DOI: 10.1111/cgf.13372

EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Procedural Modeling of a Building from a Single Image

Gen Nishida[1] Adrien Bousseau[2] Daniel G. Aliaga[1]

[1]Purdue University [2]Université Côte d'Azur, Inria

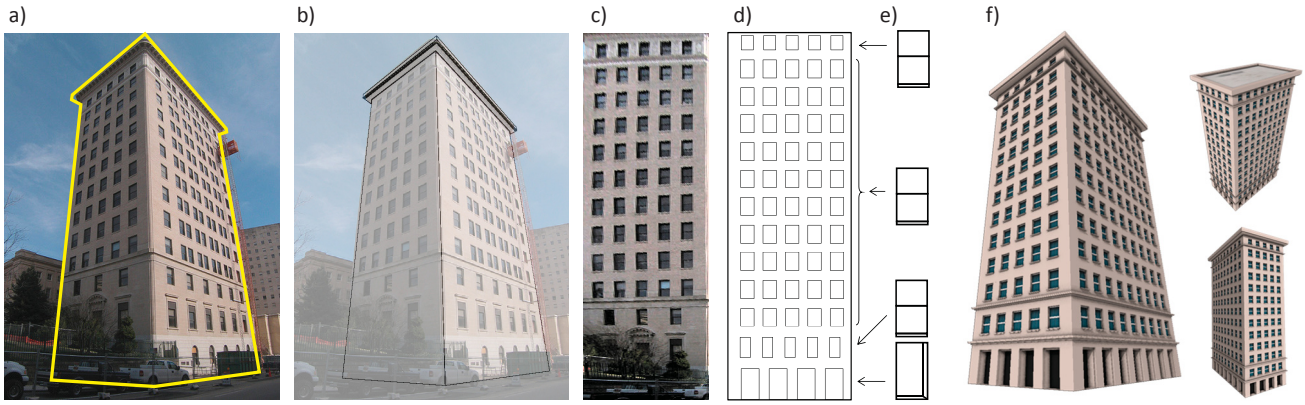

Figure 1: ***Procedural Modeling from a Single Image.*** *a) Given an image and a silhouette of a building, b) our approach automatically estimates the camera parameters and generates a building mass grammar as a first step. Then, c) the façade image is rectified, and d) the façade grammar is generated. e) For each window non-terminal, the best window grammar is selected by maximum vote. f) Finally the output grammar is constructed and a corresponding 3D geometry is generated.*

Abstract

Creating a virtual city is demanded for computer games, movies, and urban planning, but it takes a lot of time to create numerous 3D building models. Procedural modeling has become popular in recent years to overcome this issue, but creating a grammar to get a desired output is difficult and time consuming even for expert users. In this paper, we present an interactive tool that allows users to automatically generate such a grammar from a single image of a building. The user selects a photograph and highlights the silhouette of the target building as input to our method. Our pipeline automatically generates the building components, from large-scale building mass to fine-scale windows and doors geometry. Each stage of our pipeline combines convolutional neural networks (CNNs) and optimization to select and parameterize procedural grammars that reproduce the building elements of the picture. In the first stage, our method jointly estimates camera parameters and building mass shape. Once known, the building mass enables the rectification of the façades, which are given as input to the second stage that recovers the façade layout. This layout allows us to extract individual windows and doors that are subsequently fed to the last stage of the pipeline that selects procedural grammars for windows and doors. Finally, the grammars are combined to generate a complete procedural building as output. We devise a common methodology to make each stage of this pipeline tractable. This methodology consists in simplifying the input image to match the visual appearance of synthetic training data, and in using optimization to refine the parameters estimated by CNNs. We used our method to generate a variety of procedural models of buildings from existing photographs.

CCS Concepts

•***Computing methodologies*** → ***Shape modeling;***

1. Introduction

Procedural modeling is a popular way to create virtual architectures because it can generate varying content through simple parameter

changes. However, creating an architectural grammar can be difficult and time consuming. In this paper, we present a tool that allows the user to generate such a grammar automatically using a single example photograph.

Generating virtual models of buildings from existing imagery can be approached from several directions. On one extreme, multi-view stereo focuses on accurately reconstructing buildings from a multitude of images (e.g., [SCD*06]) but requires many images or a carefully selected set. At the other extreme, previous single-image modeling methods yield similar-looking building models but require significant user effort (e.g., [OCDD01]) or have very limited modeling abilities (e.g., [HEH05]). Further, none of these methods yield semantic information or a parameterized procedural model as output.

Our system combines procedural modeling with machine learning to automatically generate a parameterized 3D procedural model of a building from a single example (ground-level or aerial) photograph where the building has been outlined (Figure 1a). Our approach improves 3D content generation in at least two significant ways. First, we do not need multiple images from a variety of carefully chosen vantage points nor assume they can be found on the Internet. Second, from a single example photograph we automatically generate a family of similar style buildings by modifying parameters such as building height, window size, and even building shape.

The main challenge we faced in developing our system is the very large dimensionality of the space of camera parameters and parametric buildings. Existing inverse procedural modeling methods, using Markov Chain Monte Carlo (MCMC) (e.g., [TLL*11]) or sequential Monte Carlo (e.g., [RMGH15]), cannot successfully search this large space in a practical amount of time (e.g., see Table 2 and Figure 10 in our results section). Our novel methodology is based on the three following ideas.

- Our first idea to cope with the design-space complexity is to split the generation of a complete building into three, smaller stages: building-mass generation, façade generation, and windows generation. This decomposition matches the multi-scale structure of buildings and enables a sequential pipeline where each stage informs the next one: estimating the camera parameters and extracting the building mass (Figure 1b) assists in then rectifying the façades (Figure 1c), and extracting the façade layout (Figure 1d) helps to determine individual window styles (Figure 1e).
- Our second idea is to combine machine learning and optimization to quickly and precisely generate the procedural model and its parameters that best fit during each stage. In particular, we train a set of convolutional neural networks (CNNs) to recognize grammars and another set to estimate grammar parameters. We then refine these parameters with optimization methods. We found this three-step approach to be faster than popular MCMC optimization and more robust than hand-crafted heuristics. However, deep convolutional networks require a large number of annotated images for training, especially if one wants to be invariant to changes in lighting, texture, occlusion, and other sources of noise.
- Our third idea is to simplify the image content, when possible, for each stage to remove a great number of such variations, effectively turning the images into line drawings that capture geometric information without being polluted by photometric information. Abstracting the input image brings it to a similar visual domain as our training data, which we automatically synthesize by rendering models from the possible procedural design space. We only rely on manually-annotated training data for repetitive small-scale components of the building (e.g., floors, windows and doors), for which a relatively small number of annotations is sufficient to provide enough variety.

We show that our method successfully captures the essence of a wide variety of building styles and allows creative applications such as example building use in a virtual city. In particular, we evaluate our approach against several methods and several datasets. We automatically generate procedural building models using the first 50 office building photographs of ImageNet [RDS*15], the first 30 office building photographs of SUN [XHE*10], and perform comparisons for the main stages of our approach. Altogether, our method allows users to create buildings in around 20 seconds, including silhouette specification and fully automated processing.

In summary, our main contributions include:

- A complete system to generate a procedural model of a building from a single photograph with minimum user input,
- Novel independent modules that combine machine learning and optimization to recover building elements from a single image. In particular, we introduce a method to jointly estimate camera pose and building shape from a silhouette, and a method to parse façades that contain significant variety in style and sizes.

2. Related Work

Our work is at the intersection of image-based modeling and inverse procedural modeling of buildings. We refer readers to the survey by Musialski et al. [MWA*13] for an extensive discussion of urban reconstruction methods.

Automatic multi-view reconstruction. Structure-from-motion and multi-view stereo algorithms offer a robust framework to accurately reconstruct real-world 3D scenes from uncalibrated images [SCD*06]. However, multi-view stereo produces unorganized point clouds or surface meshes, while we aim to produce semantically-rich parametric models suitable for creative editing. Kelly et al. [KFWM17] recently made a significant step towards this direction by fusing multiple sources of information (coarse 3D mesh, street-level imagery, footprint) to produce structured 3D urban reconstructions. In contrast, we target the creation of a plausible procedural model using as little as a single photograph of a building.

Automatic single-view reconstruction. Automatically reconstructing buildings from a single image is a highly ill-posed task. The most successful approaches rely on machine learning to predict surface orientation or depth, effectively creating a pop-up of the visible surfaces in the scene [HEH05, LZcZ14, LSL15]. While we also employ machine learning, we train our algorithm to predict the type and parameters of procedural shapes. This procedural approach brings strong domain-knowledge that helps regularize the generation and predict occluded parts.

Part of our problem is to estimate the shape of the building,

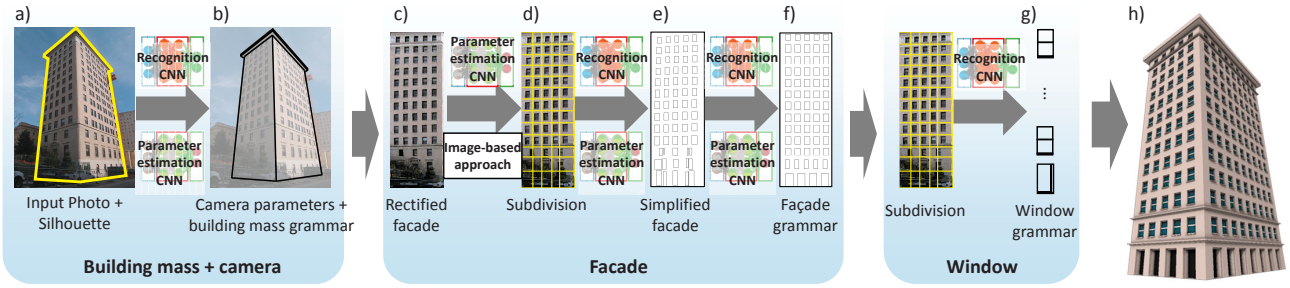

Figure 2: ***Our Pipeline.*** *a) Given a single image with the target building's silhouette highlighted, b) our system automatically estimates camera parameters and building mass shape. c) Then, one of the façade images is rectified and d) subdivided into tiles. e) For each tile image, the size and location of windows is estimated to create a simplified façade image. f) This image is used to generate a façade grammar. g) The tile images are also used to recognize the window type. h) The final output is a 3D building grammar and a corresponding building geometry that looks similar to the input photo.*

the intrinsic parameters of the camera (focal length and principal point), and the extrinsic parameters of the camera with respect to the building (rotation and translation). A number of approaches have been proposed in the computer vision literature to estimate part of these quantities from a single image. For example, methods based on vanishing points can recover the intrinsic camera parameters [HZ03], but require correspondences from other images or user annotations to reconstruct the object of interest [CDR99, GMMB00]. Model-based pose estimation methods attempt to jointly estimate the intrinsic camera parameters and its pose given an image of a known 3D object [Low87]. While early work relies on feature matching and non-linear optimization to align a single 3D model with its image [Mun06], the field progressively adopted machine learning techniques to deal with more complex object classes [ZSSS13, AME*14].

For example, recent machine learning techniques estimate camera (or object) orientation. These methods differ in their strategies to divide the orientation parameter range into discrete bins and/or perform a regression-like analysis. For example, Massa et al. [MMA16] create bins to represent different orientation values and yield errors of about 15 degrees in at most 40% of the cases (worse otherwise). Su et al. [SQLG15] also obtain orientation estimates with errors of over 10 degrees. Hara et al. [HVC17] obtain rotation estimates slightly under 10 degrees and provide comparative results to multiple other methods obtaining up to 40 degrees of error. While we take inspiration from these methods we make key adaptations to achieve the level of accuracy necessary for the subsequent steps of our pipeline. In particular, we cast camera intrinsic and extrinsic parameter estimation (8 parameters), as well as building mass parameter estimation (5-18 parameters), as a joint regression task. We simplify the task of the deep network by giving it a silhouette image rather than a realistic photograph. Further, this enables us to use our procedural modeling engine to generate a very large number of training images (300,000 images per grammar). Finally, we refine the prediction of the network with an optimization to tightly fit the 3D object to the input silhouette. Taken together, these improvements allow us to achieve fast and very accurate alignment of the building model with the input image (Table 2).

Interactive image-based modeling. User annotations can greatly help recover 3D models from a single or a few photographs. Existing interactive systems often combine annotations with heuristics for a limited set of shapes, such as cuboids [ZCC*12], generalized cylinders [CZS*13] and symmetric architectures [JTC09]. Other approaches such as the seminal façade system [DTM96] or the single-image method of [OCDD01] make less assumptions at the price of more involved user workflows. In contrast, our approach only needs users to provide the outline of the target building and is agnostic to the type of shape, as long as it can be expressed by a low-dimensional parametric model. The downside of our approach is that we are limited to the range of shapes expressible by the procedural grammars we train for; nonetheless, our evaluation shows that it can handle a wide variety of real-world buildings and our set can easily be extended.

Deep learning has recently been used to recover procedural models from line drawings in the context of sketch-based modeling [NGDA*16, HKYM16]. In particular, we build on recognition CNNs and parameter estimation CNNs to generate procedural grammars, as originally proposed by Nishida et al. [NGDA*16]. However, both [NGDA*16] and [HKYM16] assume a fixed viewpoint, while we train a CNN for the more complex task of jointly estimating the parameters of the procedural shape and of the camera. In addition, Nishida et al. [NGDA*16] require users to follow a strict multi-step modeling workflow while we only need a photograph with an outlined silhouette. Our evaluation shows that our system can generate buildings in a few seconds where Nishida et al. [NGDA*16] needs 5-15 minutes of interaction (Figure 9).

Inverse procedural modeling. Procedural modeling offers a compact and flexible representation of 3D models in the form of parameterized grammars [VAW*10]. However, creating a procedural grammar from scratch is notoriously difficult, which motivates research in inverse procedural modeling (IPM). Some IPM methods recover both grammar and parameter values from 2D data (e.g., [ARB07, SBM*10]) or from 3D models (e.g., [BWS10, DAB16]). Other methods focus on extracting parameter values for a given procedural model (e.g., [TLL*11, VGDA*12, RMGH15]). None of these methods however focus on parameter estimation to generate a 3D model from a single building image and its silhouette with-

out any further user intervention. Our approach is closer to the latter family of methods as we aim at estimating parameters for pre-defined grammars. However, we achieve high expressibility by defining a building as a combination of best-fitted grammars for mass, façade, and windows.

We have a similar goal as [VAB10] and one of the applications of [FW16]. Vanegas et al. [VAB10] extract textured 3D building masses from aerial photographs by extruding the provided building footprint and automatically applying floor transitions. However, they assume a Manhattan-world building and model façades as bitmap textures. Fan and Wonka [FW16] propose a parametric model of residential buildings and apply it to the reconstruction of houses from a single image. However, their approach requires the user to manually annotate all masses, façades, windows, and doors, including the selection of some geometry types (e.g., porch, garage, gable roof, hip roof, and flat roof).

Façade reconstruction. While there are many papers about façade reconstruction, most require the user to manually segment the façade and label each region (e.g., [MWW12, MG13, WYD*14]), and only a few papers deal with the automatic segmentation and labeling of the façade. We build on the method by Müller et al. [MZWVG07], which uses mutual information to segment the façade into tiles and recursively splits each tile based on an edge map to extract windows. However, we found that this method is quite sensitive to the distortions, noise and occlusions encountered in our rectified façades. We achieve higher robustness by training a CNN to estimate the procedural layout of the façade which in turn is refined by an image-based approach. Recently, several deep network architectures have been proposed for image segmentation [RFB15, BKC17] and façade parsing [LZZH17]. While these methods can produce accurate parsing, the output is a pixel-wise segmentation from which reconstructing a façade geometry is a challenging open problem.

The use of a procedural grammar to regularize façade analysis is similar in spirit to that of [KST*09, TSKP10, TKS*11, CSP14]. In particular, Koutsourakis et al. [KST*09] use Markov Random Field optimization to estimate the parameter values of a shape grammar that matches a rectified façade image. However, their method only supports one non-conditional grammar that covers a particular style of façade. In a follow-up work, [TSKP10, TKS*11] combine machine learning and procedural modeling for façade segmentation applied to Haussmannian style façades. A data-driven estimate of the segmentation is obtained using random forests trained on annotated façades, while a procedural model acts as a regularizer that penalizes labelings that do not follow the predefined rules. Cohen et al. [CSP14] uses dynamic programming to efficiently explore the solution space and produce higher quality solutions. These approaches can produce accurate segmentations for façade images that are similar to those in the training dataset. However, our experiments show that they make assumptions that do not generalize well to the greater variety of façades we target (see Figure 15 and Table 4). Note that all the aforementioned approaches were demonstrated on clean, well-rectified façade images. In contrast, we aim for a method that supports a wide range of façades and can be integrated into an end-to-end pipeline and as such be robust to distortions produced by approximate rectification and aliasing artifacts due to grazing-angle capture.

Overall, our objective is to provide a complete 3D procedural building model including the backside, façade, and window geometry from a single ground-level photograph. As far as we know, this is the first attempt to tackle this challenging problem.

3. Overview

Our approach automatically generates a procedural building model from a single photograph and a highlighted silhouette of the target building. Our system both estimates camera parameters and generates a grammar that yields 3D geometry similar to the target building. In this paper, we focus on street-level images because they are typically of high resolution. Nonetheless, our approach is general and is applied to aerial images as well.

Each of the three stages of our method (Figure 2) makes use of recognition CNNs to select appropriate procedural grammars and of parameter estimation CNNs to estimate their parameters. In addition, we propose dedicated simplification steps to match the visual appearance of the images with the training data and refinement steps that improve the estimated parameters with local optimizations. The recognition CNNs solve an image classification problem, for which we use AlexNet [KSH12] as it requires less training time than deeper networks like VGG16 [SZ14] and GoogLeNet [SLJ*15], while resulting in an almost equivalent accuracy in our preliminary experiments. Prior work on viewpoint estimation also advocates the use of classification tasks for parameter estimation [SQLG15, MMA16, HVC17]. However, estimating parameters via classification does not scale well to our multi-parameter problems, since it would require predicting $N \times M$ probabilities, where N is the number of parameters to estimate and M the number of discrete values that a parameter can take. Instead, we rely on a standard regression task, for which we modify the AlexNet architecture by removing the last softmax layer, redimensioning the output to match the parameter vector, and retraining all layers (see supplemental materials for more details on network architectures). In the following, we describe the functionality of each stage.

- **Building-mass generation.** Using the target building silhouette as a simplification of the input photograph, this stage uses a recognition CNN to find the building mass style, and then, uses a style-dependent parameter estimation CNN to determine grammar and camera parameters. In addition, we perform an optimization to refine the alignment of the building mass geometry to the photograph, which is critical to properly rectify the façades and proceed with the next stage.
- **Façade generation.** This stage proceeds with two main parts. The first part simplifies the largest area façade into a grid and locates within each grid tile the location and size, if any, of a window or door. Then, an image-based refinement is used. The second part feeds the aforementioned grid-based simplification of façades and windows/doors to a recognition CNN in order to determine the façade grammar style. The grammar parameters are then estimated using a style-dependent parameter estimation CNN. A subsequent optimization-based refinement improves the parameter values based on the simplified façade image. It is

worth noting that we came up with this novel façade generation approach after extensive experiments with existing approaches (see Figures 13 and 15 and Table 4 for details). None of the existing approaches could handle the wide variety of façade images we target.

- **Window generation.** Using the tiled procedurally-generated façade, a recognition CNN straightforwardly estimates the window style. Finally, we use a custom method to estimate façade colors.

Our final output combines the building mass grammar, the façade grammar, and the window/door grammars as well as all grammar parameter values. This collection of parameterized grammars is able to represent many different building configurations and can easily be used to create a city of buildings, as shown in Figure 18. More precisely, given an image of the target building I and its silhouette S, our pipeline estimates camera parameters θ^C and an output grammar $G = \left(G^M(\theta^M), G^F(\theta^F), G_i^W(\theta_i^W)\right)$, where G^M denotes the building-mass grammar, G^F denotes the façade grammar, and G_i^W denotes a set of window or door grammars. The building mass grammar contains parameters θ^M that specify the detailed size of the shape. The façade grammar contains the parameters θ^F to specify the floor height, width of columns, window size, margin size between windows, and style-specific parameters. The window/door grammars each have parameters θ_i^W that describe the 3D geometry. Please refer to the supplemental material for more details about the parameters.

Our approach and comparisons make use of several published datasets. The training images for our façade and window CNNs are created by altering images from the CMP dataset [Tv13] which contains a wide variety of façade styles. Our evaluation images are from ImageNet [RDS*15] and from SUN [XHE*10]. Our façade comparisons make use of the above datasets and also the ECP dataset [Teb11] (which contains Haussmanian style façades from Paris).

4. Building Mass Generation

The first stage of our method selects the grammar for the building mass G^M (Section 4.1), and jointly estimates parameter values of the camera and building mass, θ^C (Section 4.2) and θ^M (Sections 4.3 and 4.4), respectively, such that the silhouette of the generated procedural building matches the silhouette in the photograph.

Our approach to estimate these parameters is inspired by the recent use of deep learning to predict viewpoint from a single image of a known object category [SQLG15, MRA15, HVC17], and to predict parameters of procedural shapes drawn from a fixed viewpoint [NGDA*16,HKYM16]. However, predicting shape and viewpoint parameters concurrently requires significantly more training images to achieve sufficient accuracy. We discuss several strategies that we devised to filter out non-informative images from the training set. In addition, we describe how to refine the parameters estimated by the CNN using a derivative-free optimization.

4.1. Building Mass Grammar

We define grammars for 30 different building masses in a systematic way (Figure 3). We define six core building mass shapes having different footprints. Then, we add parameterized grammars supporting alternative roof shapes. We further add parameterized grammars with two and three stacks of building masses. While using only a limited number of grammars to represent building masses may restrict the supported shapes, this restriction enables us to generate the shape of the building mass from a single image, which is otherwise an ambiguous problem and cannot be solved in general. Please refer to our supplemental material for more information about per-grammar building mass parameters. Each image in Figure 3 shows one example by randomly selecting parameter values. We highlight that our framework is not limited to this set of grammars. Additional grammars, and thus shapes, can be added unless it incurs significant ambiguity with the silhouette produced by other already included grammars.

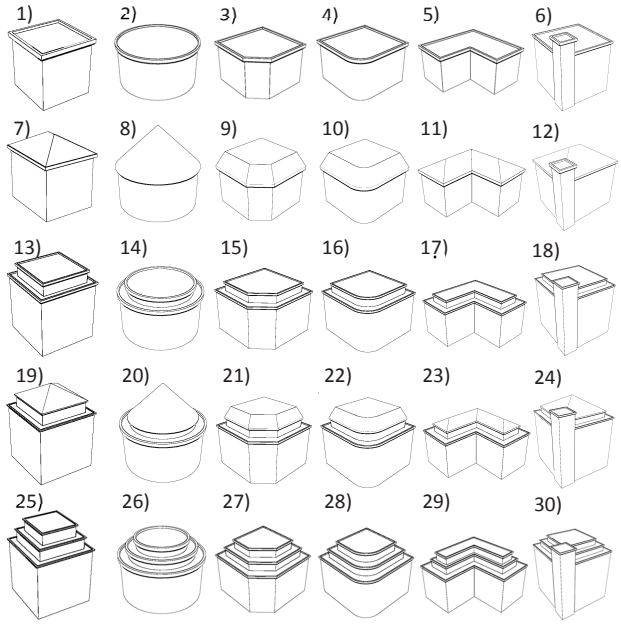

Figure 3: ***Building Mass Grammars.*** *We define 30 grammars for the building mass in a systematic way. The first row defines single stack buildings with different footprints. The second row of buildings has the same footprints but have different roof shapes. The third and fourth rows show buildings with two stacks of masses, whereas the buildings in the bottom row have three stacks of building masses.*

4.2. Camera Model

We adopt a camera model that covers at least most street-level photographs. The projection matrix of a typical pinhole camera is

$$M = K[R|t].$$

K is the intrinsic camera matrix represented by

$$K = \begin{bmatrix} f & 0 & C_x & 0 \\ 0 & f & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where f is the focal length and C_x and C_y are the coordinates of the principal point on the image plane. Also, R denotes the cam-

era rotation matrix, and $t = [t_x, t_y, t_z]^T$ represents the camera position [HZ03]. There are 9 parameters in total, but the coordinates projected by M are computed up to a missing scale factor because of homogeneous coordinates. To remove the scale ambiguity, we determine the z coordinate of the camera position using the Dolly zoom as follows:

$$t_z = \frac{L}{2\tan fov/2}$$

where L is the assumed width of the scene at roughly the depth of the building as captured by the photograph and fov denotes the approximate horizontal field-of-view. We fix $L = 25$ meters throughout the paper. As a result, we have a set of 8 parameters $\theta^C = (\phi_x, \phi_y, \phi_z, fov, C_x, C_y, t_x, t_y)$ to estimate, where ϕ_x, ϕ_y, and ϕ_z denote the rotation angles around x, y, and z axes. Once the camera parameters are estimated, the focal length f can be easily calculated from fov by

$$f = \frac{image\ width}{2\tan fov/2}$$

in order to construct the intrinsic camera matrix K as well as the projection matrix M. Our camera model does not take into account lens distortion nor other kinds of distortions. We found that these types of distortions are negligible compared to the parameters that we consider in our camera model.

4.3. CNN Training

For each building mass grammar, we synthesized 300,000 images of silhouettes to train the corresponding building mass parameter estimation CNN. In addition, we randomly selected 300,000 images among all these synthesized images to train the building mass recognition CNN. Both types of CNNs were trained for 160,000 iterations each with a batch size of 256. While automatically generating procedural models and synthesizing their silhouettes is straightforward, we must consider several aspects to improve the end quality.

- First, since we only render silhouette lines, we seek to eliminate ambiguous training images from which a unique building mass shape cannot be determined. To this end, we discard training images in which the top or bottom face of the bounding box of the building mass is visible. This is reasonable because images taken from the ground level usually do not have the top or bottom faces visible. Further, for building shapes that have slanted roofs or multiple stacks of masses, we also check if at least one of the roof faces or the side faces of each stack of masses is visible — if not, then the generated image is discarded.
- Second, we estimated the camera parameters of many images obtained from the Internet by using the vanishing point approach [CDR99, GMMB00], and from those estimations defined a fixed range of camera parameter values as follows:

$$\begin{cases} \phi_x \in [0, 40] \\ \phi_y \in [20, 70] \\ \phi_z \in [-10, 10] \\ fov \in [20, 90] \end{cases} \quad \text{and} \quad \begin{cases} C_x \in [-0.8, 0.8] \\ C_y \in [-0.8, 0.8] \\ t_x \in [-15, 15] \\ t_y \in [-15, 15] \end{cases}$$

  This restriction is also important because some extreme parameter values that rarely occur in the street-level images may result in ambiguous silhouettes. For instance, $\phi_y = 0$ means that the photograph is a front-view, which makes it difficult to infer the side shape of the building.
- Third, we discard all images in which the building is too big to fit within the image canvas. This is because if only a small fraction of the building is visible in the image, there will be no way to recover the complete 3D geometry faithfully. After computing the 3D geometry from the grammar, all vertices are projected to the image plane and if at least one vertex is outside the image by more than 3% of the image resolution, then the image is discarded
- Fourth, after the silhouette of the 3D geometry is rendered, the endpoints of the line segments are randomly translated within a range of [-1%, +1%] of the image resolution to train the CNN to be robust to minor errors in the input silhouette S.

4.4. Parameter Estimation

After the best building mass grammar is selected from the 30 candidates by the recognition CNN, our approach runs a dedicated parameter estimation CNN on the building silhouette to estimate the grammar parameters and calibrate the camera. While we considered using methods based on vanishing points to estimate part of the camera parameters [CDR99, GMMB00], we rejected this option for several reasons:

1. Outlining a building silhouette is an easier task than marking sets of lines converging to the same vanishing points, especially for novice users who have limited knowledge of perspective.
2. Vanishing point methods are not well suited for buildings that do not have dominant orthogonal orientations, such as cylinders (e.g., bottom left corner of Figure 7 and fourth column in Figure 8).
3. Even if we use an existing method to estimate camera parameters, this information needs to be provided in some form to the CNN in charge of estimating the building mass parameters, which is a challenge on its own.

We found that directly asking the CNN to jointly estimate camera and building parameters from the silhouette image is a simple solution to the above challenges, which proved to be very effective in practice. Nevertheless, while the parameter estimation CNN finds a good initial estimate of θ^C and θ^M, this estimate is often not precise enough to yield accurate façade rectification, as illustrated in Figure 11. This is reasonable considering the large parameter space of 8 camera parameters and up to 18 building mass parameters being simultaneously estimated. After multiple experiments, we found that using a numerical optimization approach is adequate to accurately refine the CNN-based parameter estimation. Hence, we define a distance metric between the input silhouette S and the silhouette $\hat{S}$ generated by the geometry using the estimated parameters. Let $D = DistTrans(S)$ be the distance transform of the input silhouette and $\hat{D} = DistTrans(\hat{S})$ be the distance transform of $\hat{S}$. Then, our distance metric is defined by

$$dist(S, \hat{S}) = \frac{1}{Z}\Big[\sum_{x,y \in I} D_{x,y} X(\hat{D}_{x,y} = 0) + \sum_{x,y \in I} \hat{D}_{x,y} X(D_{x,y} = 0)\Big]$$

Figure 4: ***Façade Generation.*** *a) Given the façade image, b) our approach uses a CNN to find the initial estimate of floor and column boundaries which are refined by the image-based approach. c) Then, for each tile image, window recognition CNN and window parameter estimation CNN are used to detect windows, and d) a simplified façade image is generated. For the façade region excluding the window region, a dominant color is calculated as a façade color. e) Finally, the façade recognition CNN and façade parameter estimation CNN are used to recognize the façade style and estimate parameter values for the selected façade grammar.*

where $X(\alpha)$ is an indicator function that returns 1 when α is true and return 0 otherwise, and Z is a normalization factor defined as

$$Z = \sum_{x,y \in I} X(\hat{D}_{x,y} = 0) + \sum_{x,y \in I} X(D_{x,y} = 0).$$

We minimize the aforementioned distance metric by a bound-optimization-by-quadratic-approximation (BOBYQA) algorithm [Pow09] to refine the parameter values:

$$(\hat{\theta}^C, \hat{\theta}^M) = \arg\min_{\theta^C, \theta^M} dist(S, \hat{S}).$$

This optimization is similar in spirit to traditional model-based pose estimation methods [Low87, Mun06], where our CNN prediction acts as a very good initial guess. We show in Section 7.3 that this optimization often ends up in poor minima without proper initialization. Note also that using only the target building (boundary) silhouette instead of all interior contour lines is easier to optimize as it reduces potential mismatches between the two shapes.

5. Façade Generation

Once camera parameters and building mass are estimated, our method selects and rectifies a façade to be used in our façade generation process. A common strategy for façade analysis consists in first labeling image pixels into elementary classes (walls, windows, doors, balcony, etc) before estimating a façade structure that best agrees with the labels while respecting architectural constraints [TSKP10,TKS*11,MMWVG12,CSP14,MMVG16]. However, we found that existing façade labeling algorithms trained on available datasets do not generalize well to our evaluation façade dataset (Figure 15). We propose an alternative approach that replaces the per-pixel façade labeling by a global estimation of the grid layout of the façade, and subsequently locates windows and doors within each cell of the grid (Figure 4a-d). Our experiments reveal that this coarse-to-fine approach trained on an existing dataset generalizes well to diverse façades. This process results in a simplified façade composed of windows and doors outlines, from which we estimate a façade grammar using synthetically-trained recognition and parameter estimation CNNs (Figure 4e). We now detail each step of our façade parsing approach.

5.1. Façade Rectification

Our approach first rectifies the visible façades of the building and selects the best representative for façade and window processing. Since our grammars for building mass include curved shapes (e.g., cylinders), we model a façade as one or multiple vertical, planar rectangular faces $\{f_i\}$. Then, a perspective transformation matrix T is calculated for each face f_i to rectify it. Finally, the rectified façade image is generated by combining the rectified faces. Please refer to the supplemental material for more details.

In some cases, the rectified façade image might be distorted because of errors in the input silhouette and the building mass generation. These distortions and other aliasing artifacts are amplified when the façade is captured at grazing angle. In addition, even when multiple façade images are rectified without significant distortions, the façades of hidden sides of the building remain unknown. We address these challenges by selecting the façade with the largest projected area and use it for all sides of the building.

5.2. Façade Simplification

Following the same strategy as for building mass, in this step we convert the façade image into a simplified line drawing that captures the structure of the layout without texture and lighting. Our approach is inspired by the façade grid subdivision algorithm of [MZWVG07], which analyses gradient magnitude along image rows and columns to locate transitions between floors and columns of windows. However, we found that this gradient-based approach is very sensitive to the textures, lighting and occlusions present in our façades. Instead, we train a CNN to robustly estimate the number of floors and columns in a rectified façade and only use the gradient-based method to refine the grid.

We trained the façade grid parameter estimation CNN using images based the CMP façade dataset [Tv13], which contains 378 façade images with the numbers of floors and columns already annotated. To avoid overfitting, we convert the images to grayscale and augment the dataset with random rotations, translations, horizontal mirroring, blurring, and/or changes of brightness. In total, we generated 50,000 façade images to train our CNN, which significantly improved generalization to façades other than those in the original dataset.

Following [MZWVG07], we refine the grid by aligning its boundaries with local minima of the image gradients projected along the vertical and horizontal axis, as measured by the following functions:

$$\begin{cases} V(y) = \sum_x |\frac{\partial I}{\partial x}| \\ H(x) = \sum_y |\frac{\partial I}{\partial y}| \end{cases}.$$

The local minima of $V(y)$ correspond to floor boundaries, whereas the local minima of $H(x)$ correspond to the vertical boundaries. However, since image gradients are often noisy, $V(y)$ and $H(x)$ contain many false local minima. Please refer to the supplemental materials for the visual analysis of $V(y)$ and $H(x)$. Müller et

al. [MZWVG07] exhaustively search among the many minima for the ones that yield the grid layout best respecting pre-defined floor height and tile width bounds. In contrast, the aforementioned CNN provides us with a good estimate of floor height and tile width automatically. We thus only perform a refinement of the grid by locally displacing the coordinates of the vertical and horizontal boundaries, x_i and y_j, to minimize $\sum_i H(x_i) + \sum_j V(y_j)$ while keeping floor height and tile width within 75% and 140% of the estimated values. Figure 13 illustrates the effect of this refinement step on the recovered façade subdivision.

After the façade is subdivided into tiles and refined, our approach tests for the existence of a window (or door) in each tile, and locates the position of the window's (or door's) bounding box. While edge information is often used to detect a window/door boundary (e.g., [LN04, MZWVG07]), we found that the edges extracted from the rectified façade image are usually very noisy. Instead, our approach uses a window recognition CNN to check for the existence of a window/door, and if it exists, a window parameter estimation CNN to estimate four parameters: the 2D relative coordinates of the top left corner of the window/door bounding box and its width and height. For these CNNs, we obtain the training images from the same grayscale CMP façade images as we use for the grid parameter estimation CNN. By subdividing the façade images into tiles, we extracted over 10,000 tile images, and manually annotated the position and size of the window in each tile. We further augment the training images with random rotation, translation, blurring, horizontal mirroring, and change of brightness. Note that when the tile image is rotated, translated, or mirrored horizontally, the coordinates of the window boundary are updated accordingly. These modifications significantly increase the variation in the training images and avoid overfitting. For the window recognition CNN, since there are severe unbalance in the number of tiles between those that contain a window (i.e., positive example) and those that do not contain a window (i.e., negative example), we augmented more negative examples to balance. Altogether, we generated 100,000 positive examples and 100,000 negative examples for training the window recognition CNN. The positive examples are also used for training the window parameter estimation CNN. Our results show that the trained CNNs can precisely detect windows/doors in our rectified façades (Figures 4c and 14).

Note that the floor heights and window tile widths computed in this step are not the final façade dimensions and might not be regular, if such is expected. During the next step, the façade structure will be regularized, if so discovered, in terms of floor heights and tile widths.

5.3. Façade Grammar

Once the simplified façade image F is computed (Figure 4d), we use a façade recognition CNN to recognize the best façade grammar and a façade parameter estimation CNN to estimate the parameter values. We define 16 façade grammars in a systematic way (Figure 5). The first four grammars have some variation vertically but no difference horizontally. The next four grammars have different styles in both axes. The next four grammars have different styles in the middle, and the last four grammars have different styles on both sides and in the middle. The same window color represents the same non-terminal. For instance, all the windows of façade grammar 1 are the same non-terminal. But, façade grammar 3 has three different non-terminals for the windows: a first one for the ground floor, a second one for the top floor, and a third one for other floors. Similarly to the building mass grammars, each grammar G^F contains many parameters θ^F to specify the details of the façade structure, such as the floor height, tile width, the position and size of the windows, and so on. Please refer to the supplemental material for more details about the parameters and their description. After parameters θ^F are estimated by the CNN, the BOBYQA algorithm is used to refine the parameter estimation based on the simplified façade image F in a similar manner to the mass parameter estimation.

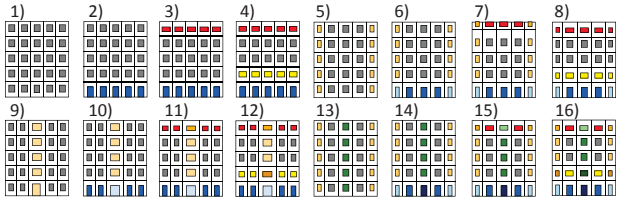

Figure 5: ***Façade Grammar.*** *The first four grammars have variation in a vertical direction but no variation horizontally. The next four grammars (5-8) have different styles at building edges. The first four grammars in the bottom row (9-12) have different styles in the building middle. The last four grammars (13-16) have different styles in building sides and in building middle.*

To train the façade recognition and parameter estimation CNNs, we synthesized 160,000 images in total, each of which show only window boundaries. Since the size and location of the detected windows in the simplified façade image may contain some noise and error, we perturbed the size and location of the windows in the synthesized images by adding a uniform noise that is up to 10% of the range of each parameter value. Then, the CNNs were trained with the synthesized images for 40,000 iterations.

5.4. Façade Color

Once the façade structure is identified and the façade is subdivided, we estimate the dominant color of the façade itself as a façade color. To do so, we run a k-means color clustering on the pixels that lie outside of the detected windows, and keep the centroid of the largest cluster as the façade color. We use k=10 clusters for all our results, and perform the color clustering in the perceptually uniform L*a*b color space.

We also considered computing one dominant color for each non-terminal node of the façade grammar. However, we found this local color computation to be sensitive to shadows and occlusions, which often cover significant parts of the façade (see supplemental materials for successful and failure cases).

6. Window Generation

To finish the building generation process, we use the estimated façade grammar to decompose the image into windows and doors tiles, and select for each tile a window/door grammar out of 31

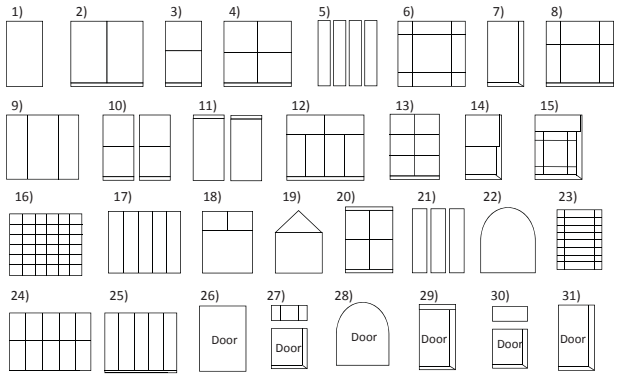

Figure 6: ***Window Grammars.*** *We define 31 window grammars including 6 entrance shapes inspired by 3D window models available on the Internet.*

predefined grammars using a window style recognition CNN (Figure 6). We then select the best window style for each non-terminal G_i^W by maximum vote among the corresponding tiles. This voting strategy makes our approach robust to local variations within similar tiles.

To generate the training images, we use the same tile images that are extracted from the CMP façade dataset as described in Section 5.2. This dataset results in over 10,000 tile images, for which we manually annotated the window type. To further increase the variation in the training dataset, we apply a similar augmentation as in Section 5.2 to generate 100,000 tile images in total. Then, the window style recognition CNN is trained for 40,000 iterations with the batch size of 256 using the generated images.

Note that while our method recovers procedural grammars for the windows and doors, we currently keep the parameters of these grammars fixed. While estimating the parameters of each window to best fit a given image would further improve the quality of the end model, it would require annotating a large number of images for each window type. We keep the recovery of such details, as well as other façade elements such as balcony, for future work.

7. Results

We implemented our approach on an i7-based PC workstation with 16 GB of memory and NVidia GTX980 graphics card. Our tool was implemented in C++ using OpenGL/GLSL. We used the Caffe library [JSD*14] as the framework to train the CNNs using GPUs and to generate building mass, façade and window geometries at run-time.

Our implementation includes 31 CNNs for building mass generation, 20 CNNs for façade generation, and 1 CNN for window generation. The input image for all CNNs is resized to 227 x 227 to enable using the pretrained weights of [KSH12] as initial values. Starting with pretrained weights quickens convergence and avoids overfitting during our training.

7.1. Generated 3D Buildings

Figure 7 shows some of the 3D building models generated by our approach. To evaluate the quality of our end-to-end system, we provide an extensive qualitative evaluation in the form of 80 buildings generated from external image collections. We selected the first 50 office building photographs from ImageNet [RDS*15] and the first 30 office building photographs from SUN [XHE*10] excluding aerial images and images that show only a part of the building, and highlighted the silhouette of a target building in each image. Then, our system automatically generated 3D building grammars and the corresponding 3D geometry. While some complex details are not captured, our approach generates a grammar that contains the prominent characteristics of the building in the input image. Please refer to our supplemental materials for the full set of results.

Figure 8: ***Aerial Images.*** *We applied our approach on aerial images by changing the range of camera parameters appropriately. In particular, we set* $\phi_x \in [10, 60]$ *for training the CNNs.*

We also applied our approach to aerial images (Figure 8), for which we changed the range of the camera parameter ϕ_x to $[10, 60]$ to train the CNNs. We observed a lower accuracy in camera parameter estimation on aerial images, which we attribute to the weaker perspective of such images, which are often captured from far away. Nevertheless, our approach manages to produce plausible building shapes and façades on these challenging cases.

7.2. Accuracy Evaluation

To evaluate the accuracy of the trained CNNs, we split each set of synthesized images into 80% for training and 20% for test datasets. For instance, we synthesized 300,000 images for each building mass grammar as described in Section 4.3. We randomly selected 80% of them as the training images to train the building mass parameter estimation CNNs. Then, the remaining images were used for evaluating the accuracy of the trained CNNs (Table 1). For other CNNs, please refer to the corresponding sections for the details about the image synthesis (Section 5.2 for the numbers of floors and columns, window existence, and window position and size, Section 5.3 for façade, and Section 6 for window style). Our results indicate that the trained CNNs generalize well to the test datasets without overfitting. Note that the computation time for window existence, window position and size, and window style is for each tile image. Thus, the total computation time depends on the number of tiles on a façade image, but for all selected building photographs, the total computation time was within 10 seconds.

7.3. Building Mass Comparison

We first compare our approach to [NGDA*16] (Figure 9) using the executable from the authors' website. Since [NGDA*16] uses a fixed viewpoint, the user has to mentally rotate the target building in the photograph to sketch. In addition, the user needs to draw each building element (building mass, roof, windows, and ledges) in a

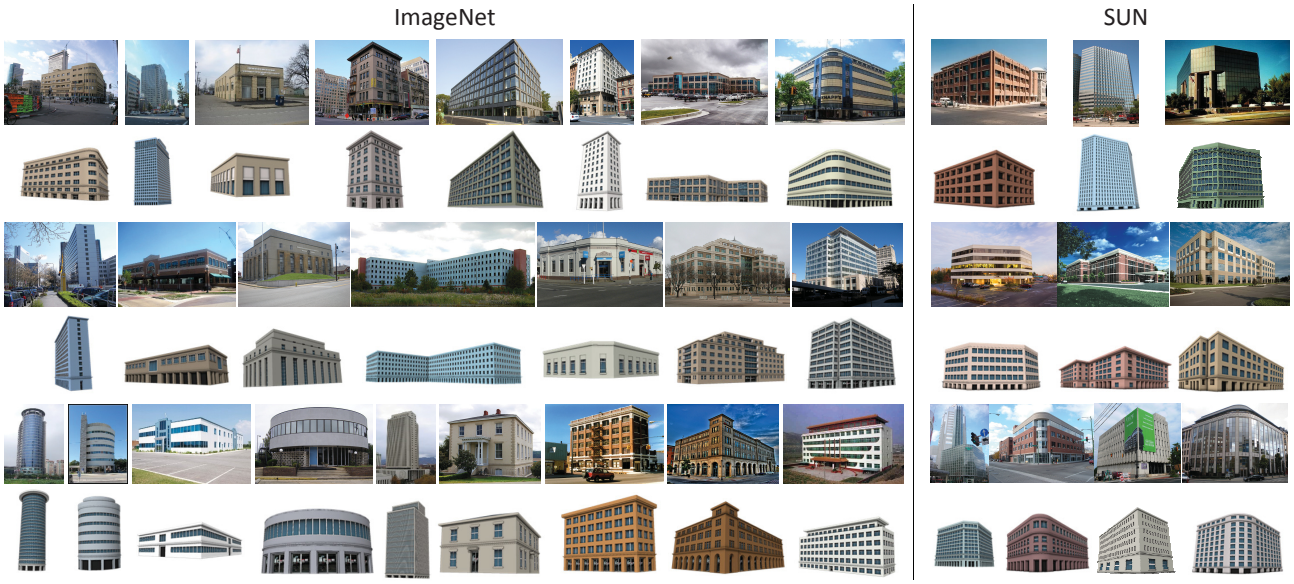

Figure 7: ***Generated 3D Buildings.*** *For each building, a photograph and highlighted target building silhouette are provided to our system (odd rows), and a procedurally-generated 3D building (even rows) is automatically generated. Left side of figure has images from ImageNet and right side has images from SUN dataset. Please refer to the supplemental materials for a complete set of input photographs and generated 3D buildings.*

Table 1: ***Accuracy of the Trained CNNs.*** *The trained CNNs were evaluated against the test dataset. The second column shows the classification accuracy, whereas the third column shows the average RMSE of parameter estimation. For #floors and #columns, the RMSE was calculated based on the predicted #floors and #columns, but for other parameter estimation CNNs, the RMSE was calculated based on the predicted parameter values that are normalized between 0 and 1. The last column shows the computation time.*

	Accuracy	RMSE	Time [sec]
Camera parameters	–	0.006	2.6
Building mass	99 %	0.004	
#floors, #columns	–	0.25	0.02
Window existence	93 %	–	0.02
Window position and size	–	0.058	0.02
Façade	95 %	0.006	0.04
Window style	91 %	–	0.02

fixed order, which takes around 4 minutes and multiple sketches. In contrast, our approach supports photographs from an arbitrary viewpoint, and it requires only a few mouse clicks followed by automatic processing to produce a visually similar result in much less time.

We also compare the building mass portion of our approach to other non-CNN-based methods (i.e., coordinate descent, nonlinear bound-constrained optimization, and MCMC) in terms of the accuracy and computation time (Table 2). In this comparison, we use for each of the non-CNN-based approaches the vanishing point method [CDR99, GMMB00] to estimate the camera parameters. Note that the relative position of the camera with respect to the

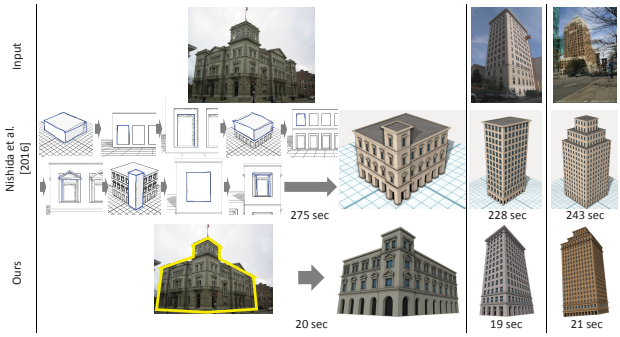

Figure 9: ***Building Generation Comparison.*** *[NGDA*16] requires the user to draw each element (i.e., building mass, roof, and windows), which takes around 4 minutes, and the fixed viewpoint makes it difficult for the user to sketch a target building. In contrast, our approach needs only a few user drawn lines and then computes a result fully automatically.*

building cannot be recovered by the vanishing point approach, so the camera position t_x and t_y has to be estimated by the non-CNN-based method. As expected, camera parameter estimation using the vanishing point method is very accurate given precise user input. However, the estimation for building mass parameters (i.e., width, depth, height) takes a lot of time for coordinate descent and for MCMC algorithms, and the accuracy of all the non-CNN-based approaches is very poor. In contrast, our approach can estimate both the camera parameters and the building mass parameters accurately within three seconds or less. Even though the accuracy of our estimation of camera parameters is inferior to the one by the vanishing

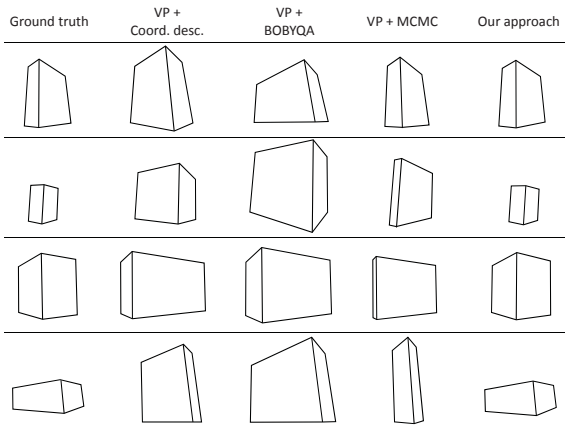

Figure 10: ***Visual Comparison of Building Mass Generation.*** *Given a silhouette of the ground truth image, the estimation results are compared between our approach and other non-CNN based approaches. Estimation by other approaches is very poor even though the camera parameters are precisely estimated by the vanishing point approach.*

Table 2: ***Comparison of Computation Time and Accuracy for Building Mass Generation.*** *Each cell shows the average estimation error (100 trials) as a percentage of the parameter value range. We compare our approach to coordinate descent, BOBYQA, and MCMC for building mass estimation, using vanishing points to estimate the camera parameters except the camera position. We use the distance metric defined in Section 4.4 to evaluate accuracy.*

		VP + Coord. desc.	VP + BOBYQA	VP + MCMC	Ours
Camera	ϕ_x	0.002 %	0.002 %	0.002 %	0.08 %
	ϕ_y	0.004 %	0.004 %	0.004 %	0.1 %
	ϕ_z	0.005 %	0.005 %	0.005 %	0.2 %
	fov	0.003 %	0.003 %	0.003 %	0.01 %
	C_x	0.19 %	0.19 %	0.19 %	2 %
	C_y	0.13 %	0.13 %	0.13 %	1 %
	t_x	0.3 %	0.3 %	0.2 %	0.2 %
	t_y	0.3 %	0.8 %	0.2 %	0.1 %
Building	W	8 %	35 %	8 %	2 %
	D	10 %	10 %	8 %	2 %
	H	5 %	25 %	8 %	1 %
$dist(S, \hat{S})$		7657	14922	3103	0.85
Time [sec]		95.5	1.8	9531.8	2.6

point approach, our distance metric (Table 2) and visual comparison (Figure 10) indicate that it is sufficiently accurate for façade generation.

7.4. Building Mass Refinement

Figure 11 shows how the refinement of the building mass estimation improves the rectified façade image. While the building mass parameter estimation CNN can provide a good initial estimate for the camera parameters and the parameters of the building mass, the rectified façade image without refinement still shows some noticeable error (e.g., the cropped top floor and left side). The refinement by BOBYQA improves the parameter estimation for the building mass.

Figure 11: ***Effect of Building Mass Refinement.*** *a) Without refinement of the building mass estimation, there is a noticeable error in the rectified façade image. b) Our refinement optimization suitably improves the resulting façade image for the subsequent façade generation stage.*

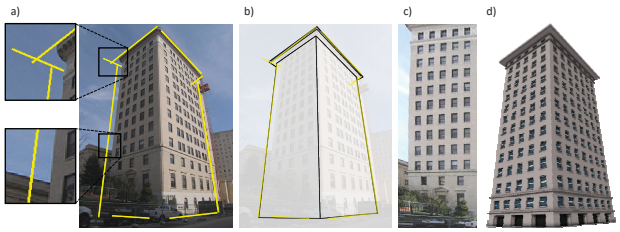

Figure 12: ***Robustness to the Input Silhouette.*** *Our building mass (b) and façade grammar estimation (d) is robust to misalignment due to inaccurate input silhouettes (a), despite approximate façade rectification (c).*

The accuracy of the provided building silhouette benefits our method (Figure 1). However, our building mass stage is robust to some silhouette inaccuracy and discontinuity (see Figure 12a and b). While the quality of façade image rectification is more sensitive to inaccurate silhouettes (see Figure 12c), our façade stage is heavily guided by window layout which is less affected by the silhouette errors. Thus, the final result is still a good representation of the captured building (Figure 12d). In addition, the constrained space of the building mass and façade grammars also contributes to the overall robustness of our system.

7.5. Façade Subdivision Evaluation

Figure 13 shows the improvement of façade subdivision by our approach compared to pure CNN-based and image-based approaches. The pure CNN-based approach estimates the floor height and column width, and this information is used to uniformly subdivide the

Table 3: ***Quantitative Evaluation of Façade Subdivision.*** *Our façade stage accurately estimates the number of floors and the number of windows per floor.*

	ECP		Office building	
	#floors	#windows	#floors	#windows
RMSE	0.10	0.22	0.55	0.50

Figure 13: ***Comparison of Façade Subdivision.*** *The top row shows the façade subdivision by a CNN-based method, which uniformly subdivides the façade based on the estimated floor height and column width. The second row shows the results by an image-based method using predefined numbers of floors and columns to estimate the floor height and column width. The bottom row shows improved subdivision by our approach.*

façade. Our approach uses this result as an initial estimate but refines the boundaries based on the gradient magnitude. As a result, it can handle different floor heights and column widths in the same façade. We also compare our approach with an image-based approach [MZWVG07] that uses predefined ranges for floor height and column width as constraints and performs an exhaustive search to find the best set of boundaries based on the gradient magnitude. For this image-based approach, we obtained the a priori constraints by computing the average numbers of floors and columns of the CMP façade images, and deduce the floor height and column width for each façade. As a result, since the computed floor height and column width is approximate, the resulting subdivision may detect false boundaries or miss some boundaries — its performance depends highly on the accuracy and tightness of the a priori constraints. Table 3 shows a quantitative evaluation of the number of floors and windows estimated by our method. The root-mean-square errors (RMSE) are all less than 1, indicating relatively good estimates.

7.6. Window Detection Comparison

Figure 14 shows a comparison of window detection between an edge-based method and our approach. The edge-based method relies solely on the edges. For each tile image, the edge-based method runs a Canny edge detector, and the smallest rectangle that contains all the edges is detected as a window boundary. This method is not sufficient to precisely extract a window boundary especially given many false edges or blurry/occluded window boundaries. As a result, an edge-based method alone may fail to find window boundaries correctly (top row of Figure 14). In contrast, our approach uses a CNN that exploits not only the edges but also other features such as color and texture to find window boundaries, which significantly improves window detection (bottom row of Figure 14).

We also compare our approach with previous façade parsing techniques [WFP10, TKS*11] (Figures 15). In addition, we include a comparison to the fully-convolutional segmentation approaches, U-Net [RFB15] and DeepFacade [LZZH17]. We cropped the ECP façade images [Teb11] by removing the roof region since our approach extracts only the façade region excluding the roof. We also used our office building images, which consists of 80 rectified façade images, for this comparison. The dataset of the façade images and the ground truth labels are provided as the supplemental materials. For [TKS*11], we used the trained random forest provided by the authors for the initial parsing and ran Q-learning for 5,000 episodes to obtain the results. For U-Net and DeepFacade, the model was trained for 100 epochs using randomly selected 80

Figure 14: ***Comparison of Window Detection.*** *The top row shows the windows detected by an edge-based approach, whereas the bottom row shows our more accurate results.*

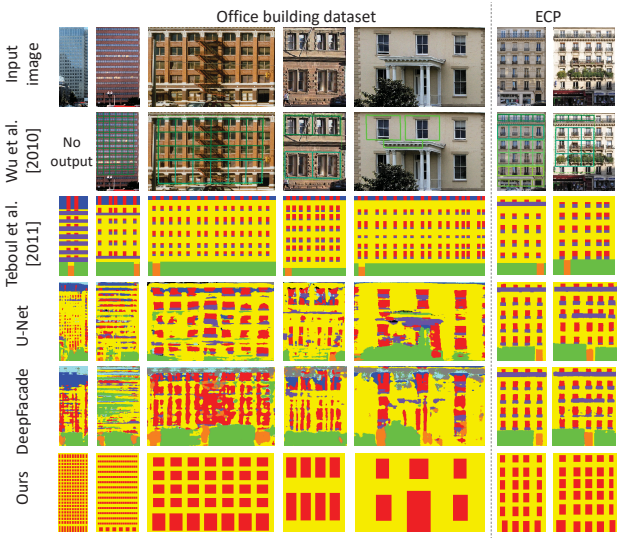

Figure 15: ***Façade Parsing Comparison.*** *[WFP10] detects only the repetitions of texture instead of the actual windows. [TKS*11], U-Net [RFB15], and DeepFacade [LZZH17] classify the façades into 7 classes: window (red), wall (yellow), balcony (purple), door (orange), roof (blue), sky (sky blue), and shop (green). Our approach detects only the windows, so the results show only two classes: window (red) and wall (yellow). While existing approaches perform well on the ECP dataset, they mis-classify many windows on our office building façades (see supplemental materials for more results) and tend to hallucinate shops at the ground level. In contrast, our approach produces results of similar quality on both dataset.*

Table 4: ***Pixel-Wise Accuracy of Façade Parsing.*** *For the ECP dataset, the images are cropped to remove the roof region since our building mass stage extracts only the façade region. Also, since previous approaches classify façade elements in more classes than ours, we convert window, door, and shop classes to window, and all other classes to wall. For each approach, F1-score and accuracy of classification is measured. In the supplemental document, the precision and recall are also shown in Table A. Overall, while previous approaches outperform ours on the ECP dataset, their accuracy degrades significantly on our more general office façades. In contrast, our approach achieves a similar accuracy of around 0.75 on both datasets. The pictorial quality of accuracy 0.75 can be appreciated in Figure 15.*

		Teboul et al. 2011	U-Net	Deep Facade	Ours
ECP	Window	0.73	0.94	0.91	0.58
	Wall	0.89	0.94	0.96	0.82
	Total	0.85	0.96	0.95	0.75
Office building	Window	0.42	0.48	0.56	0.69
	Wall	0.66	0.73	0.71	0.78
	Total	0.57	0.64	0.65	0.74

images of the ECP dataset. [WFP10] detects only the repetition of texture, so the actual windows cannot be detected. [TKS*11], U-Net, and DeepFacade work well for the Haussmanian-style facades of the ECP dataset, but fail to detect windows for a wider variety of office façade images (Figure 15 and Table 4). Teboul et al. [TKS*11] use a random forest for the initial façade parsing and refines it using a predefined façade grammar via reinforcement learning. While the façade grammar works as constraints to improve the parsing, the limited number of epochs often results in a poor local minimum unless a good initial parameter estimation is provided. In contrast, our approach uses the façade parameter estimation CNN to quickly get a good initial parameter estimation, which enables our approach to work well for both the ECP dataset and more general office building façade images. The fully-convolutional approaches [RFB15, LZZH17] obtain accurate parsing on the ECP dataset, but produce noisy results on our façades which contain windows at many different scales.

7.7. Window Style Recognition

Figure 16 shows a qualitative evaluation of our window generation stage. We generated the tile images as described in Section 6, and trained the window style recognition CNN using the 80% of the tile images. Then, the other 20% of the tile images were used for evaluating the accuracy of the trained CNN. The results indicate that our augmented training images allow the CNN to robustly recognize the window style without overfitting.

7.8. Variation

Since the output of our approach is a grammar, our system can easily generate a wide variety of 3D buildings by randomly changing the parameter values and grammar selections (Figure 17). Using only a few photographs, the user can generate many 3D building models with some variation without any knowledge of procedural

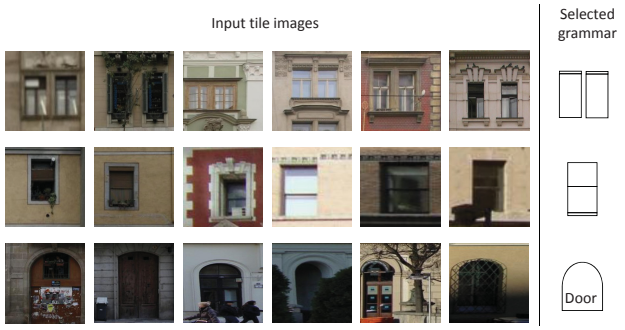

Figure 16: ***Window Style Recognition.*** *Given a tile image, our window style recognition CNN selects the best window grammar.*

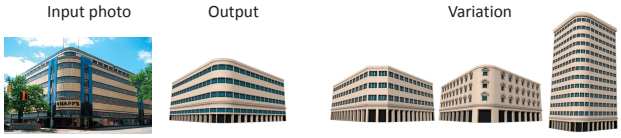

Figure 17: ***Variation.*** *Our system yields not only a 3D building grammar that is generated based on the input but also a wide variety of 3D geometry by randomly changing parameter values and grammar selections.*

modeling. As one of the applications, we created a virtual city by generating many buildings using the grammars that were obtained by our system (Figure 18). We also generated the other elements of the city, including roads and sidewalks, using procedural modeling.

7.9. Limitations

Figure 19 illustrates the main limitations of our approach. While the currently implemented building mass grammars support a wide variety of building shapes, some buildings may not be supported. In such a case, our approach finds the closest shape using the implemented grammars. Notice that the camera parameters are estimated nicely even for this case. With regard to façade generation, some façade images may not have any clear horizontal and verti-

Figure 18: ***Virtual City Corner.*** *Some of the generated 3D building grammars were used to generate many buildings in a virtual city. Other elements of the city including roads and sidewalks were also generated by procedural modeling.*

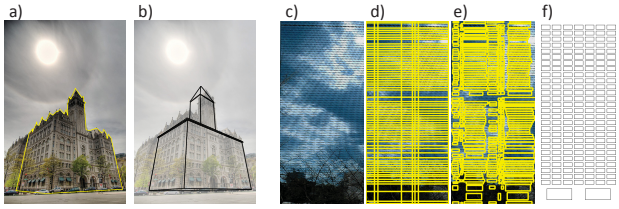

Figure 19: ***Failure Cases.*** *a) Our currently implemented grammars may not support some complicated building shapes. b) However, our CNN-based approach does find the closest shape and still estimates camera parameters well. Also, our façade generation approach does not work well if the façade does not have clear horizontal and vertical boundaries. From left to right, c) the rectified façade image, d) façade subdivision, e) detected windows, and f) generated façade structure are shown.*

cal edges. In this case, subdividing the façade into tiles does not work well, and the window detection fails. Nonetheless, our façade CNNs still select a grammar and estimate parameter values that produce a plausible façade structure.

8. Conclusions and Future Work

Procedural modeling is a popular approach to synthesize urban environments, but requires writing suitably parameterized grammars. We proposed an approach to significantly automate the generation of procedural buildings by taking a photograph as example input. Our system does not aim at an exact reproduction of a building, but rather at capturing its overall shape, the layout of its façade, and the style of its windows. To do so, we decompose the problem into logical stages (mass, façade, windows) and treat each stage with a common methodology that consists in simplifying the input to make it amenable to analysis by CNNs trained with synthetic data, and refining the output with custom optimizations. The resulting pipeline can generate a diversity of procedural buildings with no user intervention.

There are several interesting avenues for future work. First, our system relies on a number of recognition CNNs and parameter estimation CNNs, each trained separately. Since many of these CNNs perform a similar task, it is likely that they could share some of their weights, which would significantly reduce the memory consumption of our method. However, identifying which parts of the CNN could be shared among multiple tasks and which parts need to be specific to each task would require intensive experimentation. Second, while AlexNet produced satisfying parameter estimations for our problem (Table 1 and 2), there are newer architectures such as ResNet [HZRS15] for even easier training and higher accuracy. Third, we would like to explore automatically obtaining the silhouette of many buildings in a single photograph — this might enable generating in place multiple buildings from one image.

Acknowledgements

We would like to thank the authors of DeepFacade [LZZH17] for helping us generate façade parsing results, and the anonymous reviewers for their constructive suggestions. This research was partially funded by the ERC starting grant D^3 (ERC-2016-STG 714221), NSF CBET 1250232 and IIS 1302172, and by research and software donations from Adobe and Google.

References

[AME*14] AUBRY M., MATURANA D., EFROS A., RUSSELL B., SIVIC J.: Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *CVPR* (2014). 3

[ARB07] ALIAGA D. G., ROSEN P. A., BEKINS D. R.: Style grammars for interactive visualization of architecture. *TVCG 13*, 4 (2007), 786–797. 3

[BKC17] BADRINARAYANAN V., KENDALL A., CIPOLLA R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017). 4

[BWS10] BOKELOH M., WAND M., SEIDEL H.: A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29*, 4 (2010), 104:1–104:10. 3

[CDR99] CIPOLLA R., DRUMMOND T., ROBERTSON D.: Camera calibration from vanishing points in image of architectural scenes. In *Proceedings of the British Machine Vision Conference* (1999), pp. 38.1–38.10. 3, 6, 10

[CSP14] COHEN A., SCHWING A. G., POLLEFEYS M.: Efficient structured parsing of facades using dynamic programming. In *CVPR* (2014), pp. 3206–3213. 4, 7

[CZS*13] CHEN T., ZHU Z., SHAMIR A., HU S., COHEN-OR D.: 3sweepp: Extracting editable objects from a single photo. *ACM Trans. Graph. 32*, 6 (2013), 195:1–195:10. 3

[DAB16] DEMIR I., ALIAGA D. G., BENES B.: Proceduralization for editing 3d architectural models. In *3DV* (2016), pp. 194–202. 3

[DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), SIGGRAPH '96, pp. 11–20. 3

[FW16] FAN L., WONKA P.: A probabilistic model for exteriors of residential buildings. *ACM Trans. Graph. 35*, 5 (2016), 155:1–155:13. 4

[GMMB00] GUILLOU E., MENEVEAUX D., MAISEL E., BOUATOUCH K.: Using vanishing points for camera calibration and coarse 3d reconstruction from a single image. *The Visual Computer 16*, 7 (2000), 396–410. 3, 6, 10

[HEH05] HOIEM D., EFROS A. A., HEBERT M.: Automatic photo pop-up. *ACM Trans. Graph. 24*, 3 (2005), 577–584. 2

[HKYM16] HUANG H., KALOGERAKIS E., YUMER E., MECH R.: Shape synthesis from sketches via procedural models and convolutional networks. *TVCG PP*, 99 (2016), 1–1. 3, 5

[HVC17] HARA K., VEMULAPALLI R., CHELLAPPA R.: Designing deep convolutional neural networks for continuous object orientation estimation. *CoRR abs/1702.01499* (2017). 3, 4, 5

[HZ03] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, New York, NY, USA, 2003. 3, 6

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015). arXiv:1512.03385. 14

[JSD*14] JIA Y., SHELHAMER E., DONAHUE J., KARAYEV S., LONG J., GIRSHICK R., GUADARRAMA S., DARRELL T.: Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia* (2014), MM '14, ACM, pp. 675–678. 9

[JTC09] JIANG N., TAN P., CHEONG L.: Symmetric architecture modeling with a single image. *ACM Trans. Graph. 28*, 5 (2009), 113:1–113:8. 3

[KFWM17] KELLY T., FEMIANI J., WONKA P., MITRA N. J.: Bigsur: Large-scale structured urban reconstruction. *ACM Trans. Graph. 36*, 6 (2017), 204:1–204:16. 2

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1097–1105. 4, 9

[KST*09] KOUTSOURAKIS P., SIMON L., TEBOUL O., TZIRITAS G., PARAGIOS N.: Single view reconstruction using shape grammars for urban environments. In *ICCV* (2009), pp. 1795–1802. 4

[LN04] LEE S. C., NEVATIA R.: Extraction and integration of window in a 3d building model from ground view images. In *CVPR* (2004), vol. 2, pp. II–113–II–120 Vol.2. 8

[Low87] LOWE D. G.: Three-dimensional object recognition from single two-dimensional images. *Artif. Intell. 31*, 3 (1987), 355–395. 3, 7

[LSL15] LIU F., S. C., LIN G.: Deep convolutional neural fields for depth estimation from a single image. *CoRR abs/1411.6387* (2015). 2

[LZcZ14] LIU X., ZHAO Y., C. ZHU S.: Single-view 3d scene parsing by attributed grammar. In *CVPR* (2014), pp. 684–691. 2

[LZZH17] LIU H., ZHANG J., ZHU J., HOI S. C. H.: Deepfacade: A deep learning approach to facade parsing. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (2017), pp. 2301–2307. 4, 12, 13, 14

[MG13] MARTINOVIĆ A., GOOL L. V.: Bayesian grammar learning for inverse procedural modeling. In *CVPR* (2013), pp. 201–208. 4

[MMA16] MASSA F., MARLET R., AUBRY M.: Crafting a multi-task cnn for viewpoint estimation. In *British Machine Vision Conference (BMVC)* (2016). 3, 4

[MMVG16] MATHIAS M., MARTINOVIĆ A., VAN GOOL L.: Atlas: A three-layered approach to facade parsing. *International Journal of Computer Vision 118*, 1 (2016), 22–48. 7

[MMWVG12] MARTINOVIĆ A., MATHIAS M., WEISSENBERG J., VAN GOOL L.: A three-layered approach to facade parsing. In *ECCV* (2012), pp. 416–429. 7

[MRA15] MASSA F., RUSSELL B. C., AUBRY M.: Deep exemplar 2d-3d detection by adapting from real to rendered views. *CoRR abs/1512.02497* (2015). 5

[Mun06] MUNDY J. L.: Object recognition in the geometric era: A retrospective. In *Toward Category-Level Object Recognition* (2006), Springer Berlin Heidelberg, pp. 3–28. 3, 7

[MWA*13] MUSIALSKI P., WONKA P., ALIAGA D. G., WIMMER M., GOOL L., PURGATHOFER W.: A survey of urban reconstruction. *Comput. Graph. Forum 32*, 6 (2013), 146–177. 2

[MWW12] MUSIALSKI P., WIMMER M., WONKA P.: Interactive coherence-based façade modeling. *Comput. Graph. Forum 31*, 2pt3 (2012), 661–670. 4

[MZWVG07] MÜLLER P., ZENG G., WONKA P., VAN GOOL L.: Image-based procedural modeling of facades. *ACM Trans. Graph. 26*, 3 (2007). 4, 7, 8, 12

[NGDA*16] NISHIDA G., GARCIA-DORADO I., ALIAGA D. G., BENES B., BOUSSEAU A.: Interactive sketching of urban procedural models. *ACM Trans. Graph. 35*, 4 (2016), 130:1–130:11. 3, 5, 9, 10

[OCDD01] OH B. M., CHEN M., DORSEY J., DURAND F.: Image-based modeling and photo editing. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), pp. 433–442. 2, 3

[Pow09] POWELL M. J. D.: The bobyqa algorithm for bound constrained optimization without derivatives. Cambridge University Press. 7

[RDS*15] RUSSAKOVSKY O., DENG J., SU H., KRAUSE J., SATHEESH S., MA S., HUANG Z., KARPATHY A., KHOSLA A., BERNSTEIN M., BERG A. C., FEI-FEI L.: Imagenet large scale visual recognition challenge. *IJCV 115*, 3 (2015), 211–252. 2, 5, 9

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention 9351*, 3 (2015), 234–241. 4, 12, 13

[RMGH15] RITCHIE D., MILDENHALL B., GOODMAN N. D., HANRAHAN P.: Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. *ACM Trans. Graph. 34*, 4 (2015), 105:1–105:11. 2, 3

[SBM*10] STAVA O., BENES B., MECH R., ALIAGA D. G., KRISTOF P.: Inverse procedural modeling by automatic generation of l-systems. *Comput. Graph. Forum 29*, 2 (2010), 665–674. 3

[SCD*06] SEITZ S. M., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR* (2006), pp. 519–528. 2

[SLJ*15] SZEGEDY C., LIU W., JIA Y., SERMANET P., REED S., ANGUELOV D., ERHAN D., VANHOUCKE V., RABINOVICH A.: Going deeper with convolutions. In *CVPR* (2015), pp. 1–9. 4

[SQLG15] SU H., QI C. R., LI Y., GUIBAS L. J.: Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *ICCV* (2015). 3, 4, 5

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014). 4

[Teb11] TEBOUL O.: Ecole centrale paris facades database. URL: http://vision.mas.ecp.fr/Personnel/teboul/data.php. 5, 12

[TKS*11] TEBOUL O., KOKKINOS I., SIMON L., KOUTSOURAKIS P., PARAGIOS N.: Shape grammar parsing via reinforcement learning. In *CVPR* (2011), pp. 2273–2280. 4, 7, 12, 13

[TLL*11] TALTON J. O., LOU Y., LESSER S., DUKE J., MĚCH R., KOLTUN V.: Metropolis procedural modeling. *ACM Trans. Graph. 30*, 2 (2011), 11:1–11:14. 2, 3

[TSKP10] TEBOUL O., SIMON L., KOUTSOURAKIS P., PARAGIOS N.: Segmentation of building facades using procedural shape priors. In *CVPR* (2010), pp. 3105–3112. 4, 7

[Tv13] TYLEČEK R., ŠÁRA R.: Spatial pattern templates for recognition of objects with regular structure. In *GCPR* (2013), Springer Berlin Heidelberg, pp. 364–374. 5, 7

[VAB10] VANEGAS C. A., ALIAGA D. G., BENEŠ B.: Building reconstruction using manhattan-world grammars. In *CVPR* (2010), pp. 358–365. 4

[VAW*10] VANEGAS C. A., ALIAGA D. G., WONKA P., MÜLLER P., WADDELL P., WATSON B.: Modelling the appearance and behaviour of urban spaces. *Comput. Graph. Forum 29*, 1 (2010), 25–42. 3

[VGDA*12] VANEGAS C. A., GARCIA-DORADO I., ALIAGA D. G., BENES B., WADDELL P.: Inverse design of urban procedural models. *ACM Trans. Graph. 31*, 6 (2012), 168:1–168:11. 3

[WFP10] WU C., FRAHM J., POLLEFEYS M.: Detecting large repetitive structures with salient boundaries. In *ECCV* (2010), pp. 142–155. 12, 13

[WYD*14] WU F., YAN D., DONG W., ZHANG X., WONKA P.: Inverse procedural modeling of facade layouts. *ACM Trans. Graph. 33*, 4 (2014), 121:1–121:10. 4

[XHE*10] XIAO J., HAYS J., EHINGER K. A., OLIVA A., TORRALBA A.: Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR* (2010), pp. 3485–3492. 2, 5, 9

[ZCC*12] ZHENG Y., CHEN X., CHENG M., ZHOU K., HU S., MITRA N. J.: Interactive images: Cuboid proxies for smart image manipulation. *ACM Trans. Graph. 31*, 4 (2012), 99:1–99:11. 3

[ZSSS13] ZIA Z., STARK M., SCHIELE B., SCHINDLER K.: Detailed 3d representations for object recognition and modeling. *IEEE Transactions on Patterm Analysis and Machine Intelligence (PAMI) 35*, 11 (2013), 2608 – 2623. 3