Automatic Deep Inference of Procedural Cities from Global-scale Spatial Data

XIAOWEI ZHANG, ALY SHEHATA, BEDRICH BENES, and DANIEL ALIAGA,
Purdue University

Recent advances in big spatial data acquisition and deep learning allow novel algorithms that were not possible several years ago. We introduce a novel inverse procedural modeling algorithm for urban areas that addresses the problem of spatial data quality and uncertainty. Our method is fully automatic and produces a 3D approximation of an urban area given satellite imagery and global-scale data, including road network, population, and elevation data. By analyzing the values and the distribution of urban data, e.g., parcels, buildings, population, and elevation, we construct a procedural approximation of a city at a large-scale. Our approach has three main components: (1) procedural model generation to create parcel and building geometries, (2) parcel area estimation that trains neural networks to provide initial parcel sizes for a segmented satellite image of a city block, and (3) an optional optimization that can use partial knowledge of overall average building footprint area and building counts to improve results.

We demonstrate and evaluate our approach on cities around the globe with widely different structures and automatically yield procedural models with up to 91,000 buildings, and spanning up to 150 km^2. We obtain both a spatial arrangement of parcels and buildings similar to ground truth and a distribution of building sizes similar to ground truth, hence yielding a statistically similar synthetic urban space. We produce procedural models at multiple scales, and with less than 1% error in parcel and building areas in the best case as compared to ground truth and 5.8% error on average for tested cities.

CCS Concepts: • **Computing methodologies → Computer graphics**; **Image manipulation**; **Shape modeling**;

Additional Key Words and Phrases: Procedural modeling, uncertain spatial data, deep learning, urban modeling, content creation, satellite imagery, 3D urban modeling

ACM Reference format:
Xiaowei Zhang, Aly Shehata, Bedrich Benes, and Daniel Aliaga. 2020. Automatic Deep Inference of Procedural Cities from Global-scale Spatial Data. *ACM Trans. Spatial Algorithms Syst.* 7, 2, Article 6 (October 2020), 28 pages.
https://doi.org/10.1145/3423422

This research was funded in part by National Science Foundation Grant No. 1816514, *CHS: Small: Functional Proceduralization of 3D Geometric Models*, and Grant No. 1835739, *U-Cube: A Cyberinfrastructure for Unified and Ubiquitous Urban Canopy Parameterization.*

Authors' addresses: X. Zhang, A. Shehata, B. Benes, and D. Aliaga, Purdue University, 305 N University St, West Lafayette, IN 47907, USA; emails: {zhan2597, ashehat, bbenes}@purdue.edu, aliaga@cs.purdue.edu.

2374-0353/2020/10-ART6 $15.00
https://doi.org/10.1145/3423422

1 INTRODUCTION

Spatial urban models are of growing importance today for urban and environmental planning, geographic information systems, urban simulations, and as content for entertainment applications. An urban model typically consists of a network of road geometry defining a set of city blocks, buildings, and additional details, such as water bodies. However, creating models of realistic urban spaces is time-consuming and labor-intensive. Recent advances in big spatial-data acquisition and novel algorithms in deep learning have opened new opportunities for solving the problems in large-scale spatial data and for urban data in particular.

Amongst the multiple approaches to create such 3D urban models, procedural modeling and urban reconstruction are popular methodologies. Procedural modeling defines a set of rules and parameters to generate content (e.g., References [13, 36]) and it has been successful in application to urban spaces (e.g., References [32, 44, 49, 58]). However, rule creation is an iterative labor-intensive process and it is not easy to make a procedural model mimic a particular city or style as can be done with urban reconstruction. Various methods have addressed large-scale image-based and sensor-based reconstruction (see survey Reference [28]), but urban reconstruction requires a detailed acquisition throughout the entire urban space from many ground-level, aerial-level, or both vantage points. Further, such reconstructions must cope with data uncertainty in the form of resolution limitations, labeling errors, occluded structures, challenging automation, and other inaccuracies.

Our key inspiration is while satellite images cover a large space, they do not contain significant geometric detail of individual buildings. Nevertheless, we can observe statistical features of a city (e.g., mean and distribution of parcels and buildings both in terms of location and in terms of size). These features translate to a distinct appearance of the city at a large scale (e.g., city-specific combinations of locations of high-rise buildings and of smaller buildings). Thus, together with assumptions about urban structure, we can infer urban geometrical details. In our article, we propose a method that uses satellite imagery and global-scale population and elevation data as input to a deep-learning-based automatic method for producing a statistically similar and synthetic city-scale 3D urban model as output. The result is the ability to almost instantly create a plausible synthetic large-scale 3D urban model (Figure 1). Our method is aimed at content creation and urban planning—it is not suitable for creating an as-accurate-as-possible replica as needed, for example, in ground-level visual navigation.

Our automatic approach consists of three main components: (1) parcel area estimation, (2) procedural model generation, and (3) an optional procedural model optimization. Rather than limiting ourselves to a satellite-based photogrammetric reconstruction using the few pixels capturing the details of each building's walls and roof, our components infer a *procedural model* containing plausible details that are not present in the source imagery and yielding altogether a complete parameterized model. Our approach is the first to perform such an automatic inverse modeling from only satellite and global-scale data in just a few minutes. Further, our methodology is a starting point for modeling urban areas worldwide for urban design in city planning and simulation and for content generation in entertainment applications.

The output of our method is a large spatial procedural city model consisting of 3D buildings distributed over the target area and registered in place with the road network. We demonstrate our approach on various cities with widely different structure, in particular Chicago, Dublin, Hong Kong, Jacksonville, New Orleans, Paris, San Francisco, and Toulouse, automatically yielding procedural models with up to 91,000 buildings, and spanning up to 150 km^2. We performed both quantitative and qualitative comparisons. Overall, our results include 3D urban procedural models at multiple scales having less than one percent error in the best case. Our quantitative evaluation shows that

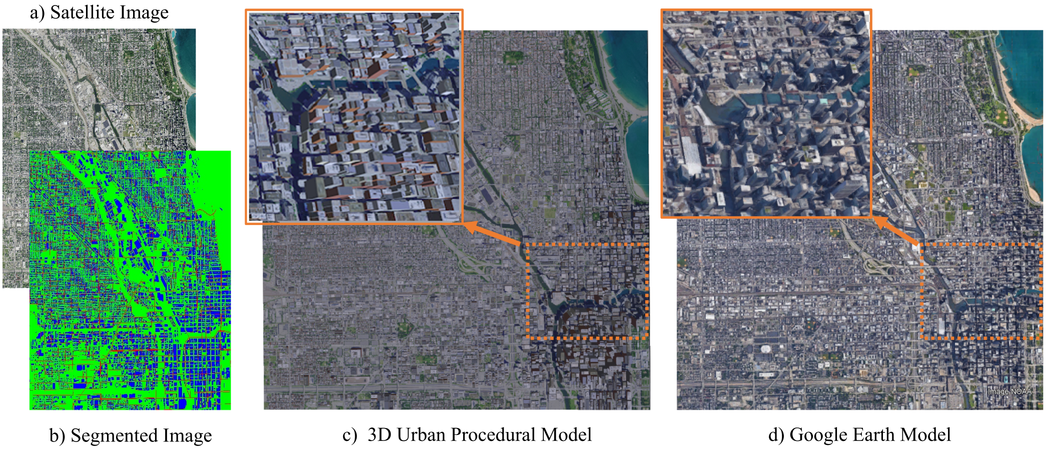


Fig. 1. Our method automatically generates a 3D urban procedural model (c) for the urban region from a segmented and labeled satellite image (a, b). We obtain, as compared to ground truth, a visually and statistically similar procedural model of a synthetic city that at a distance resembles results obtained by comprehensive reconstruction processes such as Google Earth (d).

we obtain, as compared to ground truth, a statistically similar city in terms of the mean building count and building area and their distribution. In addition, we show how well our method compensates for segmentation inaccuracies and occlusions yielding a better city model than directly using the segmentation data for constructing building models. Moreover, our optional optimization component further improves parcels, building outlines, and building geometries in the urban area (e.g., from 2.3× to 30× improvement). By means of a user study, we show a qualitative similarity as well as comparisons between our 3D output of different areas to the corresponding areas in Google Earth. In addition, we show preliminary results of using our approach for urban planning and modeling and for content generation.

Our main contributions include:

- An automatic approach to generate a 3D urban procedural model, based on a segmented and labeled satellite image, that is statistically and visually similar to the target urban area.
- A novel inverse modeling method to decompose a city into city blocks where for each city block we estimate procedural model parameters using classification and parameter estimation deep neural networks. This process is guided by an a priori analysis of typical parcels and building sizes and followed by an optional optimization stage to improve the similarity between the synthetic model and the segmented and labeled satellite image.
- A novel algorithm to complete uncertain data by generating parcels, building outlines, and building geometry. Our method uses the output of the aforementioned procedural model parameter estimation process and an analysis between population, elevation, and building size and spacing to determine building types and geometry.

2 RELATED WORK

The three main areas related to our work include (1) urban reconstruction and rendering, (2) (forward) procedural modeling, and (3) inverse procedural modeling.

Urban Reconstruction focuses on creating big urban spatial datasets from ground level, aerial, and satellite sensors. Most 3D urban reconstruction efforts use ground-level or aerial data. The survey by Musialski et al. [28] provides a comprehensive summary. Schoeps et al. [40] provide a recent benchmark of multi-view stereo results in general. Vanegas et al. [47] automatically created L-systems that reconstruct the building envelopes of 3D Manhattan buildings observed in a pair of aerial images. Shan et al. [41] produce urban reconstructions using aerial and ground-level images. Hou et al. [19] exploit planarity to obtain multi-view depth maps. Duan and Lafarge [12] focus on stereo reconstruction from calibrated wide-baseline satellite images. They obtained 2.5D reconstructions for building tops that can be observed and corresponded in both images. This is particularly challenging for smaller buildings and residential areas. Also, occlusion (e.g., by vegetation or neighboring buildings) might severely hinder reconstruction ability.

Another group of methods exploit LIDAR data as well as aerial imagery. For example, Yi et al. [59] and Poullis and You [35] produce models from LIDAR data. Bonczak et al. [7] uses both LIDAR and city administrative data to obtain an urban model. Park et al. [33] and Zhang et al. [61] propose learning-based point-cloud classification approaches, including the estimation of building height and mass. Zhou and Neumann [62] model rooftops and building walls using both LIDAR and aerial imagery. Some researches focus on tool building, such as Verdie et al. [52] that present methods to generate various level-of-details starting with a mesh or output of an urban reconstruction method. Another example is Agarwal et al. [1], which provides tools to improve efficiency and accuracy of interpolating LIDAR data. However, these methods require hard to obtain per-city LIDAR datasets and further, even if obtained, might lack information about significant parts of the city due to occlusion or lack of sampling.

In contrast to aerial or ground-level data, satellite provides much more coverage but with many other challenges. While great progress has been made, as seen in the work of Facciolo et al. [14] who won the 2016 IARPA Multiview Stereo 3D Mapping Challenge to produce 3D models from satellite imagery or Rupnik et al. [38], which improves upon the Facciolo et al work, there is a significant gap even with the latest satellite resolutions—moreover, because of distance and occlusion usually only the roof of urban structures is generally observed.

Even once a model is produced, significant challenges exist with regards to organizing the information for rendering. For example, Robles-Ortega et al. [37] focus on occlusion-culling to provide fast rendering for web-services of large urban models. Ole Vollmer et al. [53] present aggregation techniques to support on-demand level-of-detail generation. While these works produce impressive results they do not center on the creation of the urban model itself.

Procedural Modeling generates 3D models by using a set of rules, atomic elements, and parameter values. An example are L-systems that have been successfully applied to vegetation [36], noise-based methods used for textures and clouds [13], and procedural models in virtual worlds [44]. Urban procedural modeling can be dated back to the seminal work of Parish and Müller [32]. While this article focused on whole city models, later works have focused on particular elements such as buildings [26, 58], façades [5, 27, 42, 60], roads [9, 15], and layouts [24]. Cities are living structures and the complicated relationship of their habitants has been captured by various works, including Gwenola and Donikan [17], who studied animation in cities, and various works that attempted to simulate temporal changes of city geometry from underlying behavioral models [16, 34, 48, 57]. We refer the reader to surveys of urban procedural modeling (e.g., References [2, 49]). In general, the aforementioned methods require manual labor to design the procedural models.

Inverse Procedural Modeling attempts to find procedural representations of input models and it has been applied in various fields, such as vegetation [56], 2D road geometry [3], 2D vector structures [55], and 3D unstructured data [6]. One family of inverse procedural modeling approaches takes advantage of stochastic optimizations such as Metropolis optimization of L-system usage [46] and Markov Chain Monte Carlo optimization to find urban structures with desired properties [50]. Recently, Demir et al. [10, 11] used similarities in architecturals models to inversely generate procedural models. Nishida et al. [30] used deep learning to automatically infer

Table 1. Summary of Data Sources

Data Name	Data Source	Resolution	Scale
Road Vector	Open Street Maps [18]	–	Most cities
Elevation Data	JAXA [29]	30 m	Global
Population Data	LandScan [8]	1 km	Global
Satellite Data	–	1 m	Some cities

urban procedural models corresponding to user sketches. Kelly et al. [22] described a method to fuse street-level imagery, GIS footprints, and a coarse 3D mesh to produce 3D urban building mass and facade models. We highlight that the work of Reference [3] analyzes the distribution of road intersections and road geometry to create a procedural road geometry that is statistically similar to the source data. However, there is no treatment of parcels or buildings in their work. In general, while statistical approximations have been done, to our knowledge none pursued inverse building modeling at the scale of satellite imagery to produce large urban spatial datasets.

3 OVERVIEW

Data Sources. Our approach takes as input various geospatial products, summarized in Table 1. First, our method uses geo-registered segmented and labeled satellite images. In general, we assume the labels of *building* and *non-building*. If possible, then we can use *road* labels as well in the satellite images. At the global scale, building layout information or city-level GIS data is not available (i.e., it is available for certain large cities but not for all). Many satellite image segmentation methods exist, such as References [20, 31, 43, 54], as well as commercial solutions, such as eCognition [4]. These methods can, with a varying degree of accuracy, recognize and extract features from satellite images of anywhere in the world. However, these approaches do not produce 3D urban geometry, but rather produce a best-guess classification for each pixel in the image. For our method, we assume as input satellite images that have been segmented and labeled using an approach similar to the ones mentioned above. Second, we also use Open Street Maps (OSM) [18] to obtain roads, unless segmentation provides road labels. Third, our method also uses publicly available geo-registered global height data [29] and, fourth, our system exploits publicly available global population data [8], which we already have for the globe. These datasets are very coarse: for example, the height data is a sample every 30 m, with a vertical accuracy of about 5 m.

Processing. During automated processing, our approach (Figure 2) automatically produces a 3D urban procedural model. As a preprocess, the *parcel area estimation component* uses an analysis of the relationship between parcel sizes and building sizes in two large cities to train a classification network and a set of parcel area estimation networks that will be used for all our test cities. These deep neural networks are trained to take in a segmented and labeled satellite image of a city block and produce an estimated parcel area that will be used to help with building creation. We highlight that parcel boundaries are *not* directly visible in a satellite image. Hence, the use of this trained network, instead of directly using the segmented images, is to obtain the average parcel area despite the presence of noise, classification errors, and occlusion in the segmented imagery.

At runtime, the *procedural generation component* produces a model of the parcels and buildings for each city block. Individual city blocks are extracted from the segmented and labeled imagery. Then, by using estimated parcel sizes (by the aforementioned parcel estimation component) and the geo-registered global elevation and population datasets, building generation computes for each parcel the building type, building height, and setback values, and subsequently the 3D building

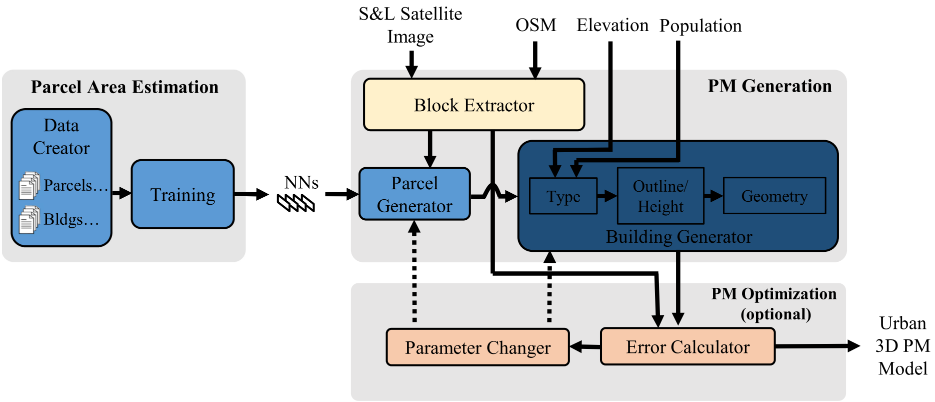


Fig. 2. Our approach consists of a procedural model generation component, a parcel area estimation component that is setup during offline processing, and an optional procedural model optimization component. The block extractor receives as input the segmented and labeled (S&L) satellite image, while the Building Generator receives the parcel output as well as information about the block's estimated elevation and population.

geometry. Ultimately, a model of the entire urban area is produced and desired urban morphology values can be calculated.

As a third (optional) step at runtime, the *procedural optimization component* improves the similarity between the synthetic city and the target city in the satellite images. This component iteratively compares synthetically generated average building footprint areas with the actual average building footprint areas for at least a fraction of the city. The optimizer alters a set of calibration parameters until convergence. Our experiments show that by knowing the average building footprint area of at least a small random fraction of the urban region (e.g., 5%) the synthetic model is on average 4x more similar to the actual city in terms of building area and building counts, resulting in a better 3D urban procedural model output.

Variables. Our subsequently defined approach makes use of the following variables (Table 2).

4 PARCEL AREA ESTIMATION

The goal of this component is to set up parcel area estimation function A, which estimates the average parcel area size for each parcel implied by a segmented and labeled satellite image of a city block (Figure 3). There is currently no existing global parcel data-set or a standard means of obtaining parcel information for all urban areas. Further, while parcel data for well-known mega-cities may be available through public city sources, even then it may not match the existing building layout. We therefore use this parcel area estimation component to approximate the parcel layout of a block, without relying on obtaining parcel data from the city. However, using satellite image segmentation and labeling into building, non-building, and road is a challenging task. Even the best methods result in some amount of mis-classification and occlusion. Also, frequently portions of segmented parcels and buildings are covered by vegetation or occluded by other structures. Moreover, the parcel boundaries are not actually visible nor labeled. Regardless, we seek to produce a procedural approximation of an urban area that has similar parcels and buildings. To accomplish A, we use deep neural networks that infer the parcel area based on the size and distribution of (noisy) labeled building pixels.

Table 2. Variables and Their Meanings

Variable	Meaning
B	a segmented and labeled satellite image
b_i	a city block image of B
A	the parcel area estimator
a_i	the average parcel area size for each parcel of a block b_i estimated by A
S	a synthetic image
s_i	a synthetic block image of S
P_G	the parcel generator
p_{ij}	a synthetic parcel j in a block s_i
q_{ij}	the number of parcels produced for s_i
B_G	the building generator
t_{ij}	the building type of p_{ij}
h_{ij}	the building height of p_{ij}
$S_{t,ij}$	the triple of setback ranges corresponding to of t_{ij}
m_{ij}	the building geometry model of p_{ij}

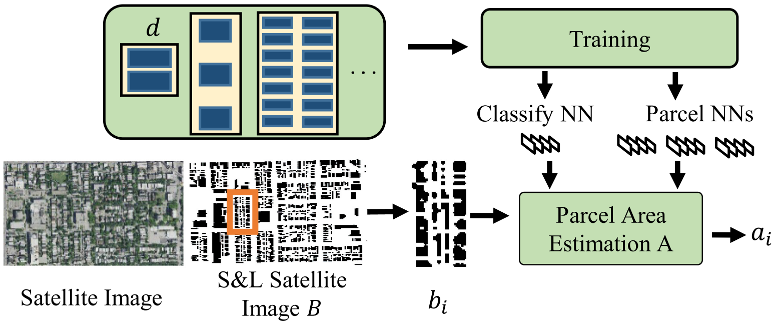


Fig. 3. Parcel Area Estimation. During preprocessing, synthetic parcel training dataset is created and used to train a parcel classification neural network (NN) and several parcel area estimation neural networks. At runtime, the parcel area estimation function A receives city block satellite images b_i and estimates the average parcel area a_i.

4.1 Canonical Representation

We use a canonical representation of city blocks, which facilitates data creation and training, because the neural network will classify and estimate parameters for a fixed-resolution labeled input image. Since city blocks in the satellite-image can be of many different shapes and sizes, we compute a tightly fitted oriented bounding box (OBB) for each city block. Then, we scale and rotate the longest axis of the OBB to fit within a predetermined image resolution (e.g., that used for our neural-network-based classification and parameter estimation) that allows arbitrarily shaped city blocks to be processed. After parcel area classification and estimation have been done, the synthetically created parcel is unrotated and unscaled.

Further, we make two simplifying assumptions about parcels: (i) we assume the parcels within a single city block are of about the same area; and (ii) we assume each parcel in the city block has zero or one building with an outline receded from the parcel boundary by three potentially different setback values: front, rear, and side. An analysis of Chicago and San Francisco tells us the single-building-per-parcel assumption is true for 90.65% and 93%, or 91.8% of the parcels on average. The front setback is used when the edge of the canonical parcel touches a road. The rear setback is opposite to the front setback and a side setback is used otherwise.

4.2 Training Data Creation

We use our canonical representation and an analysis of two real-world cities (i.e., Chicago and San Francisco) to determine the typical parcel-size range and relationship of parcel and building sizes to generate synthetic training data. Parcel and building outline data is obtained from OSM, and we compute histograms of city parcel areas and of front, side, and rear setback values (see Figures 22 and 23 in Appendix B). For each of the setback types, we compute the average value relative to parcel area. While optimized building setback ranges are used during model generation and optimization, training data is created using the aforementioned fixed relative setback values. To create the block images d for training, we randomly sample parcel area sizes within the observed parcel-size range and use the fixed triple of setback values to generate synthetic images.

4.3 Training

We seek to obtain a robust parcel area estimator such that $area(d) = A(d)$. Parcel areas vary significantly (see Figure 22), which makes it challenging for a single neural network to predict the parcel areas for any city block image. Instead, we perform a non-uniform quantization of the range of parcel sizes. We sort the analyzed parcel areas by size, divide the entire parcel area range into multiple bins, and train one network per bin. To define the bins, we find bin boundaries so that each bin has about the same number of parcels in the analysis cities. However, having multiple bins requires both a classifier neural network (to determine to which bin does a city block belong) and a parameter estimation neural network (to determine the actual parcel areas). As we show in Section 7, the accuracy resulting from using a different number of bins varies notably, and we choose the best answer (i.e., three bins) for all results.

We use convolutional neural networks (CNN) as the frameworks for our classifier and parcel area estimators. For the classifier CNN model, without loss of generality, we use the BVLC AlexNet architecture [23] except for changing the number of outputs of last fully connected layer to be the number of our estimation neural networks and initialize the network with a pre-trained network obtained from the Caffe Model Zoo [21]. Those initial weights are obtained by training with over a million training images to learn to discriminate features [39]. We fine-tune the network using 60,000 training images to achieve high accuracy after 10,000 iterations and mostly converging after 2,000 iterations. For each parcel area estimation CNN, we use a modified AlexNet architecture to fit our regression problem and use 120,000 images for training. We modified the number of outputs in the last fully connected layer of AlexNet, and also change the loss function to an Euclidean loss function. Training occurs over 100,000 iterations and is mostly converged after 40,000. We found this architecture to yield overall an average error of only 24 m^2 in estimating parcel areas (parcel areas range from about 20 to 20,000 m^2). Note that the same trained set of networks is used for all cities (and results) in this article.

5 PROCEDURAL GENERATION

Given a target area, this component successively generates a model of the parcels and buildings for each city block. Once all blocks are processed, a full 3D urban procedural model is output.

5.1 Urban Model

The urban procedural model is rendered in several layers. First, our method partitions a segmented and labeled satellite image B of an urban area into a set of city block images b_i. Given B as input,

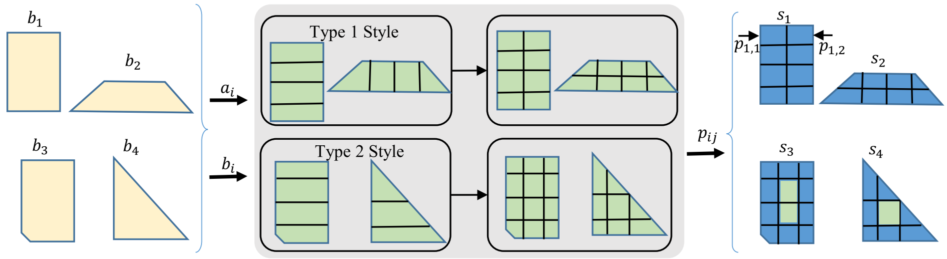


Fig. 4. Parcel Generation. Our method takes as input a city block satellite image b_i as well as the estimated parcel area a_i in said image and generates a parcel subdivision in one of two styles. Then, the parcels p_{ij} are output to form the synthetic blocks s_i.

this decomposition is achieved by converting the also geo-registered OSM road vectors into a graph G and then searching for all simple loops in G. Dead-end roads are ignored in our current implementation. Each detected loop geometry g_i segments a city block satellite image b_i out of B and is assumed to have a set of parcels, each with egress (i.e., an urban modeling and planning term that implies the parcel has street access by at least one parcel edge touching a surrounding road). We also render as a ground-plane the original (unsegmented) satellite image that contains the ground cover such as water bodies, grass, and dirt.

Then, we use parameterized rules to subdivide city blocks into one of two parcel styles (Section 5.2). After, we use parameterized rules to create buildings (Section 5.3). Moreover, we can optionally render additional hypothetical details such as window frames, trees, lampposts, and grass. See an example in Appendix Section C.

5.2 Parcel Generation

For a synthetic city block image s_i, our procedural parcel generator P_G generates parcels p_{ij} (i.e., synthetic parcel j in city block i). The set of s_i's form a synthetic image S. Given an extracted city block satellite image b_i, we estimate the average parcel area size a_i (e.g., in square meters) for each parcel in this block by using our neural network parcel area estimator A (i.e., $a_i = A(b_i)$). Afterwards, we use a recursive subdivision algorithm to generate synthetic parcels (Figure 4). Our parcel generation method is inspired by Reference [51] who proposed a generalized block subdivision method, following urban design guidelines, able to reproduce the parcel shapes and city blocks observed in many cities. Based on that work, we support two parcel subdivision types shown in Figure 16. Type 1 subdivision produces parcels whose front-side is along a street and rear-side is adjacent to another parcel of the same type, and type 2 subdivision creates city blocks that can have interior empty space (i.e., parcels with no buildings).

The parcel generator P_G subdivides the block geometry g_i by recursively splitting a tightly fit OBB of the current block partition until we obtain parcels no larger than a_i. If the generator enforces all parcels to have egress, then it results in the first of the aforementioned subdivision types. Otherwise, this approach produces the second of the aforementioned types, which has parcels in the interior of the city block. But, in all cases buildings are only placed in parcels having egress. Each recursive iteration uses either the longest or the shortest axis of the OBB to split the surrounding block geometry (e.g., initially g_i).

Altogether our procedural parcel generator performs the task

$$s_i = \{p_{i1}, p_{i2}, \dots p_{iq_i}\} = P_G(g_i, A(b_i)),$$

where q_i is the number of parcels produced for s_i.

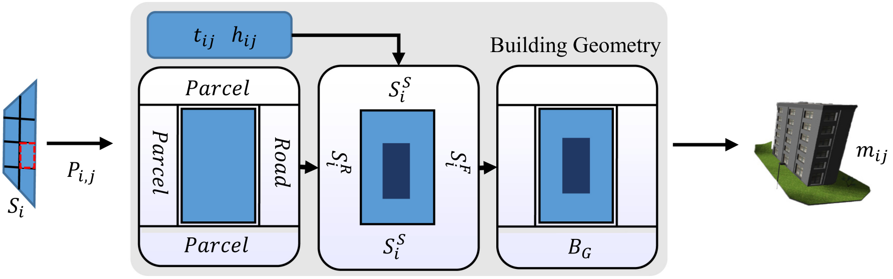


Fig. 5. Building Generation. Our approach receives a target parcel p_{ij}, computes the building type t_{ij} and building height h_{ij} from global elevation and population data, uses setbacks to define the building outline, and generates a building geometry m_{ij}. The setbacks S_i^F, S_i^S, S_i^R are sampled from per-building-type setback ranges.

5.3 Building Generation

Our building generator B_G defines one or none 3D building geometries for each parcel p_{ij} inside a synthetic city block s_i. To create building geometry model m_{ij}, the generator uses both global-scale population and global-scale elevation data to compute the building type within each parcel. It also uses a per-building-type triple of typical setback ranges (front, rear, and side setback ranges) (Figure 5). Then, the elevation data, building type, and setback ranges are used to create an appropriately sized procedural building model.

To estimate building height h_{ij} for parcel p_{ij}, our technique uses the elevation data E that represents the sum of terrain height and man-made structures. To compute elevation e_{ij} for building height estimation, we let $e_{ij} = E(p_{ij}) - E_{min}$ where the first term is the global elevation data for the given parcel and E_{min} is the minimum elevation within the vicinity of the urban area (i.e., the terrain height). Next, e_{ij} is rescaled to the range $[0, H_{max}]$. The value H_{max} is, by default, the maximum building height of a typical city, or is a single number provided as input for a target urban area. Thus, the final building height estimation is $h_{ij} = (E(p_{ij}) - E_{min})H_{max}$, which provides a good estimation when there are no significant terrain elevation changes throughout the input area.

Our method uses six procedural building types based on the Local Climate Zones (LCZ) classification by Reference [45]. LCZ defines a culturally independent global-scale classification for urban areas. LCZ is being increasingly adopted, because it captures the urban fabric in a much more precise way than prior land-use/land-cover classifications. In particular, it defines all urban areas to belong to one of ten possible zones (in total, LCZ defines 17 types including non-urban areas). The zones vary by the density of buildings as well as their heights. We use population data to estimate the density of a parcel. Thus, our six types are all combinations of low/mid/high rise with dense/sparse, i.e., (1) Low Rise Sparse, (2) Low Rise Dense, (3) Mid Rise Sparse, (4) Mid Rise Dense, (5) High Rise Sparse, and (6) High Rise Dense. Further, we define a typical front, rear, and side setback range for each building type; e.g., $[S^F_{min,k}, S^F_{max,k}]$, $[S^R_{min,k}, S^R_{max,k}]$, $[S^S_{min,k}, S^S_{max,k}]$ for $k \in [1, 6]$.

To determine the building type t_{ij} for a parcel, we split the building height range into three percentiles corresponding to: Low Rise, Mid Rise, and High Rise Buildings. Using the population data of a parcel $P(p_{ij})$, we further split each percentile into two percentiles to define sparse vs. dense areas. The result is the classification scheme for the listed six building types.

Finally, we compute an appropriate building geometry model m_{ij} based on the building's height and the setbacks associated with the building's type. Succinctly, our procedural building generator performs the task

$$m_{ij} = B_G(p_{ij}, h_{ij}, t_{ij}, S_{t,ij}),$$

where $S_{t,ij}$ is the triple of setback ranges corresponding to each of the defined building types t_{ij}.

6 PROCEDURAL MODEL OPTIMIZATION

The optional optimization component improves the similarity between a synthetic city and the target city in the satellite images. The component iteratively compares synthetically generated average building footprint areas with the actual average building footprint areas for at least a fraction of the city. The optimizer alters a set of calibration parameters until convergence.

6.1 Calibration Parameters

We improve similarity between the synthetic and the actual city by altering the parcel area ranges used during parcel area estimation and the per-building-type setback ranges. Parcel area estimation (Section 4) partitions the possible parcel areas into multiple bins; then, parcel area estimation returns a number between zero and one, which is mapped to the parcel bin range (e.g., $[20, 300]$ square meters for the first bin). If the number of buildings in s_i is different from that in b_i, then the city block deviates from the current assumed ratio between parcel size and building size. One way to improve is to generate smaller/larger parcels and thus alter the number of buildings. We accomplish this by shifting the bin boundary between adjacent parcel area bins. For example, given three parcel area bins we can shift the boundary between the first two and also the last two parcel bins. In general, for Z parcel area bins, this implies $Z - 1$ calibrated boundaries; i.e., α_u for $u \in [1, Z - 1]$. To alter the size of the buildings, we scale the setback ranges of all types by β to be larger or smaller to change building outlines and subsequently the 3D building geometry.

In addition, the parameters can be defined at a global scale (e.g., one set for the entire city) or at various local scales (e.g., different sets for different parts of the city). The former provides an intuitive global optimization while the latter enables improved local adaptability at the cost of more optimization parameters. To this end, we compute a quadtree subdivision of the urban area. The root node represents the top-level aggregation of the urban area and subsequent quad tree levels are progressively tighter aggregations. This flexibility enables us to tradeoff between global adaptation and more costly local adaptation. As is expected, our system performs better at larger levels of aggregation than at smaller ones—more details in the results section.

Hence, altogether, the calibration parameters are the set,

$$c_{mn} = \{\alpha_1, \alpha_2, \ldots \alpha_{Z-1}, \beta\},$$

where m represents the level in the octree (with c_0 being the set for the root node) and $n = \{1, 2, 3, 4\}$ is the quadtree node child index. A global optimization optimizes the set c_0. A local adaptation at the first level of a quadtree subdivision of the urban area optimizes c_{11}, c_{12}, c_{13}, and c_{14}, and so forth.

6.2 Optimization Loop

To perform an optimization, we select the urban area and the quadtree level at which to perform aggregation to compute calibration parameters. Then, we assume to have the overall average building footprint area and estimated building count for each of the quadtree areas (i.e., two scalar values for global optimization, eight scalars for optimizing at the first quadtree subdivision level). Our optimization engine uses Powell's method to minimize a weighted sum of the similarity between number of buildings and building outline area. In particular,

$$e_n = \frac{|count(s_{ij}) - count(b_{ij})|}{count(b_{ij})}, \tag{1}$$
$$e_a = \frac{|area(s_{ij}) - area(b_{ij})|}{area(b_{ij})},$$

where w_n is the weight for the number of buildings error e_n and w_a is the weight for the building area error e_a. The functions *count*() and *area*() compute the count and area of buildings, respectively, in the provided city block.

Thus, our overall optimization task is

$$argmin_{c_{mn}}(w_n e_n + w_a e_a), \tag{2}$$

for a desired quadtree aggregation level m (and for all valid values for n). Note that each time calibration parameters are changed, the synthetic city block s_i must be recomputed (i.e., starting with b_j and the current calibration parameters c_{mn}, new parcels p_{ij} and buildings m_{ij} are computed). The optimized parameters are then used during parcel area estimation of the entire urban area. The improvements yielded by this component are shown in the results section.

7 IMPLEMENTATION AND RESULTS

Our system is implemented in C++ while using several open source libraries including Boost, CGAL, Qt, OpenCV, and OpenGL. The majority of our results are generated on a desktop computer with Intel i7-8700 clocked at 3.20 Ghz, 32 GB of DDR4 RAM, and NVIDIA GeForce 1080. Our neural network training is performed on a Intel 2.4 GHz XEON computer with several NVIDIA GTX TITAN XP boards each with 12 GB of DDR4 RAM.

The overall performance time is divided into generation, offline training, and optimization time. For all our tests, procedural model generation takes about 1.5 to 2 min to automatically generate a 3D urban model from the satellite imagery spanning approximately 64 km^2. The training time for our parcel area estimation networks is about 3–5 h for the classifier network and 12–15 h for each parcel area bin network. The training only needs to be done once for all cities. Our optional procedural model optimization takes 1–2 h to do a global optimization for 64 km^2 and 5–12 min to do one local optimization covering 4 km^2.

Our test cities include Chicago, Dublin, Hong Kong, Jacksonville, New Orleans, Paris, San Francisco, and Toulouse. For each, we pick a representative 8 × 8 km^2 area. The number of buildings in each urban area varies from 10,415 buildings in Jacksonville dataset to 91,000 buildings in Dublin dataset. We obtain the height of the tallest building in each from the city website, and the value for H_{max} in the cities in the order listed is {442, 67, 484, 189, 212, 231, 326, 67} meters, respectively.

7.1 Visual Results

Figures 6, 20, and 21 qualitatively show our results. First, Figure 6 starts with a satellite image, its segmented and labeled version, and OSM road vectors, and then our method identifies each city block. Procedural model generation starts with the initial parcel area estimate provided by the neural networks and iteratively optimizes the solution by using global elevation, global population data, and a feedback loop. The final procedural model is output and can be visually compared to Google Earth.

Second, for our multiple test cities Figures 20 and 21 compares various views of our 3D urban model to corresponding views obtained from Google Earth. Notice the qualitative similarity. In addition, our accompanying video shows several virtual flyovers above the synthetic city models.

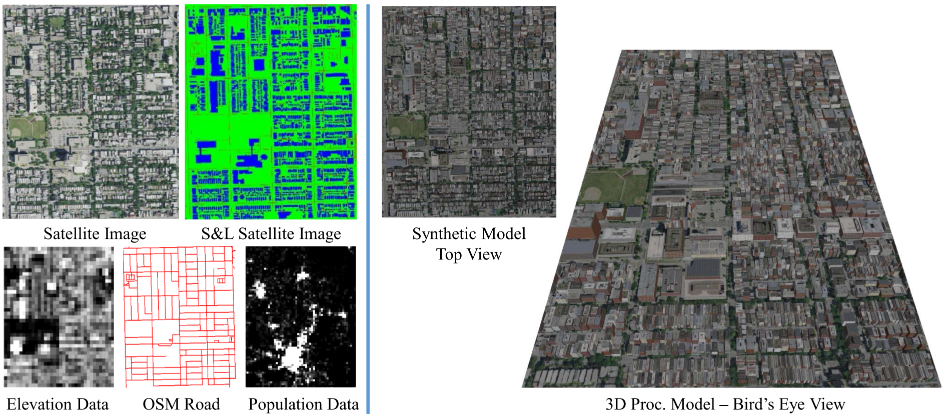


Fig. 6. Visual Pipeline. Our method uses a satellite image and its segmented version, together with OSM, elevation and population data, to create a 3D procedural model.

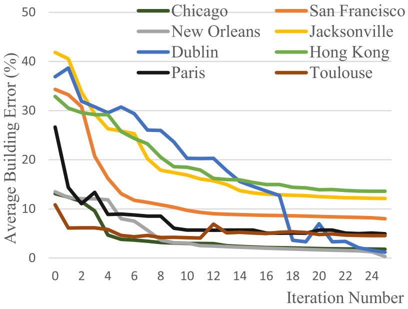


Fig. 7. Global Optimization. The progressive reduction of our error function during optimization of the calibration parameters c_0, resulting in improved parcels and buildings.

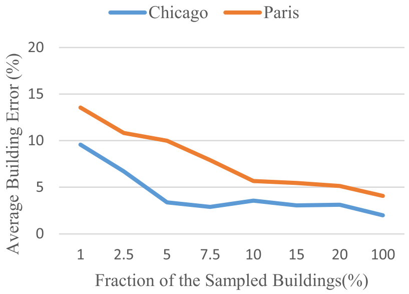


Fig. 8. Partial Knowledge of Average Building Footprint Area. The graph shows how knowing the overall average building area for a random subset of the buildings affects the final building error after optimization. For Paris and Chicago, between 5 and 10% is a good trade off point.

7.2 Numerical Results

Figures 7, 8, and 9 show numerically the improvement with various forms of our model optimization. Figure 7 shows our cities during optimization at the global aggregation level (i.e., optimizing calibration parameters c_o). The vertical axis is the average of relative building-area difference and relative building-count difference over the entire region (called *average building error*). The error reduction is also summarized in Table 3.

If the overall average building footprint area and building count is not available for the entire region, then it is sufficient to have the area/count for a small fraction of the targeted area. For example, Figure 8 explores the benefit of knowing the overall footprint area (and estimated count) of different percentages of the urban area; often just a few percent (e.g., 5–10%) is enough to obtain considerable benefit from this optimization component and almost identical to knowing the overall averages.

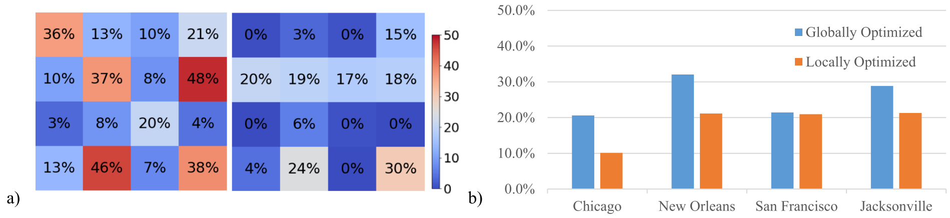


Fig. 9. Local Optimization. Using a subset of our cities, we show how a local optimization (e.g., calibration parameters c_{2k}) further reduces our error function. (a-left) Depicts the building error difference as a heatmap over the urban area (Chicago) using global optimization. (a-right) Depicts the corresponding building error differences but using local optimization. (b) A bar graph comparing building errors resulting from global vs. local optimization for a subset of cities.

Table 3. Global Optimization

City	**Initial Avg. Error**	**Optimized Avg. Error** (25 iterations)
Chicago	13.1	1.8
San Francisco	34.3	8.0
New Orleans	13.5	0.3
Jacksonville	41.9	12.1
Dublin	36.9	1.2
Hong Kong	32.9	13.61
Paris	26.7	4.9
Toulouse	10.9	4.6
AVERAGE	**26.3**	**5.8**

Figure 9 shows the error resulting from a local optimization (i.e., two levels down in the quadtree, so the 16 c_{2*} calibration parameters are computed). The color-coded heatmap on the left side of Figure 9(a) shows the average building error for each of the 16 tiles over Chicago resulting from using a global optimization. The heatmap on the right side of Figure 9(a) shows the corresponding building errors but after using a local optimization. In this case, the local optimization achieves a significant reduction overall with the total average error (i.e., the average of all 16 tiles) going from 20.12% to 9.75%. However, some tiles did in fact have an increase in error due to the local optimization converging to a wrong local minimum. In Figure 9(b), we see the total average error using locally optimized versus a globally optimized calibration parameters for several of our test cities. While local optimization requires some more calibration data, it allows our model to be more similar to the actual city layout not only from a distance but also from a closer perspective.

7.3 Statistical Comparisons

We compare the statistical distribution of generated building area and a spatially varying mean of generated building count and building area to ground truth. Figure 10 and Table 4 compare distributions of building area using *Kolmogorov-Smirnov Testing*. Table 5 performs a *t-test* to show similarity of values over space.

Figure 10 depicts the cumulative distribution functions (CDFs) for the synthetic models and for the ground truth of our cities—notice their similarity. The CDFs are computed from a histogram of the building areas. The number of histogram bins affects the granularity with which buildings of

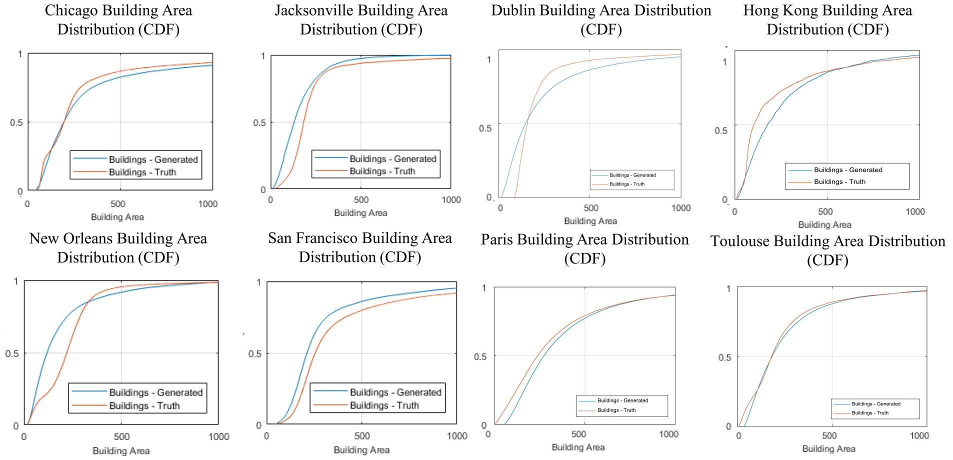


Fig. 10. Cumulative Distribution Function (CDF) Comparison. Observe the similarity between the CDF for the synthetic models and for the ground truth of our cities.

different sizes are considered equal. Thus, we seek to have CDFs considered similar yet produced from using as many bins as possible.

Table 4 quantitatively measures distribution similarity. For example, at significance level α = 0.05 (5%) the table indicates the two distributions are from the same underlying distribution. This means that it is extremely likely the distributions are similar if we use a building area granularity of 14 to 26 m^2 for building areas ranging up to 2,500 m^2.

Table 5 shows that for an adhoc subset of our urban regions the number of buildings and their sizes is not statistically different than ground truth at least when using the same 16 tiles as for local optimization. For each tile, we compute the mean number-of-buildings error and mean building-area error. Then, we use all 16 values in a *t-test* to determine if the mean of all errors is statistically different than zero with a significance level of $\alpha = 0.01$. The test indicates that the mean of all errors is not statistically different for the cities except for San Francisco building-area error (which would satisfy the test if we used a significance level of $\alpha = 0.005$ for example). In general, this test implies that the number of buildings and building areas over different regions of the city average out to be quite similar to ground truth.

7.4 Segmentation Comparison

We compare the results of our approach to directly extruding the segmented and labelled satellite image provided as input. One option is to rely on a satellite image segmentation that is very accurate and includes in-filling for occlusions. However, one of the benefits of our approach is the lack of this reliance. Table 6 compares our method (using global optimization for several iterations) to directly extruding the segmented satellite image (using the same initial elevation information we also take as input). This comparison shows how our method is able to better match ground truth. Table 6 shows the direct usage of segmentation results in building errors of >100% and 51% in Chicago and Toulouse, for example. In sharp contrast, our method has building errors of only 8.6% and 6.8% in the same respective cities. We also show corresponding qualitative comparisons for Chicago in Figure 11.

In Figure 12, we go into more detail to show how our trained network estimates parcel areas despite various levels of erroneous/noisy segmentation and labeling as well as parcel boundaries not being explicitly visible. For this evaluation, we progressively add a procedural noise to the

Table 4. Building Area Distribution Similarity Test

	$\alpha = 0.05$		$\alpha = 0.01$	
City	Area [m^2]	Dim [m]	Area [m^2]	Dim [m]
Chicago	14.2	3.8	11.1	3.3
Jacksonville	14.3	3.8	10.2	3.2
New Orleans	31.1	5.6	22.4	4.7
San Francisco	26.6	5.2	17.5	4.2
Dublin	26.3	4.1	19.1	4.3
Hong Kong	19.5	4.4	15.3	4.0
Paris	19.3	4.4	14.2	3.76
Toulouse	21.4	4.6	15.2	3.9
AVERAGE	**21.6**	**4.5**	**15.6**	**3.9**

We show that with significance level $\alpha = 0.05$ the distributions are similar if we use a building area granularity of at least 14 to 26m^2. At significance level $\alpha = 0.01$ the granularity reduces to 10 to 22m^2. Dim is the square root of area and represents the one-dimensional granularity.

Table 5. Building Number and Area Similarity Test

City	Num P-Value	Area P-Value
Chicago	0.85	0.06
Jacksonville	0.78	0.03
New Orleans	0.90	0.01
San Francisco	0.05	0.007
Dublin	0.76	N/A
Hong Kong	0.10	N/A
Paris	0.06	N/A
Toulouse	0.19	N/A
AVERAGE	**0.46**	**N/A**

We show that with significance level $\alpha = 0.01$ the number of buildings and building area errors over different regions of our cities are not statistically different than ground truth. *Note*: Only San Francisco building area does not pass the test at this significance level.

Table 6. Segmentation Comparison

Chicago	Count	Area	Count Err	Area Err	Error
Truth	49,461	452.8	–	–	–
Segm.	10,387	1667.8	79%	268%	174%
Ours	**41,765**	**460.2**	**16%**	**2%**	**8.6%**

Toulouse	Count	Area	Count Err	Area Err	Error
Truth	26,045	315.8	–	–	–
Segm.	12,392	469.1	52%	49%	51%
Ours	**22,954**	**320.9**	12%	2%	**6.8%**

We compare solutions for Chicago and Toulouse between directly using the segmented satellite image and our method. Our method shows considerably lower errors (8.6% and 6.8%).

full ground-truth segmentation shown in Figure 12(a). A noise factor of $f\%$ implies pixels near building boundaries have an $f\%$ chance of being mislabeled. Moreover, all pixels within a small radius of a to-be-mislabeled pixels are also mislabeled. We show noise levels of 10%, 20%, and 40%. After global optimization, the error function of average parcel and building area error is reduced to 0.9%, 1.2%, and 41%. As seen, up to 20% noise still yields acceptable performance.

7.5 User Studies

We also performed two user studies by using Amazon Mechanical Turk (AMT) to qualitatively evaluate our method. The user studies compare our method (using satellite imagery) to Google Earth (using their publicly available system that combines satellite imagery, aerial imagery, and semi-automatic reconstruction—which we effectively regard as ground truth). We also introduce a third method in each study serving a baseline and/or AMT user confidence estimator.

In the first study, we evaluate the *realism* of three methods at progressively farther viewing distances. We individually displayed images of portions of three cities (Chicago, Dublin, and New Orleans) from close to far viewing distances at oblique angles. We asked each of 320 AMT users a total of 36 questions. For each question, we displayed an image for five seconds and then the user was asked to provide a Yes/No answer to whether the image portrays a realistic urban area.

For each city, we generated images from 12 viewing distances spanning approximately 1–8 km at about a 45° downward looking viewing direction. At each distance, we created an image for

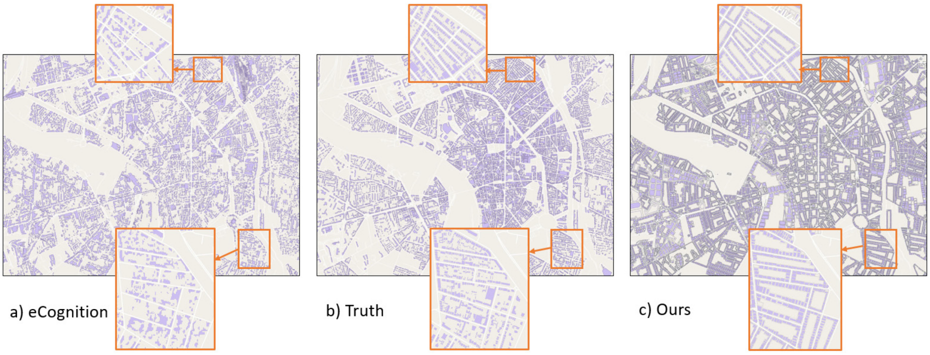


Fig. 11. Segmentation Comparison Images. We compare the results of our approach to directly extruding the segmented and labelled satellite image provided as input. (a) Segmentation and labeling result of using eCognition. (b) Ground-truth segmentation and labeling. (c) Our produced procedural model with labeling.

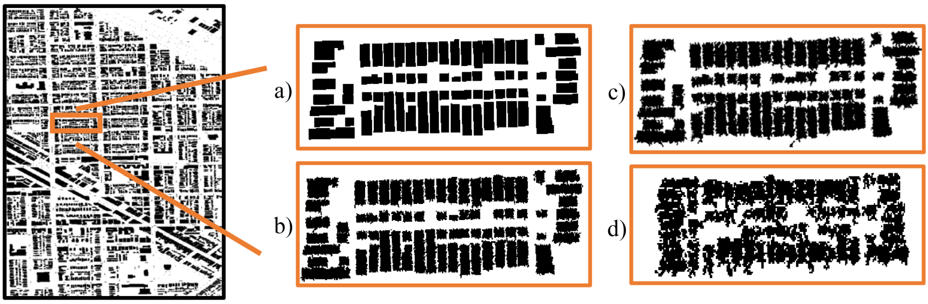


Fig. 12. Parcel Area Estimation Robustness. Our parcel area estimation function is robust to noise in the segmented and labeled satellite images. Panel (a) shows ground-truth segmentation. Panels (b, c, d) show 10%, 20%, and 40%, respectively, of added noise/misclassification. Regardless, our method was able to recover the areas quite accurately except at 40% error level.

each of three approaches: using Google Earth, using our method, and using a synthetic rendering with the same road structure and background texture as our method but with random building areas and heights. Since the building parcels (and footprints) are not known, the third method in this study generates effectively random buildings. At close range this produces a city model that is obviously not-realistic, and thus we use this fact to compute a confidence factor per AMT user. Although, we used AMT filters to only request work from high-quality users, we still need to disregard senseless responses. This same third method is also used to judge at which distance does it no longer matter what building sizes are used (and thus any method works fine). Further, to overcome potential bias introduced from the difference in rendering color styles from these these approaches, we used a global color remapping tool to make the color schemes similar.

Figures 13 and 14 show a summary of user study results. In Figure 14, the vertical axis shows the number of users that responded "yes" to the realistic image question. We use the responses of the realism question for the first three closest images of the random synthetic rendering to obtain a confidence value for each user (i.e., if the user rates the first few closest random synthetic rendering images as realistic, which they are very obviously not, then we give the worker a low confidence value). The horizontal axis is the eye viewing distance as reported by Google Earth; see Figure 13 for representative images at the different distances. Overall, from close-up users prefer Google

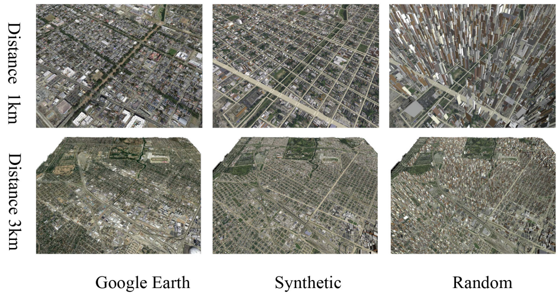


Fig. 13. Realism User Study. First row is viewing from 1km distance. Second row is a distance 3 km where our Synthetic image and Google Earth are considered almost equally realistic but the synthetic image with random building heights and areas is still notably less realistic.

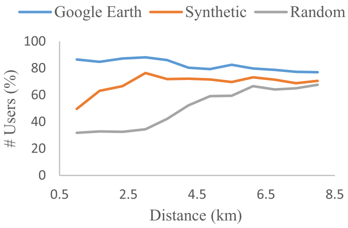


Fig. 14. Realism User Study Responses. Based on worker responses observing images of Chicago, Dublin, and New Orleans in random order, we show how realistic the different images are considered at increasingly farther distances.

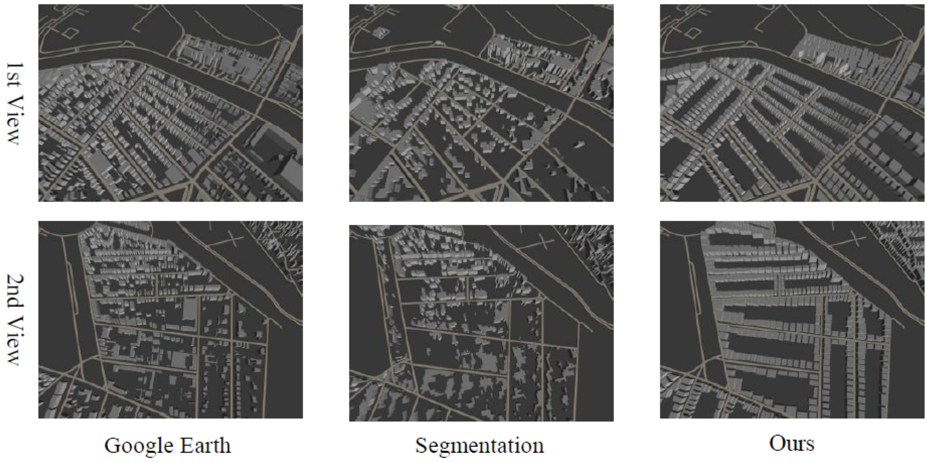


Fig. 15. Preference User Study Images. We show two views of Google Earth images, extruded building-segmentation images and our synthetic images.

Earth but from about 2.8 km users almost equally rate Google Earth and our synthetic rendering. When observed from sufficiently far away, all three image types are rated approximately equally.

In a second user study, we evaluate the *preference* of the results from three methods. In particular, we compare Google Earth, our method, and an extrusion of the buildings using segmented satellite images (i.e., segmented with eCognition and using the same building height information as our method). The study also used Mechanical Turk and it involved 100 users who were each asked to provide a preference for 12 pairwise image comparisons. We choose the same two close-up views as in Figure 11, created 6 image pairs from those two views, and 12 questions in total were created. Each image pair was shown and the users were asked which do they perceive as more realistic depiction of the urban area. The image pairs were shown in random order and in random left-right placement.

Table 7. Preference User Study Responses

Question	Google Earth	Ours	Segmentation
Q1	59	42	–
Q2	85	–	16
Q3	–	66	35
Q4	54	47	–
Q5	89	–	12
Q6	–	75	26
Q7	57	44	–
Q8	81	–	20
Q9	–	65	36
Q10	60	41	–
Q11	85	–	16
Q12	–	72	29
Total (#)	570	452	190
Total (%)	47%	37.3%	15.7%

We show responses to all 12 questions of our user study. In each, it is a pairwise comparison among Google Earth, our synthetic method, and extruded segmentation-based buildings.

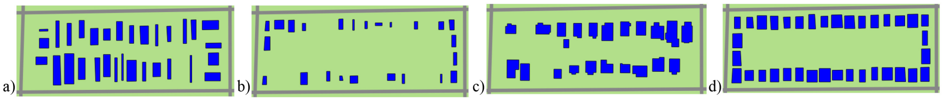


Fig. 16. Subdivision Styles. Our method supports two block subdivision styles. (a) Type 1 subdivision. (b) Type 2 subdivision. (c) The ground-truth buildings. (d) This synthetic solution uses type 2 and $w_n = 0.2$ and $w_a = 0.8$.

Figure 15 shows images from this user study and detailed responses of the user study are in Table 7. In summary, 84.3% of responses prefer either Google Earth or our method, over the extruded segmentation-based buildings. Further, a *t-test* reveals that at a significance level of $\alpha = 0.05$, there is no statistical difference between the preference of Google Earth or our method over the extruded segmentation-based method (i.e., p-value = 0.064).

7.6 City Block Subdivision

Figure 16 shows the interplay of procedural model generation and city block subdivision types. In general, our system can separately compute a 3D urban model using each of the two city block subdivision types and select the best fitting one, or the user can pre-select the type. However, the weights w_n and w_a for optimizing the number of buildings and building area, respectively, affect the fit. For example, Figure 16(a) (type 1) and Figure 16(b) (type 2) show a closeup result using weights of $w_n = w_a = 0.5$ for both types in Jacksonville. Figure 16(c) shows the ground truth, while Figure 16(d) use the weights of $w_n = 0.2$ and $w_a = 0.8$, respectively, for type 2, which yielded the overall smallest optimization error—for other cities, we use type 1. Nonetheless, in our results, we typically use $w_n = w_a = 0.5$ and leave to future work an automatic way to determine the best weights.

Table 8. Parcel Size Comparison

City	Parcel Size of Truth $[m^2]$	Parcel Size of Ours $[m^2]$	Relative Error
Chicago	618.81	600.62	2.94%
San Francisco	849.96	798.39	6.01%
New Orleans	585.70	615.61	5.11%

We show the average parcel-size performance by comparing our method with the ground truth for several cities.

Table 9. More Parcel Size Comparison

Chicago	Error
2 × 2	5.86%
4 × 4	8.14% (4.13%)
6 × 6	12.00%
8 × 8	17.20%
Blocks	45.90%

San Francisco	Error
2 × 2	7.04%
4 × 4	11.40% (5.09%)
6 × 6	17.30%
8 × 8	23.00%
Blocks	46.50%

New Orleans	Error
2 × 2	7.44%
4 × 4	12.20% (4.65%)
6 × 6	17.80%
8 × 8	22.30%
Blocks	49.20%

We show the average parcel size performance in different region levels ranging from splitting the whole city into 2 × 2 grid cells to individual blocks. Error is computed using the average of relative parcel-size errors per cell in a specific grid. Further, We place in parenthesis the error after applying a local optimization for a specific region level (i.e., 4 × 4).

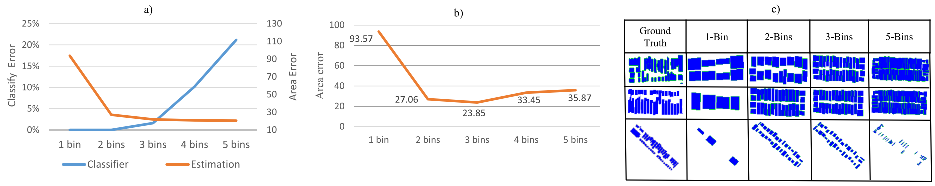


Fig. 17. Number of Bins. (a) We show tradeoff between classifier neural network (NN) error and parcel area NN error as you increase number of bins. (b) This graphs shows combined result of using both NN types. (c) Example parcel generations using parcel area estimation with a different number of bins. Overall three bins is clearly the best compromise.

7.7 Parcel Area Estimation

We evaluated the accuracy of obtaining the average parcel area size per city using our method by comparing to ground-truth data for Chicago, San Francisco, and New Orleans as shown in Table 8. Recall that parcel boundaries are not directly observable, so we use our trained parcel area estimation network to approximate parcel areas. To illustrate performance more comprehensively, we split the city into various region sizes as we did for local optimization of building areas (Section 7.2) and our user study (Section 7.5)—see Table 9. Although estimation error increases as region sizes get smaller, they are still reasonable at a subdivision level of 6 × 6. This shows that our approach is suitable for distant viewing such as in our user study (Section 7.5) and generates statistically similar cities. Further, as a quick experiment, we performed an optimization similar to that for local building area optimization (Section 6) and at the same level of subdivision (i.e., 4 × 4). This made parcel estimation error even smaller: the errors went down from 8.14% to 4.13% for Chicago, from 11.40% to 5.09% for San Francisco, and from 12.20% to 4.65% for New Orleans. Further analysis and improvements of parcel local optimizations are left as future work.

The effect of the number of parcel area bins used by Section 4 is shown in Figure 17. Using a test data set, Figure 17(a) shows how with an increasing number of bins the classifier error increases, as

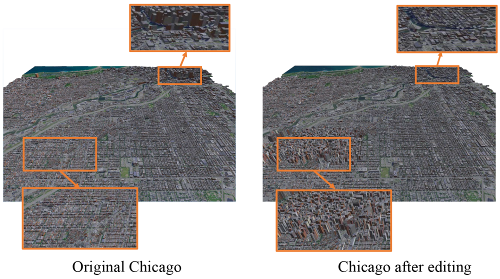


Fig. 18. Urban content generation. This figure shows how we can quickly generate and edit plausible city-scale models based on real-world cities.

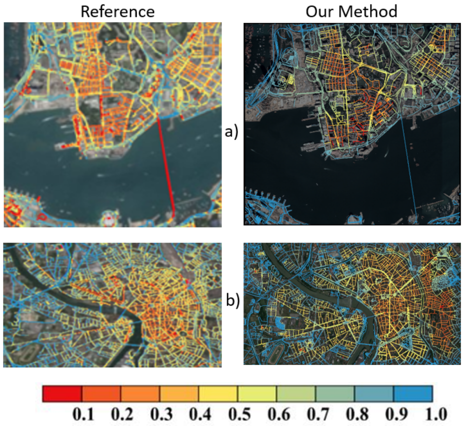


Fig. 19. Sky-view Factors. We show for Hong Kong (top) and Toulouse (bottom) the similarity of our automatically computed sky-view factor to that obtained by the lengthy semi-automatic method of using Reference [25] and Google Earth Street View images.

Table 10. Urban Morphology Distribution Test

	$\alpha = 0.05$	$\alpha = 0.01$
Parameter	Dim	Dim
A_h	6.8	4.12
A_r	0.18	0.04

Our ks-test shows that our two urban morphology values pass the test at significance levels $\alpha = 0.05$ and 0.01. For area weighted building height A_h the test passes with a granularity of at most 6.8 m for $\alpha = 0.05$ and 4.1 m for $\alpha = 0.01$. Similarly, for building surface to plan area ratio A_r, the test passes with a granularity of at most 0.18 for $\alpha = 0.05$ and 0.04 for $\alpha = 0.01$.

would be expected. However, the parcel area estimation error decreases with more bins—because each bin has a small range of areas to predict. Figure 17(b) shows the overall performance (both classification and parcel area networks are used). The performance seems clearly best for the three-bin setup—thus we use such a trained set of neural networks in all our examples. Figure 17(c) visually shows parcels generated using the initial parcel area estimates from setups with different numbers of bins. In general, the three-bin case seems to visually work better. Note that for the five-bin images, a misclassification occurred. If such a misclassification had not occurred, then the result might be similar to the three-bin case.

7.8 Application Examples

As applications, we show three tentative uses. Figure 18 demonstrates how a model of Chicago produced by our system can be edited to produce a different but detailed model of the same area. In this case, the user only need “paint” a new building height and population dataset layer; then our model is regenerated in 2 min. Table 10 demonstrates an example computation of urban morphology values useful for urban planning/urban climate studies. An urban planning collaborator from Hong Kong provided us with ground-truth area weighted building height (i.e., building height times its area) and building surface to plan area ratio (e.g., building surface area to parcel area). Our collaborator then used our synthetic model to compute the corresponding values. Using a similar ks-test as with results in Section 7.3, we show that our computed urban morphology values yield a distribution of values that is statistically similar to ground truth at either $\alpha = 0.05$ and 0.01. Figure 19 shows the sky-view factor computed for Hong Kong and Toulouse. The sky-view factor is the percentage of sky visible from, in this case, the roads (i.e., it factors in occlusions produced by the buildings) – it is an indicator often used in urban planning. The figure shows the similarity of our automatically computed result to the typical lengthy semi-automated solution, such as the method of using [25]. This method uses up to several million ground level images, from Google Earth Street View, to compute many sky-view factors per city. Our results show qualitative similarity and further analysis and improvements for sky-view computation are left as future work.

8 CONCLUSION

We presented an automatic novel inverse procedural method for urban modeling that is suitable for large distance views and generates statistically similar cities. The key idea behind our approach is a procedural model that is capable of generating parcels from city blocks and populating them with buildings. The parameters of the procedural model are initialized by neural networks that have been trained on real-world data and then improved via a procedural model optimization loop. Our method can generate a complete 3D model of a city only from an input satellite image and OSM using our global building height and population datasets.

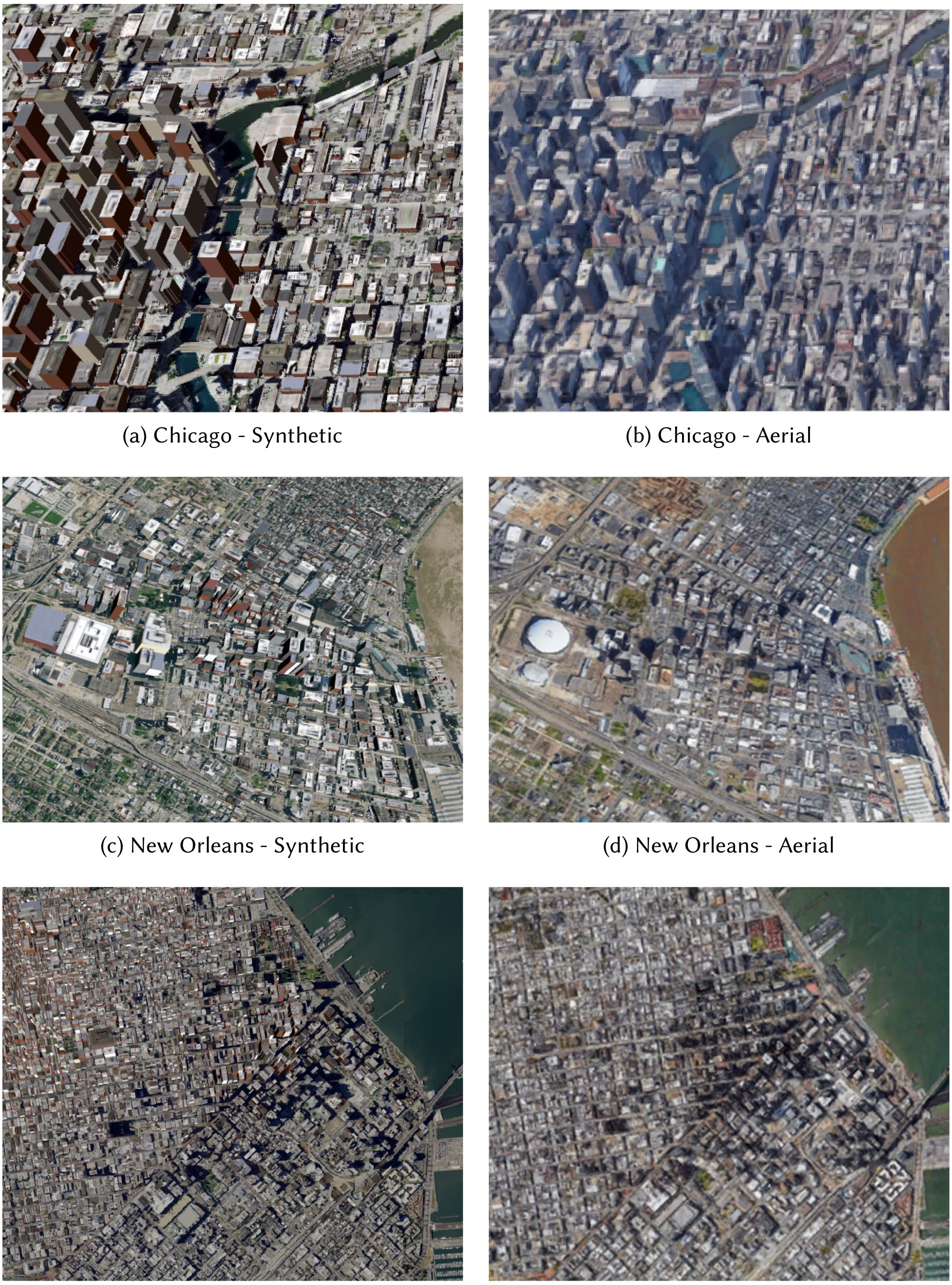
The main application of our approach is the quick automatic generation of large cities that are similar to real ones statistically and visually from a distance. Our approach is a starting point for modeling urban areas worldwide for content generation in entertainment and simulation (Section 7.8). In addition, many practical fields, such as urban planning, emergency responses, or weather modeling communities require a city's urban morphology. While some well-known cities might have dedicated teams able to create models, this tedious work does not scale to all cities. Our method enables automatically creating a 3D model suitable for such urban morphologies. Our method has already been chosen by a large international effort for precisely this purpose, including dozens of well-known researchers from around the globe.

Nonetheless, our method has various limitations. First, while we attempt to use as small a set of input data as possible, it is still difficult to get the data for all possible cities. The second limitation is the procedural model used. While we support two types of block subdivisions, there are more options (see Vanegas et al. [51]) and automatically determining which one to use is a challenge. Similarly, our method uses six predefined building styles that do not cover all possible buildings.

There are several avenues of future work. Our approach assumes the OSM road vector and the segmented satellite results were provided. One future work could be developing such neural networks to generate those data to reduce the amount of input data required by current system. In addition, the simplifications we use (e.g., only convex buildings, one building per parcel, etc.) may have significant visual effects. Further, our method does not support landmark buildings and it can be easily detected by the viewers. While our user study gives us some qualitative feedback, a more exact quantification is a task for future work. Another task is extending our building types to be geo-specific. We could either perform building typology studies (e.g., via crowdsourcing), or we can try to cluster and infer building types from global-scale data and imagery.

APPENDICES

A CITY-SCALE VIEWS



(a) Chicago - Synthetic

(b) Chicago - Aerial

(c) New Orleans - Synthetic

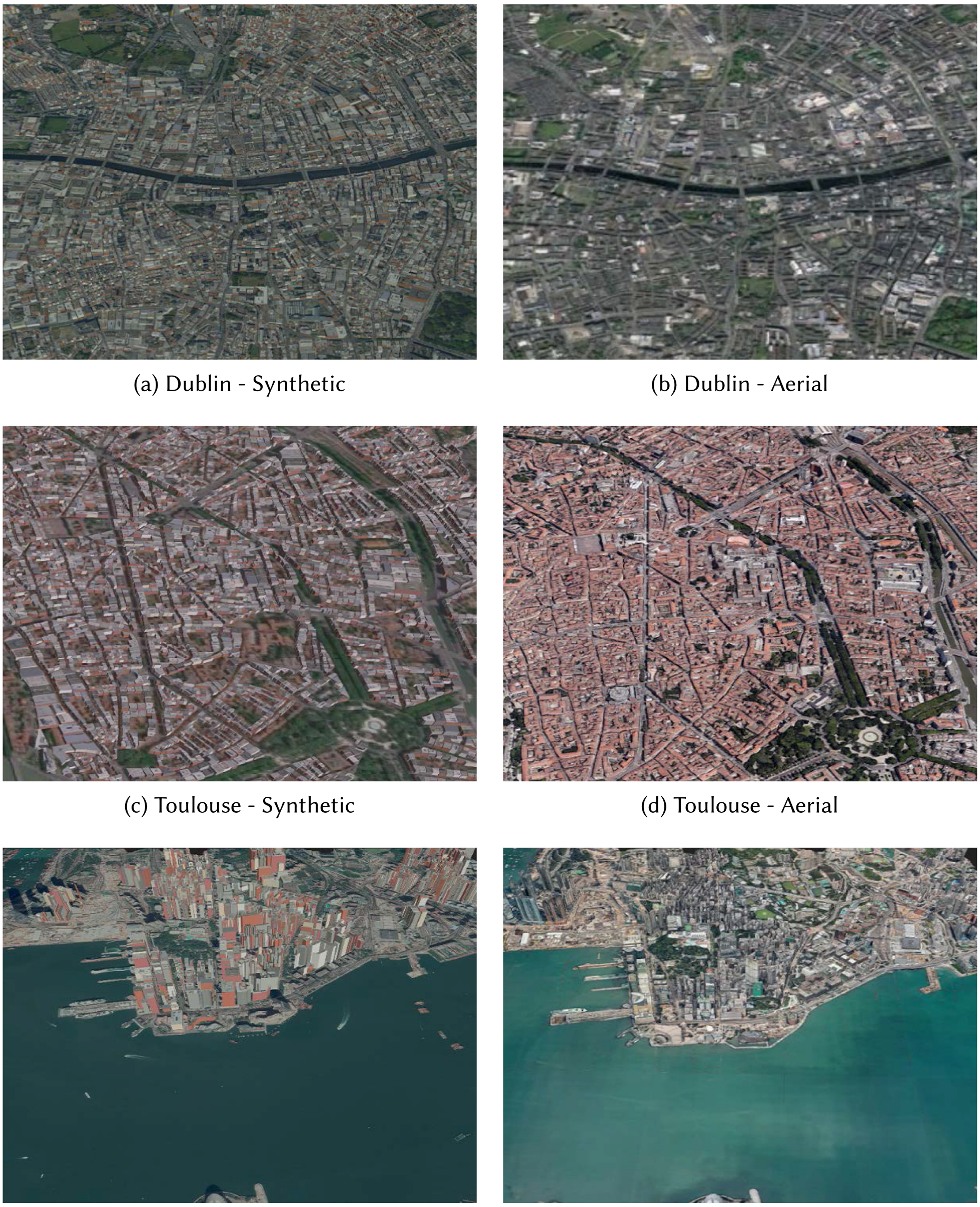
(d) New Orleans - Aerial

(e) San Francisco - Synthetic

(f) San Francisco - Aerial

Fig. 20. Example Models. We show several 3D urban procedural models and similar views using Google Earth output. We used a global color remapping tool to make the color schemes similar.

We show examples of Chicago, New Orleans and San Francisco in Figure 20, and additional examples of Dublin, Toulouse, and Hong Kong in Figure 21. The left column shows the synthetic models and the right column shows the corresponding Google Earth view.



(a) Dublin - Synthetic

(b) Dublin - Aerial

(c) Toulouse - Synthetic

(d) Toulouse - Aerial

(e) Hong Kong - Synthetic

(f) Hong Kong - Aerial

Fig. 21. Additional Example Models. Similar to previous figure, we show several 3D urban procedural models and similar views using Google Earth output.

B PARCEL SIZE AND SETBACKS

Our method uses an analysis of the real-world cities Chicago and San Francisco to succinctly generate synthetic training data. Figure 22 shows a histogram of analysis city parcel areas, and Figure 23 shows a histogram of analysis of setback values.

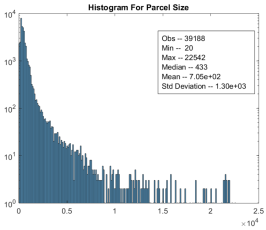


Fig. 22. Parcel-size Histogram. The histogram shows the real parcel sizes for our analysis cities—Chicago and San Francisco. We create our synthetic training data based on the distribution of the parcel-size histogram. We also divide our parcel estimation bins based on this distribution.

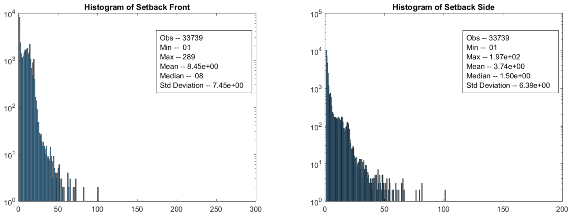


Fig. 23. Setbacks Histogram. The histogram shows the real setbacks for our analysis cities—Chicago and San Francisco. We use the average of these values during training and the various ranges per type as initial setbacks during generation.

C VISUAL ORNAMENTS

In Figure 24, we use our system to render additional hypothetical details such as roads, facades, window frames, trees, lampposts, and grass. The output is a realistic-similar 3D model.

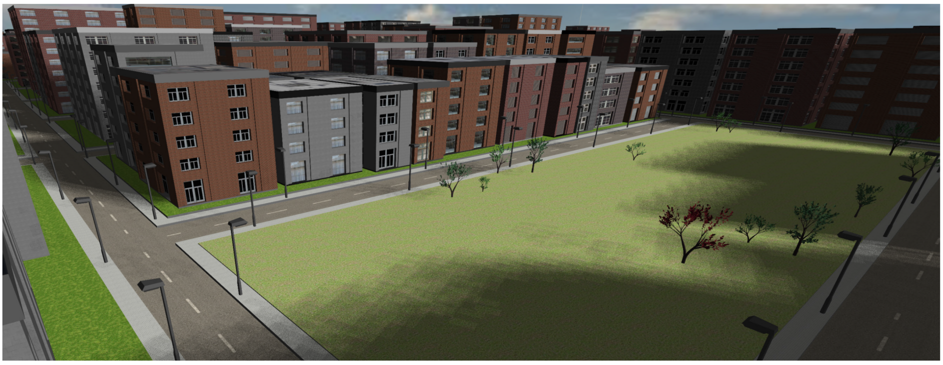


Fig. 24. Visual Ornaments. We can enable additional outputs to enhance our 3D urban procedural model.

REFERENCES

[1] Pankaj K. Agarwal, Alex Beutel, and Thomas Mølhave. 2016. TerraNNI: Natural neighbor interpolation on 2D and 3D grids using a GPU. *ACM Trans. Spatial Algor. Syst.* 2, 2, Article 7 (June 2016), 31 pages. DOI: https://doi.org/10.1145/2786757

[2] Daniel G. Aliaga, İlke Demir, Bedrich Benes, and Michael Wand. 2016. Inverse procedural modeling of 3D models for virtual worlds. In *Proceedings of the ACM SIGGRAPH Courses (SIGGRAPH'16).* ACM, New York, NY, Article 16, 316 pages. DOI: https://doi.org/10.1145/2897826.2927323

[3] Daniel G. Aliaga, Carlos A. Vanegas, and Bedrich Benes. 2008. Interactive example-based urban layout synthesis. *ACM Trans. Graph.* 27, 5 (2008), 1–10. DOI: https://doi.org/10.1145/1409060.1409113

[4] Martin Baatz, Ursula Benz, Seyed Dehghani, Markus Heynen, Astrid Höltje, PPeter Hofmann, Iris Lingenfelder, Matthias Mimler, Malte Sohlbach, Michaela Weber, and Gregor Willhauck. 2004. eCognition user guide. *Definiens Imaging,* GmbH, Munich, Germany.

[5] Fan Bao, Michael Schwarz, and Peter Wonka. 2013. Procedural facade variations from a single layout. *ACM Trans. Graph.* 32, 1, Article 8 (Feb. 2013), 13 pages. DOI: https://doi.org/10.1145/2421636.2421644

[6] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. 2010. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.* 29, 4, Article 104 (2010), 1–10. DOI: https://doi.org/10.1145/1778765.1778841

[7] Bartosz Bonczak and Constantine E. Kontokosta. 2019. Large-scale parameterization of 3D building morphology in complex urban landscapes using aerial LiDAR and city administrative data. *Comput. Environ. Urban Syst.* 73 (2019), 126–142. DOI: https://doi.org/10.1016/j.compenvurbsys.2018.09.004

[8] Eddie A. Bright, Amy N. Rose, and Marie L. Urban. 2012. LandScan 2012. Retrieved from https://landscan.ornl.gov/.

[9] Guoning Chen, Gregory Esch, Peter Wonka, Pascal, Müller, and Eugene Zhang. 2007. Interactive procedural street modeling. *ACM Trans. Graph.* 27, 3 (2007), 35. DOI: https://doi.org/10.1145/1278780.1278822

[10] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes. 2015. Coupled segmentation and similarity detection for architectural models. *ACM Trans. Graph.* 34, 4, Article 104 (July 2015), 11 pages. DOI: https://doi.org/10.1145/2766923

[11] Ilke Demir, Daniel G. Aliaga, and Bedrich Benes. 2016. Proceduralization for editing 3D architectural models. In *Proceedings of the International Conference on 3D Vision (3DV'16).* 194–202. DOI: https://doi.org/10.1109/3DV.2016.28

[12] Liuyun Duan and Florent Lafarge. 2016. Towards large-scale city reconstruction from satellites. In *Proceedings of the European Conference on Computer Vision (ECCV'16).*

[13] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. 2003. *Texturing and Modeling* (3rd ed.). Academic Press, Orlando, FL.

[14] Gabriele Facciolo, Carlo de Franchis, and Enric Meinhardt-Llopis. 2017. Automatic 3D reconstruction from multi-date satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'17).* 1542–1551. DOI: https://doi.org/10.1109/CVPRW.2017.198

[15] Eric Galin, Adrien Peytavie, Nicolas Maréchal, and Eric Guérin. 2010. Procedural generation of roads. *Comp. Graph. Forum* 29, 2 (2010), 429–438.

[16] Ignacio Garcia-Dorado, D. G. Aliaga, and S. V. Ukkusuri. 2014. Designing large-scale interactive traffic animations for urban modeling. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 411–420.

[17] Thomas Gwenola and Stephane Donikian. 2000. Modelling virtual cities dedicated to behavioural animation. *Comp. Graph. Forum* 19, 3 (Sep. 2000), 71–80.

[18] Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervas. Comput.* 7, 4 (2008), 12–18.

[19] Yaolin Hou, Jianwei Peng, Zhihua Hu, Pengjie Tao, and Jie Shan. 2018. Planarity constrained multi-view depth map reconstruction for urban scenes. *ISPRS J. Photogram. Remote Sens.* 139 (2018), 133–145. DOI: https://doi.org/10.1016/j.isprsjprs.2018.03.003

[20] Jing Huang and Suya You. 2016. Point cloud labeling using 3d convolutional neural network. In *Proceedings of the Interbational Conference on Pattern Recognition (ICPR'16).* IEEE, 2670–2675.

[21] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia (MM'14).* ACM, 675–678. DOI: https://doi.org/10.1145/2647868.2654889

[22] Tom Kelly, John Femiani, Peter Wonka, and Niloy J. Mitra. 2017. BigSUR: Large-scale structured urban reconstruction. *ACM Trans. Graph.* 36, 6, Article 204 (Nov. 2017), 16 pages. DOI: https://doi.org/10.1145/3130800.3130823

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS'12).* 1097–1105. Retrieved from http://dl.acm.org/citation.cfm?id=2999134.2999257.

[24] Marcus Lipp, Daniel Scherzer, Peter Wonka, and Michael Wimmer. 2011. Interactive modeling of city layouts using layers of procedural content. *Comp. Graph. Forum* 30, 2 (2011), 345–354. DOI: https://doi.org/10.1111/j.1467-8659.2011.01865.x

[25] Ariane Middel, Jonas Lukasczyk, Ross Maciejewski, Matthias Demuzere, and Matthias Roth. 2018. Sky view factor footprints for urban climate modeling. *Urban Climate* 25 (2018), 120–134. DOI: https://doi.org/10.1016/j.uclim.2018.05.004

[26] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc van Gool. 2006. Procedural modeling of buildings. *ACM Trans. Graph.* 25, 3 (July 2006), 614–623. DOI: https://doi.org/10.1145/1141911.1141931

[27] Pascal Müller, Gang Zeng, Peter Wonka, and Luc van Gool. 2007. Image-based procedural modeling of facades. *ACM Trans. Graph.* 26, 3, Article 85 (July 2007). DOI: https://doi.org/10.1145/1276377.1276484

[28] Przemyslaw Musialski, Peter Wonka, Daniel G. Aliaga, Michael Wimmer, Luc van Gool, and Werner Purgathofer. 2013. A survey of urban reconstruction. In *Comp. Graph. Forum*, Vol. 32. Wiley Online Library, 146–177.

[29] Konstantinos G. Nikolakopoulos. 2017. Evaluating ALOS AW3D30 data. In *Proceedings of the 5th International Conference on Remote Sensing and Geoinformation of the Environment (RSC'17)*, Vol. 10444. International Society for Optics and Photonics, 1044408.

[30] Gen Nishida, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive sketching of urban procedural models. *ACM Trans. Graph.* 35, 4, Article 130 (July 2016), 11 pages. DOI: https://doi.org/10.1145/2897824.2925951

[31] Sakrapee Paisitkriangkrai, Jamie Sherrah, Pranam Janney, and Anton Van-Den Hengel. 2015. Effective semantic pixel labelling with convolutional networks and Conditional Random Fields. In *Proceedings of the Computer Vision and Pattern Recognition Workshops (CVPRW'15)*, 36–43.

[32] Yoav I. H. Parish and Pascal Müller. 2001. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'01)*. ACM Press, 301–308. DOI: https://doi.org/10.1145/383259.383292

[33] Yujin Park and Jean-Michel Guldmann. 2019. Creating 3D city models with building footprints and LIDAR point cloud classification: A machine learning approach. *Comput. Environ. Urban Syst.* 75 (2019), 76–89. DOI: https://doi.org/10.1016/j.compenvurbsys.2019.01.004

[34] Chi-Han Peng, Yong-Liang Yang, Fan Bao, Daniel Fink, Dong-Ming Yan, Peter Wonka, and Niloy J Mitra. 2016. Computational network design from functional specifications. *ACM Trans. Graph.* 35, 4 (2016), 131.

[35] Charalambos Poullis and Suya You. 2009. Automatic reconstruction of cities from remote sensor data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE, 2775–2782.

[36] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. 2012. *The Algorithmic Beauty of Plants*. Springer Science & Business Media.

[37] Maria Dolores Robles-Ortega, Lidia M. Ortega, and Francisco R. Feito. 2017. Efficient visibility determination in urban scenes considering terrain information. *ACM Trans. Spatial Algor. Syst.* 3, 3, Article 10 (Nov. 2017), 24 pages. DOI: https://doi.org/10.1145/3152536

[38] Ewelina Rupnik, Marc Pierrot-Deseilligny, and Arthur Delorme. 2018. 3D reconstruction from multi-view VHR-satellite images in MicMac. *ISPRS J. Photogram. Remote Sens.* 139 (2018), 201–211. DOI: https://doi.org/10.1016/j.isprsjprs.2018.03.016

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2014. ImageNet large scale visual recognition challenge. Retrieved from http://arxiv.org/abs/1409.0575.

[40] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. 2017. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'17)*, Vol. 3.

[41] Qi Shan, Changchang Wu, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M. Seitz. 2014. Accurate geo-registration by ground-to-aerial image matching. In *Proceedings of the 2nd International Conference on 3D Vision (3DV'14)*. IEEE Computer Society, Washington, DC, 525–532. DOI: https://doi.org/10.1109/3DV.2014.69

[42] Chao-Hui Shen, Shi-Sheng Huang, Hongbo Fu, and Shi-Min Hu. 2011. Adaptive partitioning of urban facades. *ACM Trans. Graph.* 30, 6, Article 184 (Dec. 2011), 10 pages. DOI: https://doi.org/10.1145/2070781.2024218

[43] Jamie Sherrah. 2016. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. Retrieved from http://arxiv.org/abs/1606.02585.

[44] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A survey on procedural modelling for virtual worlds. *Comp. Graph. Forum* 33, 6 (2014), 31–50. DOI: https://doi.org/10.1111/cgf.12276

[45] Ian D. Stewart and Tim R. Oke. 2012. Local climate zones for urban temperature studies. *Bull. Amer. Meteorol. Soc.* 93, 12 (2012), 1879–1900.

[46] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. 2011. Metropolis procedural modeling. *ACM Trans. Graph.* 30, Article 11 (Apr. 2011), 14 pages. Issue 2. DOI: https://doi.org/10.1145/1944846.1944851

[47] Carlos A. Vanegas, Daniel G. Aliaga, and Bedrich Benes. 2010. Building reconstruction using manhattan-world grammars. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'10)*. 358–365. DOI: https://doi.org/10.1109/CVPR.2010.5540190

[48] Carlos A. Vanegas, Daniel G. Aliaga, Bedrich Benes, and Paul A. Waddell. 2009. Interactive design of urban spaces using geometrical and behavioral modeling. *ACM Trans. Graph.* 28, 5 (2009), 1–10. DOI: https://doi.org/10.1145/1618452.1618457

[49] Carlos A. Vanegas, Daniel G. Aliaga, Peter Wonka, Pascal Müller, Paul Waddell, and Benjamin Watson. 2010. Modelling the appearance and behaviour of urban spaces. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 25–42.

[50] Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. 2012. Inverse design of urban procedural models. *ACM Trans. Graph.* 31, 6, Article 168 (Nov. 2012), 11 pages. DOI: https://doi.org/10.1145/2366145.2366187

[51] Carlos A Vanegas, Tom Kelly, Basil Weber, Jan Halatsch, Daniel G. Aliaga, and Pascal Müller. 2012. Procedural generation of parcels in urban modeling. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 681–690.

[52] Yannick Verdie, Florent Lafarge, and Pierre Alliez. 2015. LOD generation for urban scenes. *ACM Trans. Graph.* 34, 3 (2015), 15. Retrieved from https://hal.inria.fr/hal-01113078.

[53] Jan Ole Vollmer, Matthias Trapp, Heidrun Schumann, and Jürgen Döllner. 2018. Hierarchical spatial aggregation for level-of-detail visualization of 3D thematic data. *ACM Trans. Spatial Algor. Syst.* 4, 3, Article 9 (Sep. 2018), 23 pages. DOI: https://doi.org/10.1145/3234506

[54] Michele Volpi and Devis Tuia. 2017. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* 55, 2 (Feb. 2017), 881–893.

[55] Ondrej Šťava, Bedrich Benes, Radomir Měch, Daniel G. Aliaga, and Peter Krištof. 2010. Inverse procedural modeling by automatic generation of L-systems. *Comp. Graph. Forum* 29, 2 (2010), 665–674. http://dx.doi.org/10.1111/j.1467-8659.2009.01636.x

[56] Ondřej Šťava, Soeren Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. 2014. Inverse procedural modelling of trees. *Comp. Graph. Forum* 33, 6 (2014), 118–131. DOI: https://doi.org/10.1111/cgf.12282

[57] Basil Weber, Pascal Mueller, Peter Wonka, and Markus Gross. 2009. Interactive geometric simulation of 4D cities. *Comp. Graph. Forum* (Apr. 2009). Retrieved from http://www.procedural.com/publications/2008_EG_Urban_Simulation/2008.EG.Weber.UrbanSimulation.Paper.pdf.

[58] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. 2003. Instant architecture. *ACM Trans. Graph.* 22, 3 (2003), 669–677. DOI: https://doi.org/10.1145/882262.882324

[59] Cheng Yi, Yuan Zhang, Qiaoyun Wu, Yabin Xu, Oussama Remil, Mingqiang Wei, and Jun Wang. 2017. Urban building reconstruction from raw LiDAR point data. *Comput.-Aided Design* 93 (2017), 1–14. DOI: https://doi.org/10.1016/j.cad.2017.07.005

[60] Hao Zhang, Kai Xu, Wei Jiang, Jinjie Lin, Daniel Cohen-Or, and Baoquan Chen. 2013. Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph.* 32, 4, Article 121 (July 2013), 13 pages. DOI: https://doi.org/10.1145/2461912.2461923

[61] Liqiang Zhang, Zhuqiang Li, Anjian Li, and Fangyu Liu. 2018. Large-scale urban point cloud labeling and reconstruction. *ISPRS J. Photogram. Remote Sens.* 138 (2018), 86–100. DOI: https://doi.org/10.1016/j.isprsjprs.2018.02.008

[62] Qian-Yi Zhou and Ulrich Neumann. 2012. 2.5 D building modeling by discovering global regularities. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'12)*. IEEE, 326–333.

Received April 2019; revised August 2020; accepted September 2020