerate possible polygon candidates, and pick the best one with a simple metric. We now explain the details for each rule branch.

- For I-shape, a DNN detects four types of corners in four activation images (i.e., top-left, top-right, bottom-left, or bottom-right). We extract peaks above a certain threshold (0.5 in our experiments) after the non-local max suppression. This process usually results in 2 to 3 corner candidates for each corner type. We then exhaustively pick three corner candidates from three corner types, and find the tight enclosing rectangle as a candidate. We find the rectangle with the maximum intersection over union (IoU) score against the binary mask of a house (the set of pixels with heights more than 2m) as the foundation.
- For II- or III-shapes, the process is the same except that we generate a pair or a triple of rectangles sequentially as one candidate.

- For L- or U-shapes, DNNs also detect four types of corners, whose types are defined in a scheme independent of the rotations (see the right figure). For instance, “internal” concave points are detected in the first activation image regardless of the rotations. We generate candidate polygons as follows. L-polygon has six corners and we exhaustively enumerate the set of six points to make one candidate (i.e., one from type-1, two from type-2, two from type-3, and one from type-4). Adjacent corners must have the same X or Y coordinate, and we simply take the average to make them consistent. We generate more candidates by enumerating five points (excluding one from the complete set), while inferring the missing one from its neighbors. The same process is used for U.

We extrude 2D polygons vertically after shape regression, while estimating the height by the average of the bottom 10% height values inside the polygon. Note that the final optimization refines all the parameters, and the parameters do not have to be precise in this step.

#### 4.2 Roof-rules

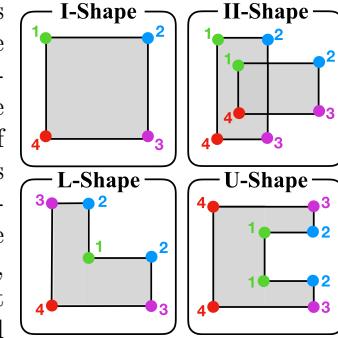
The same DNN architecture classifies the roof types (i.e., gable or hip). For II- or III-shapes, which consist of multiple I-shaped components with different roof types, a binary mask specifies an I-shaped component of interests by setting the mask values to 1 inside the corresponding I-component. Each height value is estimated by the average of the top 5% of the height values in the corresponding foundation. The ratios for hip-roofs are initialized with a common value 0.1.

#### 4.3 Dormer-rule and chimney-rule

The dormer roof structure is fixed to gable, and the rule has only a regression DNN-branch. The same encoder-decoder network [2] detects the center location of each dormer in a single activation image. We further apply the non-local maxim suppression and remove peaks lower than 0.5. For II- or III-foundations, we use the binary mask to specify I-components to add dormers. Dormers are initialized with an axis-aligned rectangle whose width and length are set to 2.0m. The roof height is initialized so that a roof angle is 30 degrees from the horizontal surface. The chimney-rule follows the same process except that its shape is initialized with a  $0.7m \times 0.7m$  square, and its height is initialized by 2.5m plus the roof height at the center position.

#### 4.4 Garage-rule

The garage-rule resembles the foundation-rule. However, we take a different approach, because arbitrary number of sub-structures are to be added, whose candidate enumeration would be challenging. We use the regression branch to infer



a set of pixels belonging to the sub-structures in an activation image. More precisely, the encoder-decoder network [2] performs pixel-wise regression. Higher value indicates the higher possibility of garage existence in each pixel. We keep pixels with values at least 0.5, find connected components whose diameters are at least 3 pixels, and try fitting rectangular or L-shaped polygons (with four rotation variants). Specifically, a tight bounding box is calculated for each connected component as a rectangle candidate. Starting from this bounding box, L polygon candidates are composed by replacing one corner by an internal point at least 4 pixels away from the boundary. We enumerate all possible polygon candidates and pick the best one based on the IoU against the binary mask of a house (i.e., same as the foundation-rule).

#### 4.5 Model refinement

We apply a standard non-linear least squares technique to optimize all the geometric parameters by minimizing the discrepancy against the input height values. We defer the details to the supplementary material.

### 5 Dataset creation

The section explains how we generate input depth, surface normal images, and ground-truth annotations.

#### 5.1 Depth and normal image generation

UK Environment Agency provides aerial LiDAR data over England as Digital Terrain Model (DTM) and Digital Surface Model (DSM). DTM only contains the terrain while DSM also contains buildings, vegetation, and other objects. The LiDAR data is arranged on the British National Grid [1]. Each grid covers a roughly square region. 25cm resolution (i.e., one sample per 25cm) is the highest resolution but covers only a small portion of the land. We downloaded 50cm resolution data of all the  $10km \times 10km$  grids. This amounts to roughly 500 grids. A grid with few houses has a small file size in a compressed form. We chose the top twenty grids based on the compressed file sizes, while skipping certain grids manually (e.g., complex mountainous terrains without houses).

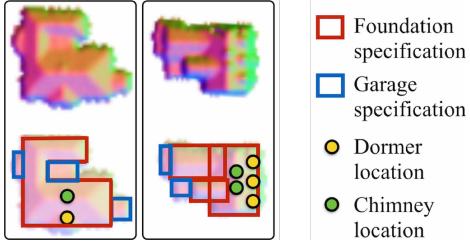
For each grid data, we subtract DTM from DSM to remove terrain influences. To isolate houses, we discard points below 2.0 meters, identify connected components, and remove small ones (i.e., areas less than 64 pixels). To further discard outliers, we use building footprints from Ordnance Survey (Britain's mapping agency). We enlarge each footprint by a factor of two around the bounding-box center, and keep only components that are fully inside at least one of the expanded footprints. The discarded components in the last step are marked as non-building structures for complex class ( $\mathcal{C}$ ).

For each remaining component, we estimate the rectification angle by an existing DNN-based system [25]. Bilinear interpolation is used for image sampling.

We find the tight axis-aligned rectangle, turn it into a square while keeping the center, then add a 20% margin all around. Lastly, we linearly map the height range in each square to [0, 255]. We use a finite-difference to also compute a surface normal image, which directly captures roof orientations. We linearly-map each vector element from its valid range ([-1.0, 1.0] or [0.0, 1.0]) to [0, 255].

## 5.2 Manual annotation

The right figure illustrates our typical annotations. For each normal image, we specify its foundational shape type (i.e., I, II, III, L, U, or  $\mathcal{C}$ ) with its 2D polygons and roof type(s) (i.e., hip or gable). Garage structures are annotated with either I- or L-polygons. The center locations are annotated for the



dormers or chimneys.<sup>4</sup> We do not annotate the remaining geometric parameters, which require time-consuming manual work: 1) shapes of dormers and chimneys; 2) heights of foundations and roofs; and 3) internal roof structures for hip. These parameters are initialized by default values or standard heuristics instead of the regression as described in Sect.4. We also rectify rotations manually to collect data for the rotation rectification DNN [25]. In total, we annotated 3,210 examples, in particular, 720, 1025, 142, 524, 247, and 552 samples for the foundation types I, II, III, L, U, and  $\mathcal{C}$ , respectively.

## 5.3 Data augmentation

First, we rotate depth images with an increment of 90 degrees with or without mirroring (factor of 8 augmentation). Second, since only a small percentage of houses contain dormers, we synthetically add dormer structures to the depth images, specially for the training of the dormer regression DNN. More concretely, for each house, we synthesize a new depth image by adding 1 to 5 synthetic dormers (the number randomly picked with uniform probabilities). Each dormer has five parameters (See supplementary materials) and we randomly specify each parameter with uniform probabilities within a specified range: 1) the center can be anywhere inside an image; 2) each lateral size is from 3 to 6 pixels; and 3) the height is from 0.5 to 2.5 meters. We repeat the process until the synthesized dormer is valid: 1) not colliding with any other dormers; and 2) residing inside only one foundational shape (i.e., not at the intersection of multiple I-shapes).

## 5.4 Synthetic data generation

The shape grammar allows generation of synthetic building models with ground-truth annotations via standard procedural modeling. While synthetic examples

<sup>4</sup> Dormers are annotated only for II-buildings to save time, as the trained model works on all the other cases. The dormer detection will be evaluated for all building types.

**Table 1.** Training and testing performance. Accuracies, IoU scores against the ground-truth, and (precision, recalls) are reported for the classification, foundation/garage regressions, and dormer/chimney detections, respectively.

Dataset	Foundation	Classification			Regression							
		Roof			Foundation				Dormer	Chimney	Garage	
		I	L	U	I	II	III	L				
Training	99.1%	96.1%	100%	100%	0.832	0.812	0.797	0.817	0.839	(100%, 100%)	(100%, 100%)	0.646
Testing	89.2%	85.0%	89.7%	66.0%	0.831	0.797	0.795	0.779	0.838	(77%, 84%)	(81%, 84%)	0.582

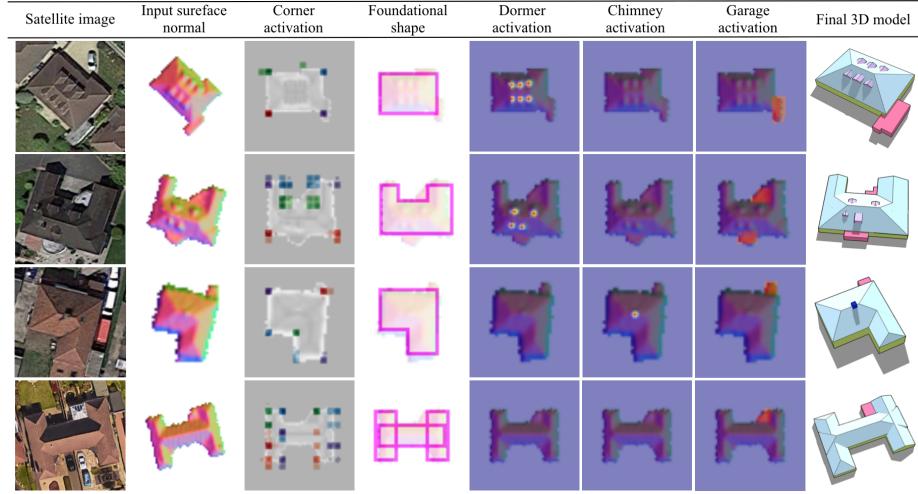
would not replace real data, it is still interesting to study how much they help training. We use Esri CityEngine [9] to generate synthetic house models, then sample 3D points at every 50cm to simulate the LiDAR scanning process. More specifically, we manually modify a default shape-grammar in CityEngine to better match the house examples in our data (See the rule-file in the supplementary document). To minimize the appearance gap between the real and synthetic images, we add uniform noise in a range [-0.3m, 0.3m] to the z coordinate (i.e., height) of each 3D point with probability 70%. Synthetic examples are axis-aligned at this point, and we randomly rotate each model around the gravity by uniformly picking an angle in the range [0, 360]. We generated 150,000 synthetic examples, in particular, 30,000 examples for each foundation type (i.e., I, II, III, L, or U).

## 6 Experimental results

We implemented the proposed system in C++ and PyTorch, using a standard PC with NVIDIA Titan X. We trained 4 classification DNNs (for the foundation and the three roof-rules), 8 regression DNNs (except the roof-rules), and 1 DNN for rotation rectification as pre-processing [25]. We used two thirds of the real data for training and the rest for testing after random sampling. Synthetic and augmented samples are used only for training. The encoder-decoder network has been initialized with a pre-trained model [2]. At test time, the network inference is instant (i.e., 100 to 150 instances per second), while the most expensive step is the model refinement, which takes about 10 seconds per house. We show only reconstruction results in the testing set. Please see the supplementary file for more results.

**Evaluations:** Figure 4 shows some of our representative reconstructions with intermediate results. A satellite image of the corresponding region is also shown at the left for reference. Our method successfully turns noisy raw sensor data into CAD-quality geometries with architectural components, whose segmentations are highlighted in colors in the 3D model renderings.

Table 1 shows the training and testing performance of our trained DNNs. Accuracies, IoU scores against the ground-truth, and precision/recalls are reported for the classification, foundation-garage regressions, and dormer-chimney detections (correct if within 3 pixels from the ground-truth), respectively. The



**Fig. 4.** Representative reconstructions with intermediate results. The satellite image is shown for reference from Google Maps.

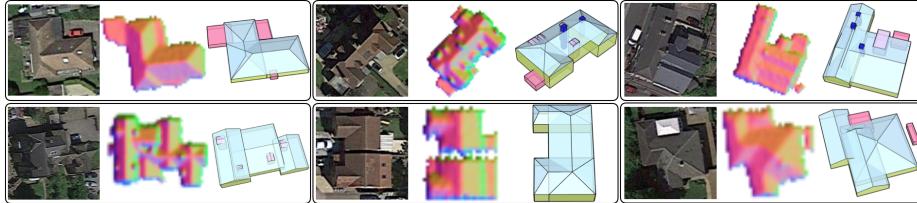
trained networks demonstrate good generalization performance for many tasks, but the testing accuracy suffers for U-roof classification, where U-shapes are rare and lack in real training data.

The confusion matrix of the foundation classification in the right figure also illustrates that U-shapes are one of the challenging cases together with III. The matrix shows confusion between II and L or III and U, where these cases are not clearly distinguishable even to human eyes. We also observe the confusion between III and C, because some III-shaped houses are quite complex.

	I	C	II	L	III	U	
0.972	0.009	0.008	0.008	0.001	0.002		0.8
0.125	0.632	0.105	0.059	0.066	0.013		0.6
0.011	0.032	0.858	0.089	0.007	0.002		0.4
0.030	0.019	0.097	0.840	0.011	0.004		0.2
0.040	0.150	0.070	0.020	0.620	0.100		0.0
0.045	0.063	0.000	0.036	0.108	0.748		

Figure 5 illustrates a few major failure cases: 1) the misclassification of foundation types; 2) missing dormers or chimneys; 3) local minima in the final refinement; and 4) unique architectural styles beyond our grammar. While the lack of ground-truth 3D models prevents quantitative evaluations, to our eyes, 20 to 25% of the examples fall into those failure modes.

**Comparative evaluations:** Figure 6 compares our algorithm against four competing methods: Poisson Surface Reconstruction (PSR) [17], piecewise planar reconstruction (PPR) [33], dual-contouring method (USC) [40], and semantic



**Fig. 5.** Major failure modes. Top-left: Foundation mis-classification (II mis-classified as L). Top-middle: Missing dormers/chimneys. Top-right: A local minimum in the final refinement, where an example is correctly classified as III. Bottom: Unique architectural styles beyond our grammar.

**Table 2.** The average mean absolute distance (MAD) against the input depth images and the average number of triangles (#tri) over 250 houses.

	I		II		III		L		U		Average	
	MAD	#tri										
Poisson	2.12	33.1	1.43	39.0	1.24	60.6	1.40	44.3	1.15	50.3	1.47	45.5
Piecewise planar	<b>1.08</b>	33.0	<b>0.92</b>	39.0	<b>0.94</b>	60.6	<b>0.99</b>	44.2	<b>0.80</b>	50.3	<b>0.95</b>	45.4
USC	1.64	166.1	1.36	158.1	1.48	213.3	1.44	154.6	1.10	152.7	1.40	169.0
Kentucky	1.61	12.9	1.81	19.9	3.13	10.0	2.19	12.4	1.90	13.2	2.13	13.7
Ours	<b>0.95</b>	33.1	<b>0.75</b>	39.0	<b>0.88</b>	60.7	<b>0.62</b>	44.4	<b>0.84</b>	50.3	<b>0.81</b>	45.5

decomposition and reconstruction (Kentucky) [20].<sup>5</sup> For PPR and Kentucky, their software is not publicly available and we used our local implementations.

Only Kentucky and our method generate semantically segmented CAD-quality geometry. However, Kentucky failed to capture main architectural structures in most cases. Two major failure modes stem from their plane extraction step. First, they often miss planes, especially small roof surfaces with clutter such as dormers. Second, they detect non-existent planes, which cause irreversible mistakes in merging pairs of planes to build high-level components. Note that our LiDAR is aerial and has only 4 samples per sq.m., making the plane-detection challenging. Kentucky uses ground-based LiDAR, which is probably 30 to 50 times denser. Despite challenges, our approach recovers even small architectural structures

<sup>5</sup> Three more potential competing methods exist. The first work is an old one with MCMC as the optimization engine [7], but we found it difficult to reproduce due to the complex algorithm and many parameters. The second work also utilizes MCMC [34]. While it may appear straight-forward, we found it difficult to adapt to our problem setting. They seek to roughly match the silhouette against a binary image/volume, allowing infinitely many solutions. Our input is noisy raw sensor data, where subtle geometric signals (e.g., small roof surfaces, chimneys and dormers) need to be recognized, allowing only one solution. The third work uses trained models to parse aerial LiDAR into architectural components [36]. We have not empirically compared, because their approach (bottom-up from primitive detection) is well-represented by Kentucky [20].

Normal image	Input depth mesh	Poisson	Piecewise planar	USC	Kentucky	Ours
	0	2.42 / 24	1.08 / 24	1.55 / 193	0.69 / 14	0.58 / 24
	0	1.29 / 79	0.84 / 80	1.58 / 240	1.50 / 28	1.06 / 80
	0	1.41 / 76	1.07 / 76	1.98 / 369	1.44 / 28	0.83 / 76
	0	0.96 / 92	0.75 / 92	1.83 / 145	1.61 / 42	0.70 / 92
	0	0.20 / 160	0.62 / 159	0.91 / 356	1.87 / 28	1.63 / 160

**Fig. 6.** Comparisons against baselines. The numbers show the mean absolute distance against the input depth images and the polygon counts.

such as dormers and chimneys through top-down learning instead of bottom-up primitive detection. The other methods produce a mere polygon soup without semantic segmentation or CAD-quality geometry. The figure demonstrates that our method makes significant improvements over the current state-of-the-art.

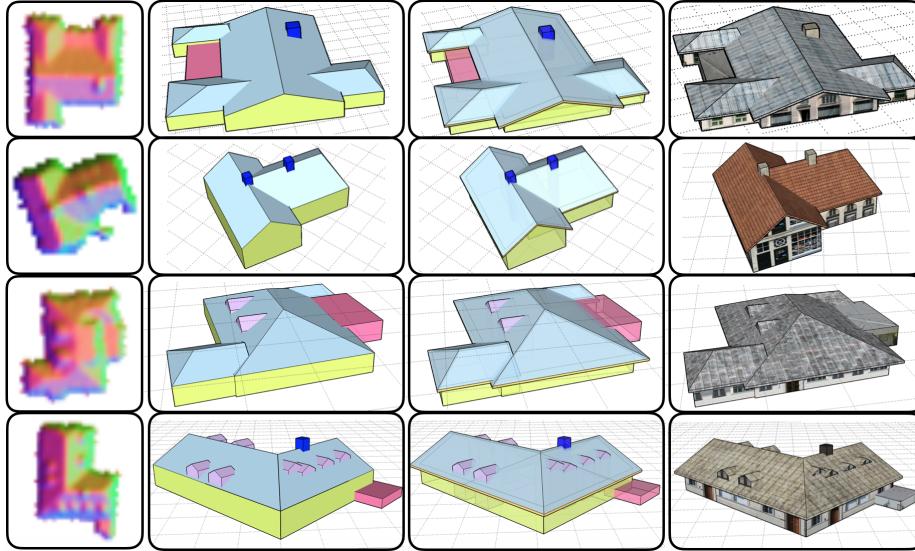
We quantitatively evaluated the accuracy of the models by taking the mean absolute distance between our model and the input depth images along the gravity direction. PSR and PPR generate dense polygon meshes, and we used a standard mesh simplification software QSLIM [11] to make the number of triangles roughly equal to ours for fair comparison. For USC, we tweaked the parameters of their software to make their models as compact as possible, although their polygon counts are still three to five times larger than ours.

The distances and the polygon counts are shown for examples in Fig. 4. Table 2 reports the averages over 250 houses, where we randomly chose 50 samples for each foundation type. Our method consistently outperforms the competing methods in the geometric accuracy. This is a surprise, as PSR+QSLIM seeks to decrease the polygon counts with a focus on maintaining the original geometry. Our method not only produces compact 3D models with proper segmentation and semantics, but also achieves the best geometric accuracy.

**Impact of synthetic and augmented data:** Table 3 shows how synthetic training samples influence foundation classification and regression. Classification improves with more synthetic data in general, while IoU scores do not improve

**Table 3.** Relative importance of the synthetic data. The first two rows show the classification accuracy of the foundation-rule. “x30” indicates that the number of synthetic examples for training is 30 times more than the real ones. The second row uses only synthetic samples for training. The bottom two rows show the regression accuracy (i.e., the IoU score against the ground-truth) for the II-foundation branch.

	0	$\times 10$	$\times 20$	$\times 30$	$\times 40$	$\times 50$
Accu. (w/ real)	82.2	85.3	85.6	87.7	89.2	88.7
Accu. (w/o real)	(N/A)	40.6	48.8	42.3	42.0	42.3
IoU (w/ real)	0.796	0.814	0.813	0.809	0.815	0.810
IoU (w/o real)	(N/A)	0.796	0.799	0.802	0.803	0.800



**Fig. 7.** Model enhancement. The procedural representation allows easy geometry augmentation and texture mapping. From left to right, an input surface normal image, our reconstructed model, an augmented 3D model, and a texture mapped 3D model.

much. We found that the raw outputs of the regression DNN degrade with fewer synthetic samples, but our post-processing (i.e., polygon candidate enumeration) is robust and yields good IoU scores even from poor corner detections. Augmentation of dormer structures improves the precision and recall from (8% and 17%) to (77% and 84%), where a detection is declared correct if within 3 pixels from a ground-truth.

**Applications:** The procedural representation easily allows for geometry augmentation and texture association(See Fig. 7). First, roof geometries do not have thickness nor overhang structures. We enlarge the roof geometry by 15%, then thicken the structure by extruding the surface along the gravity direction by 0.35m. Second, we used CityEngine and its texture-image asset to inter-



**Fig. 8.** Digital mapping is one key application domain. We rendered our models in Google Earth while changing the background texture from satellite to map-tiles. Our 3D models reveal rich architectural information with minimal increase in data size.

actively map textures, while the process can be automated by exploiting the reconstructed procedural structure. Besides content creation, digital mapping is another immediate application, which we demonstrate by importing our models to Google Earth (See Fig. 8). Our models reveal rich architectural information of houses with minimal increase in data size.

## 7 Conclusion

This paper presents a novel computational framework for procedural reconstruction, which turns noisy and sparse 3D points into high-quality 3D models with procedural structure. The qualitative and quantitative evaluations show significant improvements over the existing state-of-the-art. Our 3D models have CAD-quality geometry with semantically segmented architectural components, making them immediately effective for many applications. While the paper demonstrates the system for residential houses with LiDAR 3D points, neural procedural reconstruction is a general computational framework, independent of the particular choice of a shape grammar. This paper has a potential to enable effective procedural reconstruction system in other domains. A future direction of our works will be to make more effective use of synthetic examples and minimize the requirement on human annotations. We will publicly share all the code and data to promote further research.

## 8 Acknowledgement

This research is partially supported by National Science Foundation under grant IIS 1540012 and IIS 1618685, and Google Faculty Research Award. We thank Nvidia for a generous GPU donation.

## References

1. [https://en.wikipedia.org/wiki/Ordnance\\_Survey\\_National\\_Grid](https://en.wikipedia.org/wiki/Ordnance_Survey_National_Grid) (2017)
2. Bulat, A., Tzimiropoulos, G.: Human pose estimation via convolutional part heatmap regression. In: European Conference on Computer Vision. pp. 717–732. Springer (2016)
3. Chauve, A.L., Labatut, P., Pons, J.P.: Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1261–1268. IEEE (2010)
4. Data.gov.uk: <http://environment.data.gov.uk/ds/survey/index.jsp#/survey> (2015)
5. Demir, I., Aliaga, D.G., Benes, B.: Procedural editing of 3d building point clouds. In: International Conference on Computer Vision. pp. 2147–2155 (2015)
6. Demir, I., Aliaga, D.G., Benes, B.: Proceduralization for editing 3d architectural models. In: 3D Vision (3DV), 2016 Fourth International Conference on. pp. 194–202. IEEE (2016)
7. Dick, A., Torr, P., Cipolla, R.: A bayesian estimation of building shape using mcmc. In: European Conference on Computer Vision. pp. 574–575. Springer (2002)
8. Dylla, K., Frischer, B., Müller, P., Ulmer, A., Haegler, S.: Rome reborn 2.0: A case study of virtual city reconstruction using procedural modeling techniques. Computer Graphics World **16**(6) (2008)
9. EsriCityEngine: <http://www.esri.com/software/cityengine> (2017)
10. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Manhattan-world stereo. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1422–1429. IEEE (2009)
11. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques. pp. 209–216. ACM Press/Addison-Wesley Publishing Co. (1997)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
13. Henn, A., Grger, G., Stroh, V., Plmer, L.: Model driven reconstruction of roofs from sparse lidar point clouds **76**, 17–29 (02 2013)
14. Huang, H., Brenner, C., Sester, M.: A generative statistical approach to automatic 3d building roof reconstruction from laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing **79**, 2943 (05 2013)
15. Ikehata, S., Yang, H., Furukawa, Y.: Structured indoor modeling. In: IEEE International Conference on Computer Vision. pp. 1323–1331 (2015)
16. Jiang, H., Xiao, J.: A linear approach to matching cuboids in rgbd images. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2171–2178 (2013)
17. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Transactions on Graphics (TOG) **32**(3), 29 (2013)
18. Kim, V.G., Li, W., Mitra, N.J., Chaudhuri, S., DiVerdi, S., Funkhouser, T.: Learning part-based templates from large collections of 3d shapes. ACM Transactions on Graphics (TOG) **32**(4), 70 (2013)
19. Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: Globfit: Consistently fitting primitives by discovering global relations. In: ACM Transactions on Graphics (TOG). vol. 30, p. 52. ACM (2011)
20. Lin, H., Gao, J., Zhou, Y., Lu, G., Ye, M., Zhang, C., Liu, L., Yang, R.: Semantic decomposition and reconstruction of residential scenes from lidar data. ACM Transactions on Graphics (TOG) **32**(4), 66 (2013)

21. Martinovic, A., Van Gool, L.: Bayesian grammar learning for inverse procedural modeling. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 201–208 (2013)
22. Martinovic, A., Van Gool, L.: Hierarchical co-segmentation of building facades. In: 3D Vision (3DV), 2014 2nd International Conference on. vol. 1, pp. 409–416. IEEE (2014)
23. Mathias, M., Martinovic, A., Weissenberg, J., Van Gool, L.: Procedural 3d building reconstruction using shape grammars and detectors. In: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2011 International Conference on. pp. 304–311. IEEE (2011)
24. Merrell, P., Schkufza, E., Koltun, V.: Computer-generated residential building layouts. ACM Transactions on Graphics (TOG) **29**(6), 181 (2010)
25. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
26. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. In: Acm Transactions On Graphics (Tog). vol. 25, pp. 614–623. ACM (2006)
27. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. ACM Transactions on Graphics (TOG) **26**(3), 85 (2007)
28. Ngatatem, W., Mayer, H.: Modeling urban scenes from pointclouds. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 3857–3866 (Oct 2017). <https://doi.org/10.1109/ICCV.2017.414>
29. Nishida, G., Bousseau, A., Aliaga, D.G.: Procedural modeling of a building from a single image. Eurographics (2018)
30. Nishida, G., Garcia-Dorado, I., Aliaga, D.G., Benes, B., Bousseau, A.: Interactive sketching of urban procedural models. ACM Transactions on Graphics (TOG) **35**(4), 130 (2016)
31. Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., Kang, S.B.: Image-based plant modeling. In: ACM Transactions on Graphics (TOG). vol. 25, pp. 599–604. ACM (2006)
32. Ritchie, D., Thomas, A., Hanrahan, P., Goodman, N.: Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks. In: Advances In Neural Information Processing Systems. pp. 622–630 (2016)
33. Sinha, S., Steedly, D., Szeliski, R.: Piecewise planar stereo for image-based rendering. In: IEEE International Conference on Computer Vision (2009)
34. Talton, J.O., Lou, Y., Lesser, S., Duke, J., Mêch, R., Koltun, V.: Metropolis procedural modeling. ACM Transactions on Graphics (TOG) **30**(2), 11 (2011)
35. Tan, P., Zeng, G., Wang, J., Kang, S.B., Quan, L.: Image-based tree modeling. In: ACM Transactions on Graphics (TOG). vol. 26, p. 87. ACM (2007)
36. Toshev, A., Mordohai, P., Taskar, B.: Detecting and parsing architecture at city scale from range data. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 398–405. IEEE (2010)
37. Wang, H., Zhang, W., Chen, Y., Chen, M., Yan, K.: Semantic decomposition and reconstruction of compound buildings with symmetric roofs from lidar data and aerial imagery. Remote Sensing **7**(10), 13945–13974 (2015). <https://doi.org/10.3390/rs71013945>, <http://www.mdpi.com/2072-4292/7/10/13945>
38. Xiao, J., Furukawa, Y.: Reconstructing the world’s museums. International journal of computer vision **110**(3), 243–258 (2014)

39. Zebedin, L., Bauer, J., Karner, K., Bischof, H.: Fusion of feature-and area-based information for urban buildings modeling from aerial imagery. In: European Conference on Computer Vision. pp. 873–886. Springer (2008)
40. Zhou, Q.Y., Neumann, U.: 2.5D Dual Contouring: A Robust Approach to Creating Building Models from Aerial LiDAR Point Clouds, pp. 115–128. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
41. Zhou, Q.Y., Neumann, U.: 2.5 d building modeling by discovering global regularities. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 326–333. IEEE (2012)