USING STRUCTURAL REGULARITIES FOR A PROCEDURAL RECONSTRUCTION OF URBAN ENVIRONMENTS FROM SATELLITE IMAGERY

by

Xiaowei Zhang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Daniel Aliaga, Chair

Department of Computer Science

Dr. Bedrich Benes

Department of Computer Science

Dr. Christoph Hoffmann

Department of Computer Science

Dr. Clifton Bingham

Department of Computer Science

Approved by:

Dr. Kihong Park

ACKNOWLEDGMENTS

I would like to thank my wife, Shujun Zhang. She is a great woman and she is so supportive during my PhD study. She not only accomplishes her job in high standards, but also takes care of the whole family quite well. I would like to thank my children, and they are the source of my happiness. I would also like to thank my advisor, Prof. Daniel Aliaga for his continuous support and help throughout my PhD years. All my achievements would not have been possible without his instructions. His passion of science encouraged me to study hard to pursue my research interests. In addition, I would like to thank Prof. Bedrich Benes, who participated in most of the projects in this dissertation. He has generously taught me a wide variety of techniques and inspired me many interesting ideas, which helped me improve and broaden my knowledge. I would like to thank Prof. Chris Hoffmann for being part of my dissertation committee and teaching me the fundamentals of the computer graphics and computational geometry through a lot of course projects. I would like to thank Prof. Clifton Bingham for being a member of my dissertation committee and teaching me the machine learning techniques, which became one of the key parts of the dissertation. I would like to thank Dr. Gen Nishida for being a great mentor and a role-model researcher for my PhD life. He helped me a lot in my early years of research and I learned a lot from him both compute science knowledge and attitudes/methods of doing research. Moreover, I would like to thank my peers and friends in the CGVLab of Purdue for exchanging interesting ideas in the everyday conversation, Chris May, Meng-Lin Wu, Tharindu Mathew, Andres Bejarano, Daniel Andersen, Chengyuan Lin, Dana El-Rushaidat, Aly Shehata, Zixun Yu, Liu He and Yizhi Song.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Urban models are of growing importance today for urban and environmental planning, geographic information systems, urban simulations, and as content for entertainment applications. Various methods have addressed aerial or ground scale image-based and sensor-based reconstruction. However, few, if any, approaches have automatically produced urban models from satellite images due to difficulties of data noise, data sparsity, and data uncertainty. Our key observations are that many structures in urban areas exhibit regular properties, and a second or more satellite views for urban structures are usually available. Hence, we can overcome the aforementioned issues obtained from satellite imagery by synthesizing the underlying structure layout. In addition, recent advances in deep learning allow the development of novel algorithms that was not possible several years ago. We leverage relevant deep learning techniques for classifying/predicting urban structure parameters and modeling urban areas that address the problem of satellite data quality and uncertainty. In this dissertation, we present a machine learning-based procedural generation framework to automatically and quickly reconstruct urban areas by using regularities of urban structures (e.g., cities, buildings, facades, roofs, etc.) from satellite imagery, which can be applied to not only multiple resolutions ranging from low resolution (e.g., 3 meters) to high resolutions (e.g., WV3 0.3 meter) of satellite images but also the different scales (e.g., cities, blocks, parcels, buildings, facades) of urban environments. Our method is fully automatic and generates procedural structures in urban areas given satellite imagery. Experimental results show that our method outperforms previous state-of-the-art methods quantitatively and qualitatively for multiple datasets. Furthermore, by applying our framework to multiple urban structures, we demonstrate our approach can be generalized to various pattern types. We also have preliminary results applying this for flooding, archaeological sites, and more.

1. INTRODUCTION

Urban modeling are of growing importance today for urban planning, environmental science, geographic information systems, autonomous driving, animation and games, and urban sustainability. The blueprints and infrastructure of a city are under pressing needs to be designed for a strong rate of urbanization (e.g., two-thirds of the world's population will live in cities by 2050) and to survive crisis such as a pandemics, extreme weather, and natural disasters. 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. The increasing demand for high-quality and large-scale urban models capturing both detailed form and function has made manual urban content creation no longer a feasible option. Research in computer graphics and computer vision has responded to this need using urban (inverse) procedural modeling to quickly and (semi-)automatically synthesize parameterized virtual models of urban areas.

1.1 Procedural Modeling (PM)

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 [1], noise-based methods used for textures and clouds [2], and procedural models in virtual worlds [3]. Urban procedural modeling can be dated back to the seminal work of Parish and Muller [4]. While this paper focused on whole city models, later works have focused on particular elements such as roads [5], [6], parcel modeling [7], buildings [8], [9], facades [10]–[13], and layouts [14]. Cities are living structures and the complicated relationship of their habitants has been captured by various works including Gwenola and Donikan [15] who studied animation in cities and various works that attempted to simulate temporal changes of city geometry from underlying behavioral models [16]–[19]. We refer the reader to surveys of urban procedural modeling (e.g., [20] and [21]). In general, procedural programs generate high-quality and human-editable urban geometry when executed, but they require manual labor to design the procedural models.

1.2 Inverse Procedural Modeling (IPM)

Since it is difficult to define and cumbersome to write detailed procedural models of large areas, many works have focused on automatic creation of procedural models, or inverse procedural modeling, often starting from one or more 2D images or 3D objects. Inverse Procedural Modeling attempts to find procedural representations of input models and it has been applied in various fields, such as vegetation [22], 2D road geometry [23], 2D vector structures [24], tree creation [25], [26], and 3D unstructured data [27]. One family of inverse procedural modeling approaches takes advantage of stochastic optimizations such as Metropolis optimization of L-system usage [28] and Markov Chain Monte Carlo optimization to find urban structures with desired properties [29]. Recently, Demir et al. [30], [31] used similarities in architecturals models to inversely generate procedural models. Nishida et al. [32], [33] used deep learning to automatically infer urban procedural models corresponding to user sketches. Kelly et al. [34] 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 [23] 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.

1.3 Deep Learning

Recent advances in deep learning have also opened new opportunities for solving the problems of urban procedural modeling in particular.

Semantic segmentation is a classic topic in machine learning and computer vision. In recent years, with the amazing success of deep learning, many state-of-the-art segmentation networks [35]–[43] can be applied to urban structures. Specifically, a series of DeepLab [37], [39], [42] works maintains high-resolution by replacing strided convolution with atrous convolution. Encoder-decoder frameworks, like U-Net [36], infer high-resolution feature maps by joining the top-down and bottom-up pathways with lateral connections. GAN based frameworks, like Pix2Pix [40], consider segmentation as an image-to-image translation problem. However, those approaches most often focus on the general network structure and learning

methodology, and include many more content pixels than boundary pixels. This imbalance causes them to produce inaccurate structure edges and cannot ensure structural regularities and completeness (of man-made urban structures).

Another explosion of applications in deep learning is deep generative models [44]–[51] which can be applied to 3D modeling. Given enough training data, theoretically they can learn to generate plausible urban structures with broad variability. In particular, Kelly et al. [45] introduce a pipeline to automatically and realistically decorate building mass models by adding semantically consistent geometric details and textures. House-GAN [50] employs a generative adversarial network for floor-plan generation, while requiring room adjacency relations as input. Subsequently, Roof-GAN [51] presents a novel generative adversarial network that generates structured geometry of residential roof structures as a set of roof primitives and their relationships. However, these approaches are aiming to generate plausible urban structures with broad variability, and not focusing on accurate reconstruction. Their outputs often yield unrealistic outcomes and representations that are challenging to further edit, especially when considering intricate structural details and structural regularities of urban spaces. Besides, the aforementioned methods typically depend on a set of well-annotated datasets to train deep neural models.

1.4 Satellite Imagery

Urban procedural modeling has benefited from recent advances in computer graphics and deep learning. In particular, large-scale urban modeling is important for a variety of applications in urban content creation and in urban planning. Satellite imagery shows a lot of benefits because of its large coverage. We list several commonly used and publicly released satellite datasets across different regions in the world and at different spatial resolutions (30 cm and 50 cm) below.

SpaceNet: This dataset [52] contains building footprints in four cities (Las Vegas, Paris, Shanghai, and Khartoum) across the world. It contains the original panchromatic band, the 1.24m resolution 8-band multi-spectral 11-bit geotiff, and a 30cm resolution Pan-Sharpened 3-band and 8-band 16-bit geotiff. The labeled dataset consists of 24,586 tiles of size 200 m

× 200 m with a spatial resolution of 30 cm containing 302,701 building footprints across all areas, and are both urban and suburban in nature. Each tile comes with an 650 x 650 pixel RGB satellite image, a high-resolution panchromatic image, a low-resolution multi-spectral image, and ground truth building footprint annotation.

CrowdAI: The CrowdAI dataset was derived from the SpaceNet dataset. It provides a good dataset for comparing learning approaches on remote sensing data. Instead of considering all the channels in the multiband imagery from the SpaceNet dataset, it only focus on the RGB channels. The decision to exclude information from non-RGB channels helps create an alternate version of the SpaceNet dataset, which makes the problem easy and accessible to researchers in Deep Learning, who may or may not be very familiar with the tools used by the Remote Sensing community to manipulate the multiband imagery, and are usually more familiar with simple RGB images which are extensively utilized in Deep Learning research. Moreover, when considering only the RGB channels, the problem becomes a direct parallel of very popular instance segmentation tasks commonly studied in Deep Learning research. The CrowdAI dataset[53] contains 340,000 total tiles with 300 by 300 pixel RGB images at a 30 cm spatial resolution. Building footprint annotations are also provided.

Urban3D: Urban3D dataset contains 236 tiles of 2048 x 2048 pixel images and annotations with a spatial resolution of 50 cm. Each RGB tile in this dataset is accompanied by its Depth Surface Model and Digital Terrain Model (DSM and DTM), which provides high-resolution building height information.

1.5 Thesis Statement

Various methods have addressed aerial or ground scale image-based and sensor-based reconstruction. However, few, if any, approaches have automatically produced procedural urban models from satellite images. While the allure of modeling only from satellite images is clear, unfortunately structures obtained from the satellite images are often in low-resolution, noisy and heavily occluded, thus getting a clean and complete view of urban structures is difficult.

Despite having the highest-resolution commercially available satellite imagery (e.g., WorldView3), the main structure of a building occupies on average 90x90 pixels on the ground plane and on average the best observation of a facade is 20 pixels tall for a slant view of building objects. Although resolutions of satellite images from WorldView3 are relatively high, they are not free most of time, and users have to purchase from companies. Low-resolution satellite images (e.g., 1-3 meters per pixel) are more accessible to users but offers limited details. To illustrate the difference between various resolutions of satellite image data, look at these two satellite images of Venice in Figure 1.1. The left one is a 50cm high-resolution image from Pleiades satellite (Airbus); it allows to clearly see the buildings, small boats, narrow streets, and unfortunately, you have to pay for it. On the right is a 10m per pixel medium-resolution image from Sentinel-2 that provides a much coarser view, but – just like any open source satellite data – it's available free of charge.

(a) (b)

Figure 1.1. *Two Satellite Images of Venice.* (a) A 50cm high-resolution image from Pleiades satellite (Airbus). (b) A 10m per pixel medium-resolution image from Sentinel-2.

Aside from the relatively low resolution of satellite imagery, there are several other aspects that differentiate satellite-based multi-view stereo reconstruction from ground/aerial multi-view stereo reconstruction [54], [55]. First, satellites use scan-line sensors producing images with a different projection model than standard frame cameras. Usually a rational polynomial coefficient (RPC) model is used. Such RPCs are hard to calibrate, require iterative processes, need many ground control points, and performing 3D to 2D as well as 2D to 3D mapping is difficult [56]. Second, the image quality can vary a lot due to a number

of factors, including the viewing angles of satellite sensors are greatly limited by the orbit (i.e., not very off-nadir), images of an area might be days/weeks/months apart yielding different illumination and potentially physical changes, and radiometric quality is lower despite attempts of atmospheric corrections (see Figure 1.2).

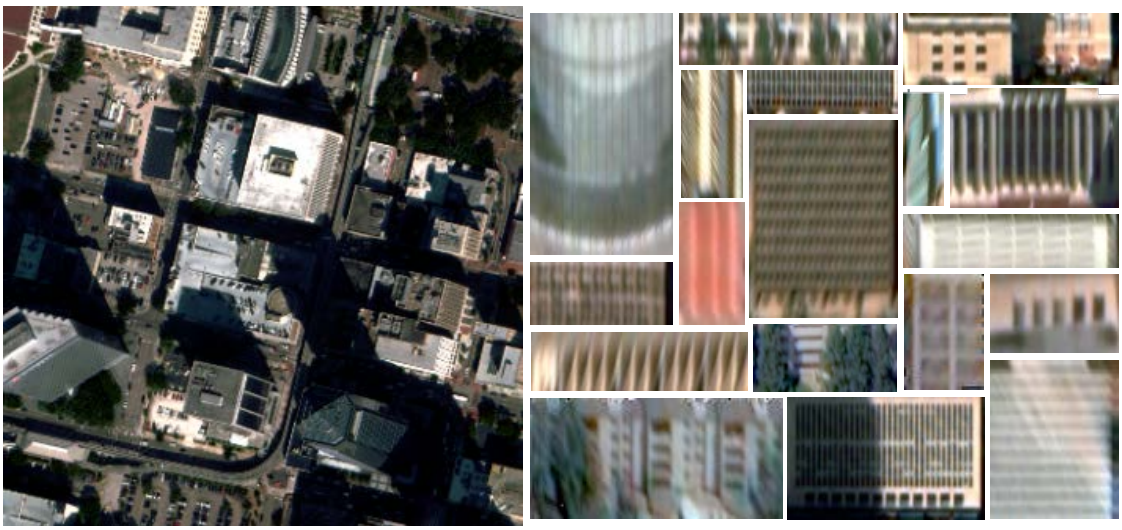


Figure 1.2. *Satellite Image and Facade Closeups.* Example satellite image and views of some typical facades.

Moreover, while there might be 1-20 satellite images observing portions of buildings, there is usually not a high quality satellite observation of every facade on a building due to shadows, foliage/occlusions, and limited resolution. Thus simply applying satellite images to building faces via projective texture mapping is inadequate. Further, such texture mapping depends on very accurate image-to-image registration, geometric modeling, and complete coverage of all building facades.

Procedural modeling methods exploit man-made patterns and their regularity (in our case of urban structures: walls are straight and parallel, corners have predetermined angles, etc.) in order to succinctly express the possible shapes. By having the ability to quickly and automatically model real world urban areas, and to easily edit them, procedural modeling enables many what-if scenario tools as well as flexible content creation.

To summarize, the thesis statement is:

In this dissertation, we present a machine learning-based procedural generation framework to automatically and quickly reconstruct urban areas by using regularities of urban structures (e.g., cities, buildings, facades, roofs, etc.) from satellite imagery, which can be applied to not only multiple resolutions ranging from low resolution (e.g., 3 meters) to high resolutions (e.g., WV3 0.3 meter) of satellite images but also the different scales (e.g., cities, blocks, parcels, buildings, facades) of urban environments.

1.6 Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 discusses our procedural reconstruction framework. Chapter 3 discusses the extension to the framework for generating procedural city-scale models. Chapter 4 discusses our system for modeling building-scale from satellite imagery. Chapter 5 talks about facade-scale reconstruction following our pipeline. Moreover, in Chapter 6, we show other applications of our system. Chapter 7 demonstrates results of different-scale modelings based on our framework, and shows evaluations quantitatively and qualitatively. Lastly, Chapter 8 provides conclusions and presents some ideas for future work.

2. URBAN PROCEDURAL RECONSTRUCTION

In the previous chapter, we first introduce urban procedural reconstruction and its related research areas in general. Then we highlight the benefits of satellite imagery in urban reconstruction and its challenges. Finally, we conclude with the thesis statement. In the following, we discuss details about critical observations and propose our framework for urban procedural reconstruction based on satellite imagery.

2.1 Structural Regularities

Figure 2.1. *Observation.* Man-made buildings exhibiting regular properties.

The first key observation is that (man-made structures) buildings exhibit "regular" properties (Figure 2.1) such as a division into one or more floors, parallel walls, walls meeting at one of a set of predetermined angles (e.g., 90 or 135 degrees), co-planarity between wall segments of different building stories and between adjacent/nonadjacent wall segments within the same floor, symmetrical arrangements, straight or curved walls, and other features. In

addition, urban facades typically exhibit a regular, grid-like facade structure. The bottom examples in Figure 2.1 show street-view facades images with windows, doors, and balconies that should be rectangular, horizontally and vertically aligned, spaced equally or with a clear pattern, and/or groups of similar size. Further, the regularities are also present in satellite facade images.

Secondly, while a single image does not contain significant z-values (i.e., 3D) information, we can exploit that the shape/type space of possible man-made structures in typical urban settings is constrained. For instance in building/roof example, we identify that the space of potential urban structure shapes can be explicitly enumerated, enabling $> 90\%$ coverage as observed in our preliminary experiments (see Chapter 4).

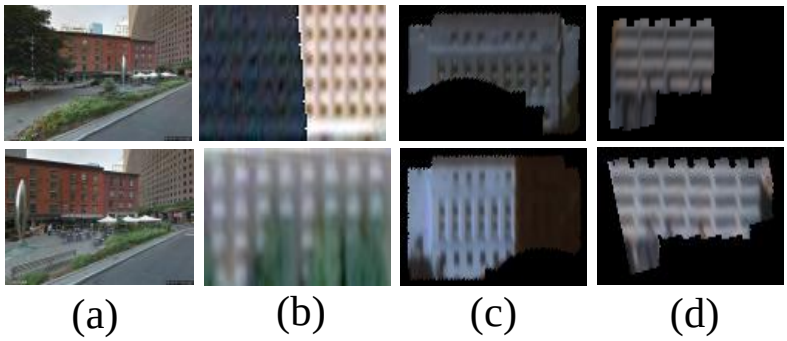


Figure 2.2. *Additional Views.* (a) Multi-views of partially occluded Google Street View images. (b) Satellite facades with occlusions/shadows. (c-d) Multi-views of satellite facade images.

Additionally, street-view and satellite-view images often have only partial observations of buildings/facades, and furthermore satellite images suffer more due to limitations in resolution, noise, complex camera models, limited viewing angles, and occlusions. Fortunately, a second or more views for both street-level and satellite are usually available as seen in Figure 2.2. Therefore, fusing or combining all views can be helpful when completing the whole urban structure coverage.

For low-resolution satellite images, the 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 shown in Figure 2.3). Moreover, the nearby urban structures most likely have the same constructions/shapes/patterns (e.g., low-rise suburb houses in one neighborhood usually have the same shapes, high-rise buildings are usually located in downtown areas, etc.).

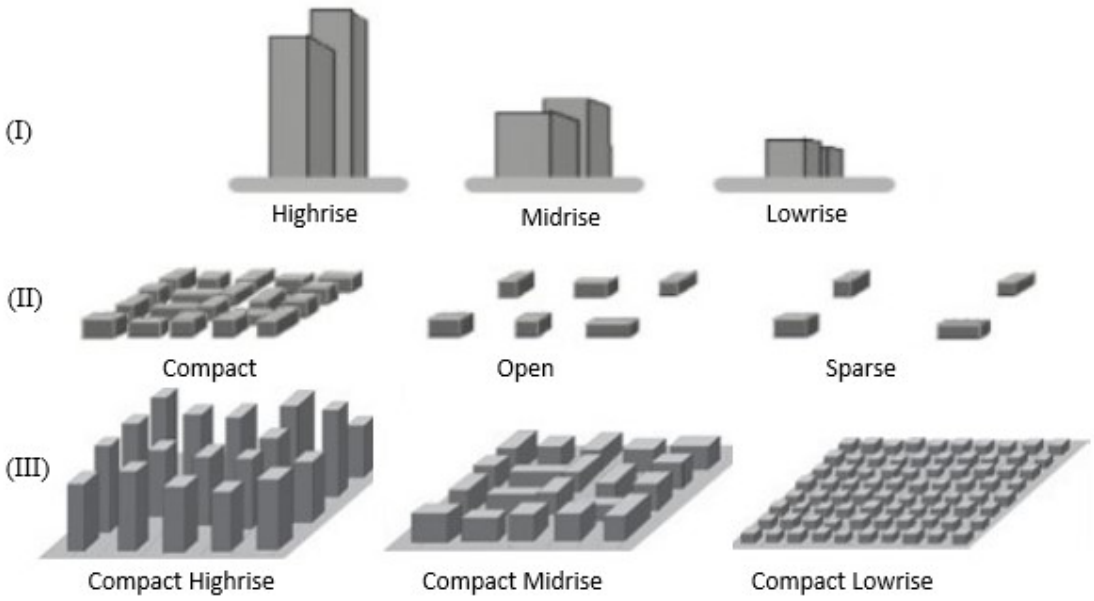


Figure 2.3. *Building Distribution.* Buildings in the urban area exhibit different statistical features.

Hence, together with the aforementioned observations of man-made structures, we can robustly infer urban geometrical details from satellite images, despite noise and occlusions. With the help of a deep learning framework, we demonstrate creating accurate procedural models of urban structures exceeding the ability of prior methods.

2.2 Framework Overview

We employ observations via a deep learning-based inverse procedural modeling approach to determine procedural parameters for a number of urban structure grammars in the presence of satellite data. This methodology significantly improves the resilience to occluded/noisy images and produces more accurate/regularized urban layouts as compared to al-

ternative direct segmentation-based methods. Since satellite images have a very limited off-nadir view (e.g., at most 20 to 40 degrees), and building surface coverage is limited (e.g., the orbital path of the satellite is not able to capture all building sides), often only fragments of a building are seen. Furthermore, urban structures that are observed may only be seen at very oblique angles, resulting in low resolution and stretched images. Nonetheless, a procedural approach has the ability to recreate the observed portion as well as create a plausible synthesized reconstruction of the occluded/not-sampled fragments. The result is plausible, regularized and complete urban structures.

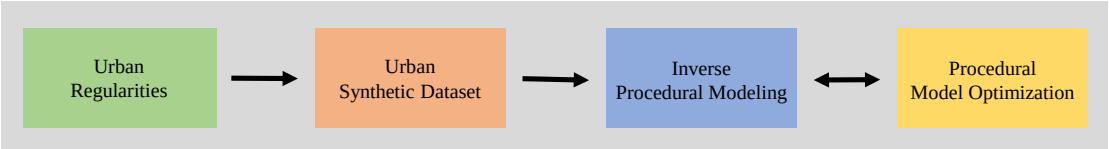


Figure 2.4. *Framework.*

To be specific, our framework has four main components: 1) find procedural grammars/representations for structures in urban areas based on urban regularities and observations, 2) create a synthetic dataset based on the grammars, 3) build customized urban inverse procedural modeling methods and train neural network models with synthetic data, 4) apply an optional procedural model optimization that leverage the partial knowledge of structures to improve results. We apply our system (see 2.4) to wide applications (e.g., city automatic reconstruction, urban building synthesis, roof synthesis, building façade synthesis, etc..) and demonstrate the ease of adoption of the architecture. Further, we compare our approach to several state-of-the-art methods and our approach consistently creates better results. In summary, our main contributions are as follows:

- we propose a framework to generate regularized urban structures in a parameterized procedural representation from satellite imagery, and
- we create synthetic training datasets which incorporates versatile regularities of urban structures and supports more general shapes/types, and

- we train our deep learning models with self-supervision to avoid time-consuming and expensive data annotations, and
- we perform comprehensive experiments demonstrating usage of our approach for different scales of urban structures and generalizations to other applications.

3. CITY SCALE

A portion of this chapter was previously published by ACM Transactions on Spatial Algorithms and Systems [57], https://doi.org/10.1145/3423422.

Low resolution satellite image data is more common and accessible to users. However, it suffers more noises and higher LOD (level of detail) reconstruction is not feasible. Nevertheless, we can observe statistical features of a city (see Chapter 2). Regarding this problem, we propose a method that is fully automatic and produces a 3D approximation of an urban city area given satellite imagery and other global-scale data including road network, population, and coarse elevation data. By analyzing the values and the distribution of urban data, i.e., distances between parcels, buildings, setbacks, 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 a set of neural networks to provide initial parcel sizes for a segmented satellite image of a city block, and 3) an optional procedural model optimization that can use partial knowledge of overall average building footprint area and building counts to improve results. 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 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.

3.1 Related Work

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. [58] provides a comprehensive summary. Schoeps et al. [59] provide a recent benchmark of multi-view stereo results in general. Vanegas et al. [60] automatically created L-systems that reconstruct the building envelopes of 3D Manhattan buildings observed in a pair of aerial images. Shan et al. [61] produce urban reconstructions using aerial and ground-level images. Hou et al. [62] exploit planarity to obtain multi-view depth maps. Duan and Lafarge [63] 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. [64] and Poullis and You [65] produce models from LIDAR data. Bonczak et al. [66] uses both LIDAR and city administrative data to obtain an urban model. Park et al. [67] and Zhang et al. [68] propose learning-based point-cloud classification approaches including

the estimation of building height and mass. Zhou and Neumann [69] model rooftops and building walls using both LIDAR and aerial imagery. Some researches focus on tool building, such as Verdie et al. [70] 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. [71] that 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. [72] who won the 2016 IARPA Multiview Stereo 3D Mapping Challenge to produce 3D models from satellite imagery or Rupnik et al. [73] 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. [74] focus on occlusion-culling in order to provide fast rendering for web-services of large urban models. Ole Vollmer et al. [75] 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.

Except for the urban reconstruction techniques, our work is also closely related to (inverse) procedural modeling and deep learning and we refer you to Chapter 1 for details.

3.2 Pipeline Overview

Data Sources

Our approach takes as input various geospatial products, summarized in Table 3.1. First, our method uses geo-registered segmented and labeled satellite images. In general, we assume the labels of *building* and *non-building*. If possible 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

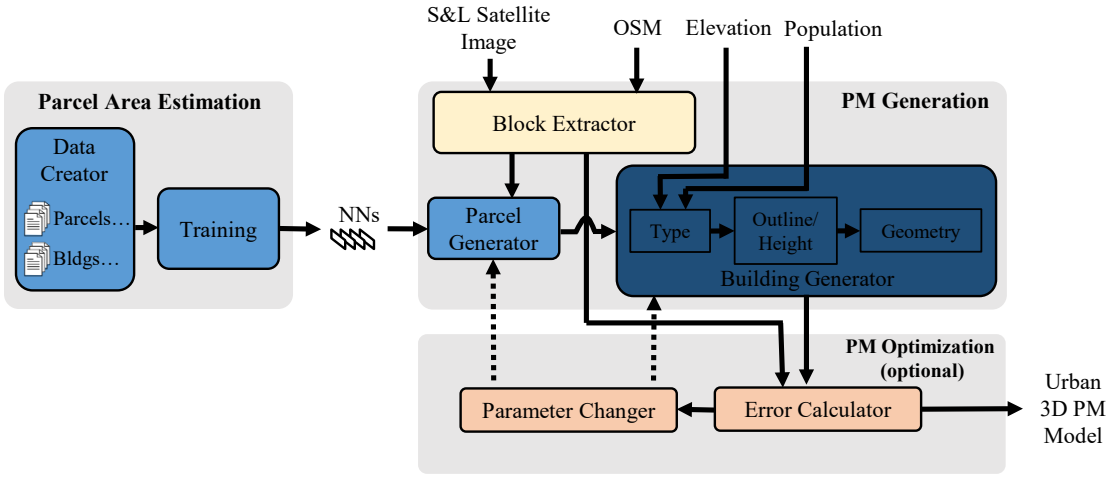


Figure 3.1. *Pipeline.* Our approach consists of a procedural model generation component, a parcel area estimation component that is setup during offline processing, and an optional procedural model optimization component. The block extractor receives as input the segmented and labeled satellite image (S&L), while the Building Generator receives the parcel output as well as information about the block's estimated elevation and population.

image segmentation methods exist, such as [76]–[79] as well as commercial solutions such as eCognition [80]. 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) [81] to obtain roads, unless segmentation provides road labels. Third, our method also uses publicly-available geo-registered global height data [82] and, fourth, our system exploits publicly-available global population data [83], which we already have for the globe. These datasets are very coarse: for example, the height data is a sample every 30 meters with a vertical accuracy of about 5 meters.

Processing

During automated processing, our approach (Figure 3.1) automatically produces a 3D urban procedural model. As a preprocess, the *parcel area estimation component* uses an

Table 3.1. *Summary of Data Sources.*

Data Name	Data Source	Resolution	Scale
Road Vector	Open Street Maps [81]	–	Most cities
Elevation Data	JAXA [82]	30 meters	Global
Population Data	LandScan [83]	1 km	Global
Satellite Data	–	1 m	Some cities

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 which 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 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 3.2).

Table 3.2. *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}
$S_{t,\mathrm{ij}}$	the triple of setback ranges corresponding to of t_{ij}
m_{ij}	the building geometry model of p_{ij}

3.3 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 3.2). 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-

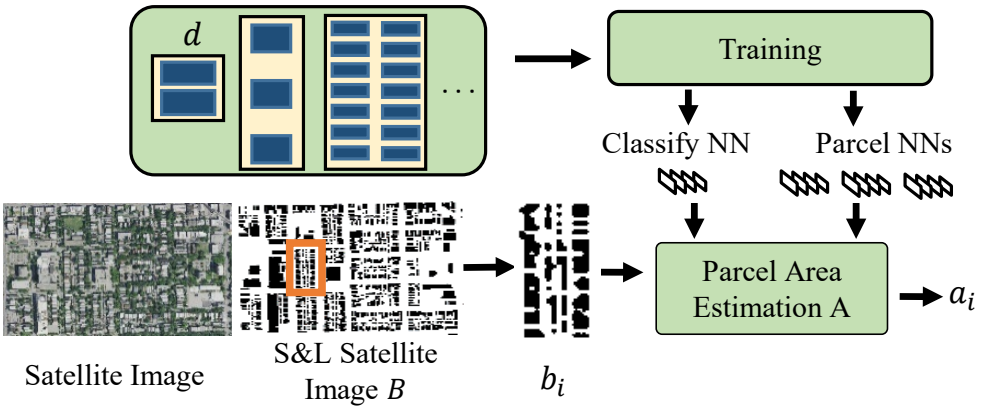


Figure 3.2. *Parcel Area Estimation.* During preprocessing, synthetic parcel training dataset is created and used to train a parcel classification neural network (NN) and several parcel area estimation neural networks. At runtime, the parcel area estimation function A receives city block satellite images b_i and estimates the average parcel area a_i.

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.

3.3.1 Canonical Representation

We use a canonical representation of city blocks which facilities 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.

3.3.2 Synthetic 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 in order 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. 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.

3.3.3 Training

We seek to obtain a robust parcel area estimator such that $area(d) = A(d)$. Parcel areas vary significantly 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). Based on our experiments, the accuracy resulting from using a different number of bins varies notably and we choose the best one (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 [84] 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. 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 in order 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 $20,000m^2$). Note that the same trained set of networks is used for all cities.

3.4 Procedural Model 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. 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, 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. After, we use parameterized rules to create buildings. Moreover, we can optionally render additional hypothetical details such as window frames, trees, lamp posts, and grass.

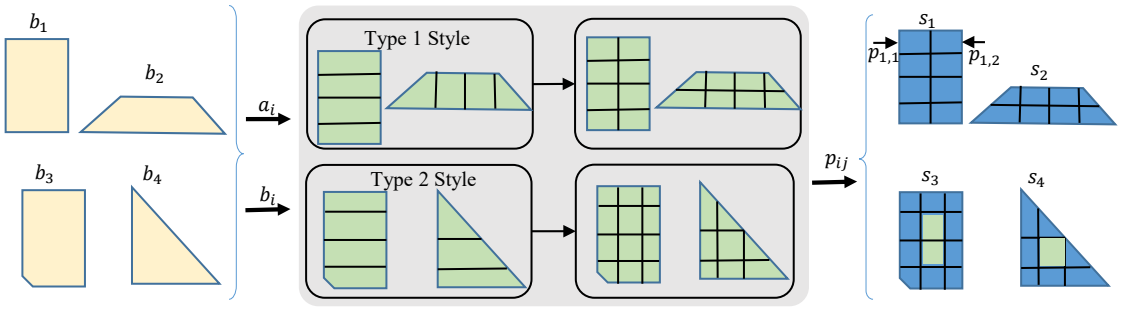


Figure 3.3. *Parcel Generation.* Our method takes as input a city block satellite image b_i as well as the estimated parcel area a_i in said image and generates a parcel subdivision in one of two styles. Then, the parcels p_{ij} are output to form the synthetic blocks s_i.

3.4.1 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 3.3). Our parcel generation method is inspired by [7] 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. 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 (parcels with no buildings).

The parcel generator P_G subdivides the block geometry g_i by recursively splitting a tightly fit oriented bounding box (OBB) of the current block partition until we obtain parcels no larger than a_i. If the generator enforces all parcels to have egress, 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

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

where q_i is the number of parcels produced for s_i.

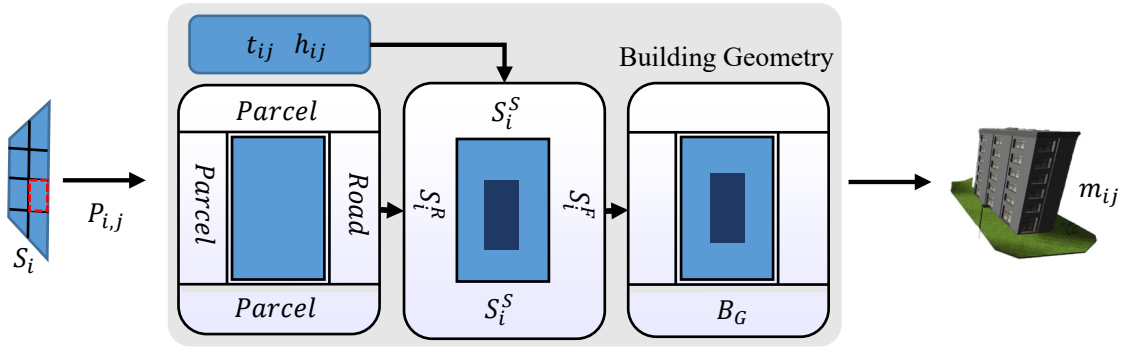


Figure 3.4. *Building Generation.* Our approach receives a target parcel p_{ij}, computes the building type t_{ij} and building height h_{ij} from global elevation and population data, uses setbacks to define the building outline, and generates a building geometry m_{ij}. The setbacks S_i^F, S_i^S, S_i^R are sampled from per-building-type setback ranges.

3.4.2 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 3.4). 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 (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 [85]. 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: 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; $[S^F_{min,k}, S^F_{max,k}], [S^R_{min,k}, S^R_{max,k}], [S^S_{min,k}, S^S_{max,k}]$ 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

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

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

3.5 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.

3.5.1 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 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; 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 so as to be larger or smaller in order 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

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

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.

3.5.2 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,

$$\begin{aligned} \mathrm{e}_n &= \frac{|count(s_{\mathrm{ij}}) - count(b_{\mathrm{ij}})|}{count(b_{\mathrm{ij}})} \\ \mathrm{e}_a &= \frac{|area(s_{\mathrm{ij}}) - area(b_{\mathrm{ij}})|}{area(b_{\mathrm{ij}})} \end{aligned} \tag{3.1}$$

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

$$argmin_{c_{mn}} \left(w_n \mathrm{e}_n + w_a \mathrm{e}_a \right), \tag{3.2}$$

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.

We rigorously evaluate our outputs and compare to state-of-the-art methods. Please see the evaluations in Section 7.1.

4. BUILDING SCALE

In previous chapter, we work with low-resolution satellite image data (1 meter $\sim$ 3 meters) and city-scale procedural reconstruction. In the subsequent chapters, we focus on higher resolution satellite imagery (e.g., 0.3 meter) and generate LOD 2 and LOD 3 building models.

Automatic creation of lightweight 3D building models from satellite image data enables large and widespread 3D interactive urban rendering. Towards this goal, we present an inverse procedural modeling method to automatically create building envelopes from satellite imagery. In the following sections, we first introduce a method to take multiple views of satellite images as input to produce a crisp procedural building model. Further, we present our system to generate procedural buildings/roofs with only a single satellite image as input.

4.1 Multi-view Satellite Images As Inputs

A portion of this section was previously published by Symposium on Interactive 3D Graphics and Games [86], https://doi.org/10.1145/3384382.3384526.

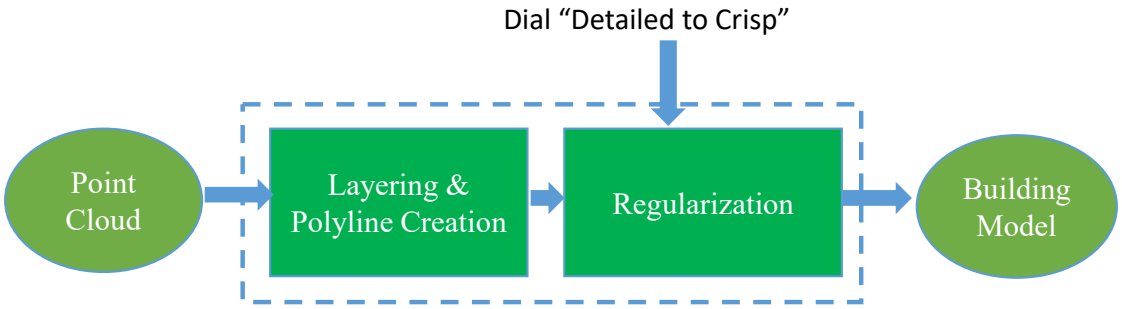


Figure 4.1. *Pipeline.* The pipeline of our geometry synthesis method.

Our key observation is that buildings exhibit regular properties (see Section 2.1 for more details). Hence, we can overcome the low-resolution, noisy, and partial building data obtained from satellite by using a two stage inverse procedural modeling technique (Figure 4.1). During a first stage, 2.5D point-clouds obtained from multi-view stereo satellite reconstruction are used to create a tree of layers and a set of line segments for each layer. During a second stage, a set of regularity constraints are pursued to arrive at parameter values producing a watertight and crisp procedural building model. Further, our methodology significantly

improves the resilience to partial/noisy data and produces crisper and more accurate models as compared to alternative satellite-based methods. Some examples are shown in Figure 4.2.

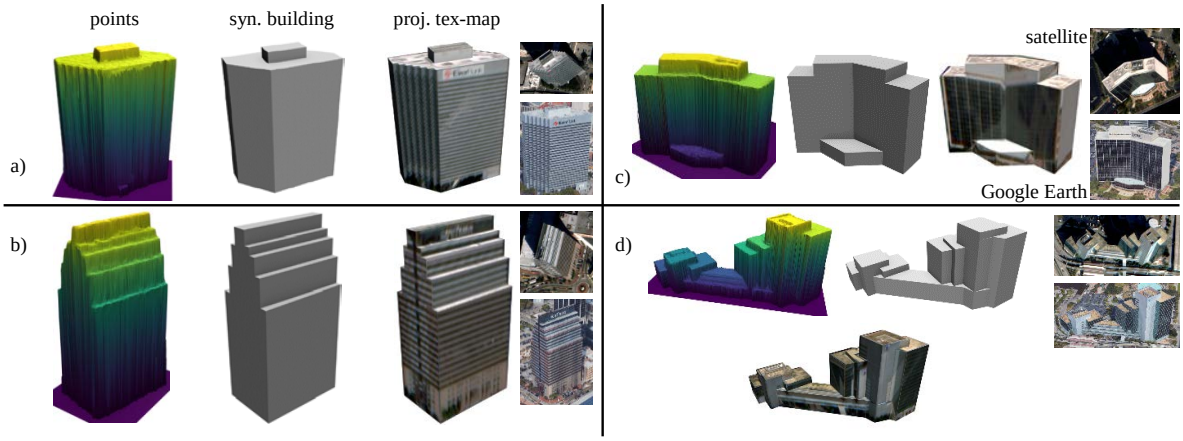


Figure 4.2. *Examples of 3D Building Models.* a-d) Our method automatically creates lightweight procedural buildings from satellite-based point clouds despite noise, occlusions, and incomplete coverage.

4.1.1 Related Work

In recent years, many researchers have been focused on building reconstruction from point clouds. [58] and [87] provide reviews of urban modeling and reconstruction. In addition, with the rapid development of deep learning, many works are using deep networks to help reconstruct and render buildings from points clouds. While numerous papers address building modeling from ground/aerial/LIDAR data, very few works address full building modeling from satellite data. Nonetheless, we discuss and compare to some ground/aerial methods that start with a point cloud as does our method. Those works could be roughly divided into 4 classes: Planar Primitive Fitting, Volumetric Primitive Fitting, Semantic Reconstruction and Deep Learning approaches (discussed in Chapter 1).

Planar Primitive Fitting. These approaches usually start with extraction/detection of planar primitives (e.g., planes), and then generate the final model with a set of primitives or further optimize them so as to create compact and visually appealing 3D models. Currently, two common methods of plane detection in point clouds are Region Growing (RG) [88], [89] and Random Sample Consensus (RANSAC) [90]. Chen and Chen [91] describe a pipeline

to reconstruct the geometry of buildings by detecting planar surfaces with a RG algorithm, and then a graph is created to represent the relationship between those planes. Finally a complete polyhedron is obtained after computing the plane connections. [92] extracts a large number of planes using the Efficient RANSAC algorithm [93] and then intersects those planes to form a set of axis-aligned candidate boxes. The final result is the subset of the boxes that have good data support and are smooth. Later, [94] generalized the same idea to reconstruct general piecewise planar objects. Their method seeks an optimal combination of the intersected planes under manifold and watertightness constraints.

Other primitive extraction methods have also been proposed. [34] yield impressive models but their work assumes availability of 2D building footprints, polygonal meshes, and both aerial and street-level imagery. [95] also produce clean buildings but their LIDAR-based approach does not work well with satellite based point clouds.

Though some of these works produce crisp building models suitable for interactive rendering their approaches do not work well for the relatively noisy and low-resolution satellite data (see later comparisons in the Results section), and also they may encounter computation bottlenecks for large complex buildings and urban scenes (e.g., we used [94] on a similar size urban environment as ours and encountered very long processing times). Solutions for such large models results in a huge number of candidate primitives and the computation may not be affordable. In contrast, our approach benefits from a simple layering strategy and we avoid the difficulty and inefficiency of finding the plane/surface primitives. Instead, we look for simpler line segments or curved primitives in 2D layer images. In our pipeline, we also make use of high-level architectural regularization terms to improve the quality of our models.

Volumetric Primitive Fitting. These approaches use a customized 3D model library which includes cube, sphere, cylinder, and other basic solids. The 3D building reconstruction is done by directly fitting the primitives. [96] presented an interactive tool called SmartBoxes to reconstruct building structures directly from point clouds using cuboid primitives. This method achieves appealing results with significant user interaction. [97] proposed a reconstruction method for indoor environments based on constructive solid geometry (CSG). They first split the 3D space into a set of horizontal slices, each of which shares the same

horizontal structure, and each horizontal slice is used to find 2D rectangular primitives. In the end, they extend the 2D primitives into 3D models. Usually these methods suffer from the limitation of using a fixed primitive library and when it comes to the noisy, incomplete satellite point cloud, it's also difficult to guarantee the accuracy of reconstruction.

Semantic Reconstruction. These methods start with segmenting the point cloud into meaningful labels (e.g., ground, building, roof, facade, etc..). and then applying the aforementioned primitive fitting methods in order to perform the reconstruction. [98] segment point clouds into meaningful structures, such as the ground, facades, roofs and roof superstructures, and then use polygon sweeping to fit predefined templates for buildings, They produce compelling results but only have a fixed set of template buildings. To obtain a level-of-detail representation of urban scenes, [99] extract a large set of plane candidates after classifying the point clouds, and then a surface model is extracted from a set of 3D arrangements based on a min-cut formulation. [100] apply a semantic segmentation and extract roof points from the point cloud. Then they iteratively apply RANSAC [90] to detect one or more core roof shapes. These roof shapes are then extended to the ground plane. The work does not include any architecturally inspired decimation or simplification to yield crisp and lightweight building models (note: in results we show a visual comparison).

4.1.2 Geometry Synthesis

Our approach is based on a regularity assumption exploited via an energy-based optimization. The optimization seeks to alter an initial polygonal model so as to produce an output that most likely resembles the underlying structure even if partial/noisy data is given. We describe our key observation, the architectural priors, layering, and regularization.

Architectural Priors: Given a hypothetical function A maximized when the provided model is equal to ground truth, we can state our geometry synthesis goal as maximizing A. Instead of arbitrarily changing a building's vertices, edges, and polygons we change them in a structured way via procedural parameters. In particular, we define $B(P_{\mathrm{i}})$ to be a parameterized procedural generation function where the parameter values P_{i} define a building that uses a set of architectural properties to varying amounts. Thus, our goal

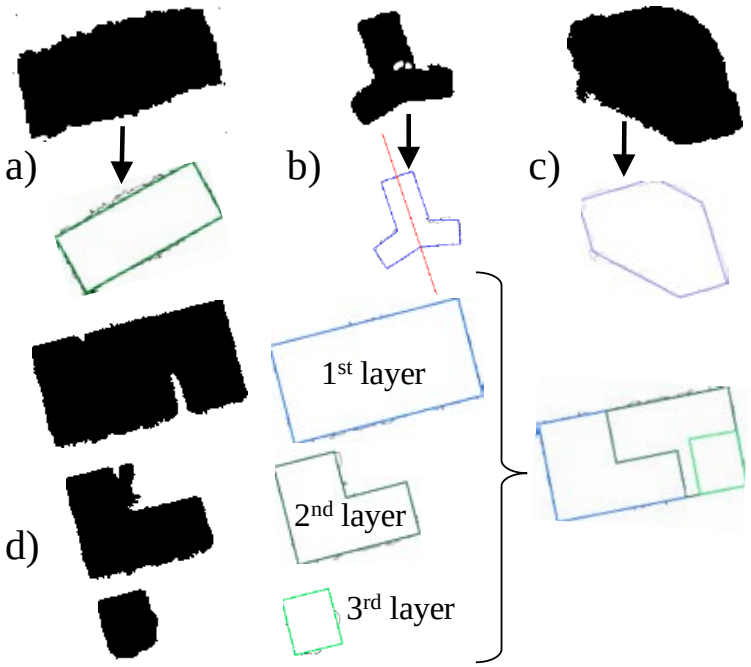


Figure 4.3. *Architectural Priors.* Geometry synthesis enforces some/all of a) parallel walls, b) symmetry (about an axis), c) predetermined corner angles, and d) inter- and intra-layer alignment.

is to maximize $A(B(P_i))$. This approach improves the resilience to noisy/partial data and produces crisp complete models.

Our approach includes the following extensible set of architectural properties (Figure 4.3):

- Symmetry: buildings might exhibit symmetry.
- Parallelism: walls might be parallel to each other.
- Predetermined Corners: corners of a building often form known angles (e.g., 90 or 135 degrees).
- Alignment: often walls between adjacent floors, or nearby same-floor segments, are aligned to each other.

However, in practice even if the listed properties are present we rarely have the ground truth data for defining the function A. Our conjecture is that the parameter set that maximizes A can also be found by maximizing

$$\alpha R(B(P_{\text{i}})) + (1-\alpha)S(B(P_{\text{i}})), \tag{4.1}$$

where R quantifies the regularization of a generated procedural model (i.e., how well the set of architectural priors are enforced) and S measures the similarity of the produced model to a mesh constructed with the incoming point cloud. Intuitively, this expression implies that by balancing regularity with similarity, we can find an optimal combination in the sense of maximizing the similarity to ground truth. [101] arrived at a similar conclusion. Hence, we can use expression 4.1 as a proxy to maximizing A.

In order to calibrate the level of regularization, we use a small set of example buildings to calibrate constant and varying procedural parameters. In particular, given a small sample of known building structures, we perform the following minimization:

$$min_{P_c} \| (\alpha R(B(P_k)) + (1-\alpha)S(B(P_k))) - A(B(P_k)) \|, \tag{4.2}$$

where $k = [1, K]$ for K known buildings, $P_k = \{P_c, P_d(I_k)\}$, $P_d(I_k)$ are parameters computed from satellite image set I_k for building k, and P_c are constant for all buildings in this geographic region (or everywhere). Thus, once we have P_c, at runtime we can produce crisp procedural models from the satellite images despite incompleteness and noise, and without needing a priori known models.

Layering and Polyline Creation: The first phase of geometry synthesis is dividing the point data into a tree of horizontal layers. We observe that (i) buildings may consist of segments of different vertical heights and (ii) some non-adjacent segments of the building might be of the same height. To perform the layering, we place all points into a grid with voxel size equal to the satellite image pixel size of 0.3m. Then, we use a relative layering threshold to determine when the intersection-of-union (IOU) between the current layer and next horizontal single-cell voxel slice is big enough to begin a next layer. See Figure 4.4 for one example.

Our layering scheme produces a tree of layers, as opposed to a linear array of layers, because of the aforementioned second observation: a building might have non-adjacent seg-

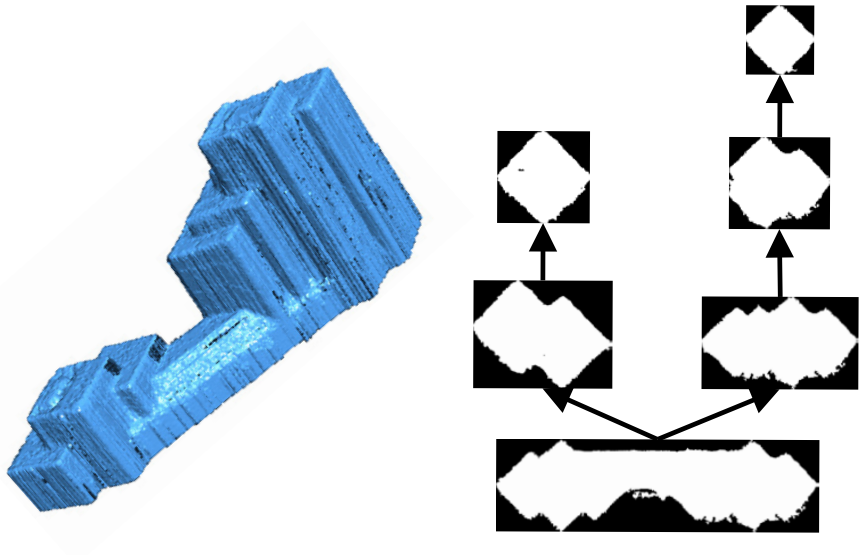


Figure 4.4. *Layering.* We show the input point cloud as a tree of layers. To the left is the point cloud and to the right is the tree of decomposed layers.

ments of the same height. When creating a new layer, we solve an approximate connected component problem to determine if the new layer has multiple disjoint components (e.g., a lower part of a building then becoming two separate towers in the upper part).

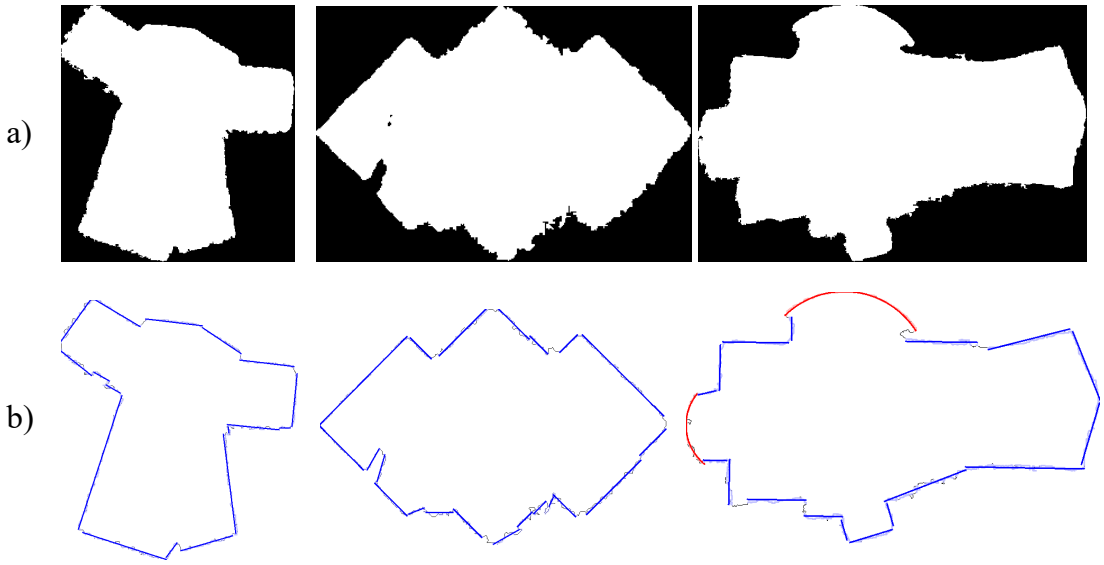


Figure 4.5. *RANSAC.* Layer examples of using a RANSAC-based method to detect line segments. a) Layer inputs, b) Line segments.

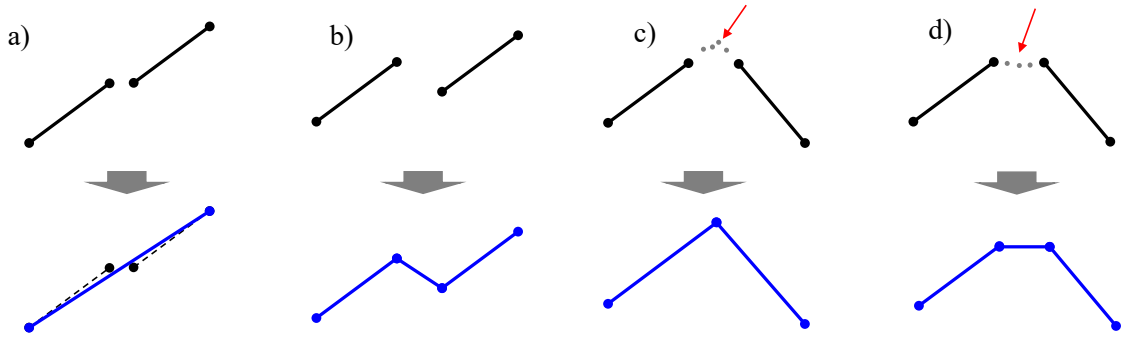


Figure 4.6. *Heuristics.* Heuristics for Connecting Line Segments to Define Each Layer. a) Almost collinear and close enough, b) Almost collinear, c) Some supporting points near the intersection, and d) No supporting points near the intersection.

We choose a representative slice for each layer and compute a single closed polyline per layer. Since a layer might consist of multiple horizontal one-cell slices through the grid, we must obtain a consensus of the layer geometry. We attempted several consensus estimation schemes and found the best one to be choosing the slice that is most similar to all other slices in the layer. Then, we use a RANSAC-based method to determine line segments with significant support (i.e., line segments that pass through, or nearly through, a sufficient number of points). The line segment determination algorithm makes use of a support threshold parameter and a closeness threshold parameter. See Figure 4.5 for some examples. To form the closed polyline, our method uses several heuristics. Figure 4.6 describes visually some of the heuristics. For example, almost collinear and close-by line segments are replaced with a best fitted single line segment and almost coincident line segment start-end points are snapped together. Other scenarios (e.g., b, c and d of Figure 4.6) are shown as well, collectively using various parameter values which we have determined empirically.

Regularization: In this second phase, we iteratively alter the regularization parameters (e.g., thresholds, weights, α) so as reduce the difference between the values produced by the similarity and regularization metrics and the desired values as per equation 4.2 (Figure 4.7). This iteration continues under user control or under an external optimization loop. The resulting set of regularization parameters are later used to produce building models. The

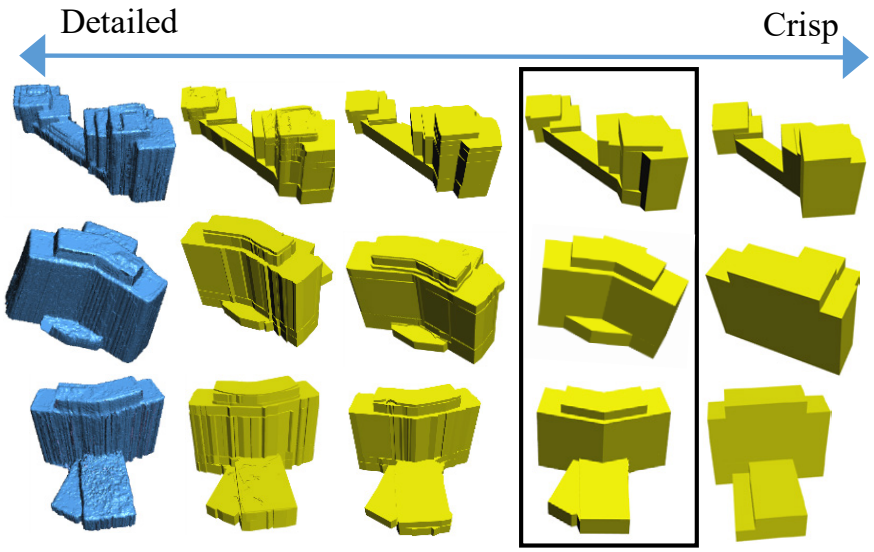


Figure 4.7. *Regularization.* We alter regularization parameters until producing an output of desired detail/crispness.

parameters can be used globally or new values can be computed for each region. In our case, we compute them once and use in all areas.

The similarity and regularization metrics are calculated via a set of functions computed using each layer's poly-lines. The similarity metric computes the IOU between the current poly-line and the original poly-line of a layer. The regularization metric makes use of several sub-metrics explained in the following paragraphs.

- *Symmetry Metric.* This function seeks an axis of reflective symmetry for the provided poly-line. For a proposed axis, the subset of the poly-line on one side of the axis is reflected over the axis. Then, if the reflected poly-line and existing poly-line are very similar (e.g., determined via IOU), it indicates a strong reflective symmetry. Our approach performs a gradient descent to compute the rotation angle and 2D intercept point that most reduces the aforementioned error sum. We prime the optimization by first evaluating the cost function with a sampling of axis rotations (e.g., one every 10 degrees) and assume all axis pass through the midpoint of the poly-line.

- *Corner Metric.* This function is evaluated for wall-to-wall corner angles of 90, 135, or 180 degrees. Given two adjacent wall segments that exceed a length threshold (e.g.,

1 meter), we determine the typical corner angle to which they are most similar. If the actual angle is within a threshold of the typical corner angle (e.g., 10 degrees), we compute an error metric proportional to the angular difference (else zero).

- *Parallel Wall Metric.* This function computes the angular difference between each wall segment and the most parallel other wall segment in the layer. If the angular difference is smaller than a threshold (e.g,. 3 degrees), the error is the angular difference (else zero).

- *Alignment Metric.* This function has both an inter- and intra-layer component. The inter-layer component seeks for a poly-line segment of the layer above it and also a segment beneath it that are closest in both orientation and distance. The intra-layer component seeks for each segment in the poly-line another segment closest in orientation and distance (but not adjacent). If the paired segments from the intra- or inter-layer components are similar enough, they are considered candidates to be aligned. The error value is a weighted sum of the orientation difference and distance value.

- *Curved Wall Metric.* This function seeks to find a sequence of polyline segments that approximately form a circular arc of least a pre-specified minimal angular span. If found, the sequence is considered a candidate to become a (circular) curved wall.

The aforementioned calibration process is performed via an additional loop placed over both the layering and regularization phases. The calibration process first decomposes a set of known models into a tree of layers. Then, a simple minimization is computed to reduce the average difference between the computed layers and the known-model layers, thus yielding best values for all the parameters described in the aforementioned layering and regularization phases. We perform this calibration once using five buildings that were manually modeled in one test area. We iteratively alter the parameters and confirm a reasonable convergence by visual inspection. Please find the evaluations in Section 7.2.1.

4.2 A Single Satellite Image As Input

A version of this section is pending publication in Computer Graphics Forum (Eurographics 2022).

Urban procedural modeling has benefited from recent advances in deep learning and computer graphics. However, few, if any, approaches have automatically produced procedural building roof models from a single overhead satellite image. Large-scale roof modeling is important for a variety of applications in urban content creation and in urban planning (e.g., solar panel planning, heating/cooling/rainfall modeling). While the allure of modeling only from satellite images is clear, unfortunately structures obtained from the satellite images are often in low-resolution, noisy and heavily occluded, thus getting a clean and complete view of urban structures is difficult.

In this section, we present a framework that exploits the inherent structure present in man-made buildings and roofs by explicitly identifying the compact space of potential building shapes and roof structures. Then, we utilize this relatively compact space with a two-component solution combining procedural modeling and deep learning. Specifically, we use a *building decomposition component* to separate the building into roof parts and predict regularized building footprints in a procedural format, and use a *roof ridge detection component* to refine the individual roof parts by estimating the procedural roof ridge parameters. Our framework yields both improved results over prior methods and produces discrete vector-based procedural structures, all from a single satellite image without the need of high-resolution aerial images, LiDAR or point-cloud data. In our comparisons to multiple techniques used well-established datasets, our method is consistently better than prior work both quantitatively and qualitatively.

4.2.1 Related Work

Our work builds on procedural and inverse procedural modeling, deep generative modeling (see related work in Chapter 1), and building footprint extraction and roof reconstruction in order to automatically produce procedural roofs from a single satellite image.

Building Footprint Extraction. Many state-of-the-art deep segmentation networks (e.g., [35]–[39], [41]–[43]) can be applied to building footprint extraction. Specifically, Fully Convolutional Networks (FCNs) [35] introduce deconvolution via upsampling operations and provide an alternative to fully connected layers in classification models. U-Net [36] infers high-resolution feature maps by joining the top-down and bottom-up pathways with lateral connections. DeepLab [37], [39], [42] maintains high-resolution by replacing strided convolution with atrous convolution. However, these approaches include many more content pixels than boundary pixels. This imbalance causes them to produce inaccurate building/roof edges.

To solve this challenge, polygon-based building boundary delineation work has been proposed. These methods focus mainly on active contours and on edge (point) assembling. In [102], [103], they present frameworks which utilize the strengths of both CNNs and active contour models [104] to produce an end-to-end polygon-based output model. Although the active contour approaches improve mask coverage compared to the aforementioned CNN-based semantic segmentation, blob-like contours that do not match building boundaries are produced. In [105]–[107], these approaches start with detecting/extracting building primitives (e.g., corners, edges or regions) using CNNs and then employ other techniques (e.g., RNNs, integer programming or Graph Neural Networks) to assemble them leading to polygon-based building outputs. However, the usability of these methods is limited because they cannot predict complex shapes because of deficiencies in primitive feature extraction and detection modules. Moreover, the RNN and GNN modules are computationally expensive and [106], [107] require extra building corner and edge annotations. Recently, Li et al. [108] design an algorithm pipeline ASIP to improve the work of [105] by extracting and vectorizing objects in images with polygons – we compare to this method, amongst others, in our results section.

Building and Roof Reconstruction. Building and roof reconstruction is an active research area. However, it usually requires multiple data sources (e.g., LiDAR, DSM, DTM, point clouds, etc.) and few works focus on reconstruction from a single satellite image. Arefi and Reinartz [109] directly detect roof ridges utilizing high resolution DSMs and orthorectified satellite images. Zheng and Zheng [110] propose a hybrid approach, combining the data- and model-driven approaches to generate LoD2 building models by using LiDAR, 2D

building footprints and high resolution orthophoto images. Li et al. [111] present a novel approach to segment the roof planes from airborne LiDAR point clouds using hierarchical clustering and boundary relabeling. Ywata et al. [112] introduce a method to extract building roof boundaries in object space by integrating a high-resolution aerial images stereo pair and three-dimensional roof models reconstructed from LiDAR data. In [113], [114], they work on a deep learning-based approach to detect and reconstruct roof parts of buildings from a single image. However, they require high resolution *aerial images* and annotate the dataset for roof ridges and building boundaries. In [115], [116], they present deep learning based approaches for automatic 3D building reconstruction. However, they need elevation data (e.g., DSM) for training and their results are not regularized. None of the mentioned approaches automatically reconstruct roofs using only a single satellite image as input.

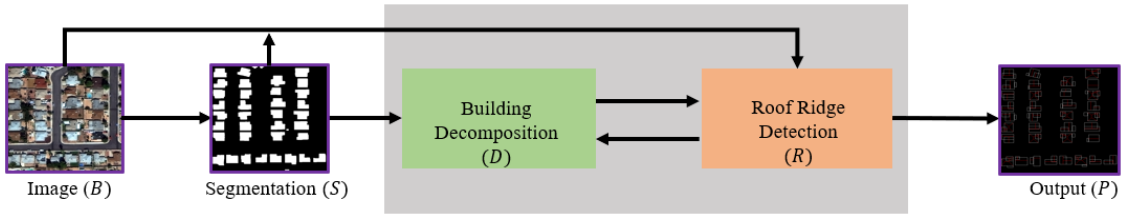


Figure 4.8. *Pipeline.* Our approach consists of a building decomposition component and a roof ridge detection component. A single satellite image gets segmented by a building instance segmentation model (e.g., Mask R-CNN [117]). And our components directly work on the initial segmentation image and generate procedural urban output.

Our approach is summarized in Figure 4.8. As a preprocessing step, individual building images $\{b_1, ...b_i, ...\}$ and corresponding segmentations $\{s_1, ...s_i, ...\}$ are extracted from the input satellite image B and the segmentation image S (e.g., s_i is cropped from S based on a loose oriented bounding box of the building instance). These are then given to the building decomposition D and roof ridge detection R components to yield the procedural output P. To assist with terminology, we provide a table summarizing the subsequent variables (Table 4.1).

In later sections, we first give a general overview of the potential building shapes and roofs space, and then describe the building decomposition component including the processing

Table 4.1. *Variables and their meanings.*

variable	meaning
B	an input satellite image of an urban area
b_{i}	a building image of B
S	the segmented image of B
s_{i}	the segmentation of b_{i}
m_{i}	the edge map of b_{i}
D	building decomposition component
t_{i}^{D}	the building shape family of b_{i}
c_{i}^{D}	the building configuration of b_{i}
r_{ij}	the j_{th} rectangle in c_{i}^{D}
R	roof ridge detection component
e_{ij}	the edge map of r_{ij}
t_{ij}^{R}	the roof shape family of r_{ij}
c_{ij}^{R}	the ridge configuration of r_{ij}

details, synthetic data creation and training of neural models. Then, we present the roof ridge detection component in a similar manner. Note: Having building decomposition component and roof ridge detection component separate can significantly reduce the required training data (i.e., if not, each building part must support diverse roof types with different roof ridge configurations) and makes training more efficient.

4.2.2 Potential Building Shapes and Roofs (PBSR)

The PBSR is an approximation to represent all possible building and roof structure combinations. For this purpose, we propose a graph representation: each node stands for a singular roof and each edge signifies the connection of adjacent roof parts.

Building Shape Families

Considering the number of nodes (or roof parts see Figure 4.9) and the different connection scenarios of roof parts, the number of possible building shape families is:

$$\sum_{\mathrm{i}} F_{\mathrm{i}} \qquad and \qquad F_{\mathrm{i}} = \sum_{\mathrm{j}} T_{\mathrm{ij}} \tag{4.3}$$

where F_{i} is the i_{th} building shape family defined according to the number of nodes. T_{ij} is the j_{th} topology of F_{i}, and T_{ij} is defined by the connections within F_{i}. For example, the building shape families of F_1, F_2, F_3 and F_4 in Figure 4.10 consist of one, two, three and four roof parts respectively. By means of graph isomorphism and the assumption of the graph being connected, F_1 and F_2 can only have one topology, F_3 can have two topologies, and F_4 can have up to four topologies. Intuitively, the building shape "L" and "T" belong to T_{21}. T_{32} includes "U" and "Z" building shapes (see Figure 4.9). A closed four-side building shape is in T_{43}. However, T_{31}, T_{41}, T_{42} and T_{44} are possible shapes but not common in the real world. In summary, we can theoretically grow the space of possible building shape families to larger values for i and account for any actual building. However, as discussed later, by limiting the possible set of building shape families, and their configurations, to a rather compact number, we can practically capture most buildings in our datasets. For example, considering i up to 4 results in 8 possible parameterized building shape families.

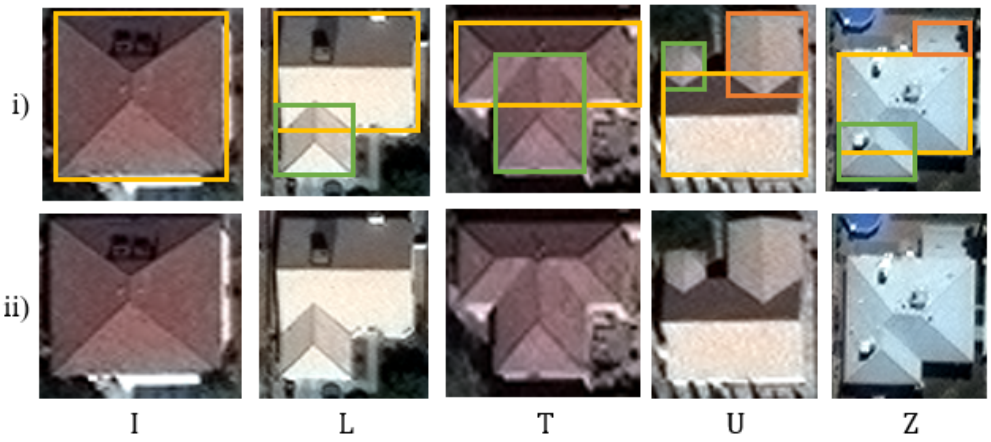


Figure 4.9. *Roof parts.* We show roof parts of certain building shapes (I, L, T, U, and Z). For each, i) one or more roof parts in different colors. ii) the corresponding building image.

Roof Families

As for roof families, we consider two edge types appearing in typical roofs: external edges (e.g., eaves) and internal edges (e.g., ridges and hips). We assume the perimeter of a single roof part are the external edges and the internal ridges follow the main direction of the roof

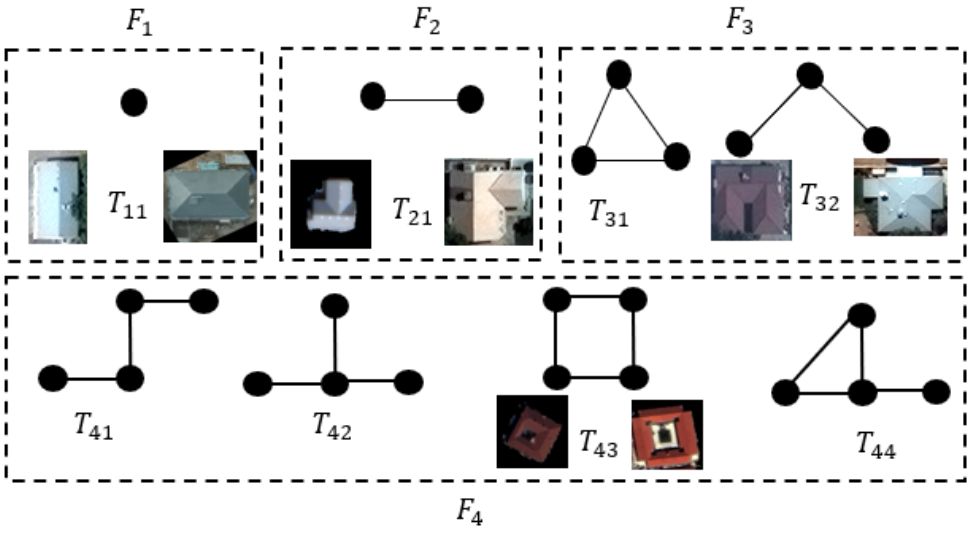


Figure 4.10. *Potential building shape families.* We show building shape families of F_1, F_2, F_3 and F_4. See main text for more details.

(e.g., parallel or perpendicular to eaves). Hips are the internal edges that connect ridges to corners. In our current implementation, we support flat, gable, hip, pyramid and half-hip roof families (see Figure 4.11) which includes most common types according to [118]–[120].

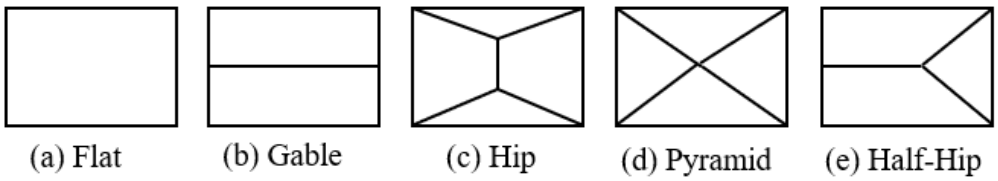


Figure 4.11. *Potential Roof Families.* We show our supported roof families.

4.2.3 Building Decomposition Component

In this part, we describe the processing details of our building decomposition component D, and also how we create a synthetic dataset and train the classifier. As shown in Figure 4.12, the component D consists of three parts: the previous described PBSR, a building shape family classifier, and a building family configuration recognizer.

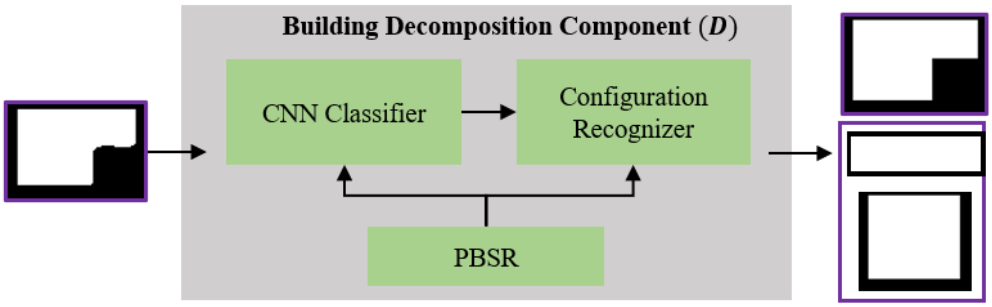


Figure 4.12. *Building decomposition component.*

Classifier

Taking the segmentation image s_{i} as input, the CNN-based classifier predicts the building shape family t_{i}^{D}. The classifier network is a ResNet [121] with a modification of the last fully-connected layer to having the number of supported building shape families. We train the classifier with 185,700 synthetic images and achieve 96.5% classification accuracy when testing on segmentation images s_{i}.

Configuration Recognizer

Next, we need to recognize the precise configuration of the determined building shape family. This enables producing a specific parameterized procedural output for the building footprint and subsequently for estimating roof parameters.

The configurations are determined by the arrangements of parameter values of roof parts. The parameters for each single roof part (or node) is $r = \{x, y, w, h\}$ (assuming it's a rectangular roof) where (x, y) is its top-left corner and (w, h) is its size. Thus, for all building shape families, the total number of possible configurations is

$$\sum_{\mathrm{i}}\sum_{\mathrm{j}}\sum_{k} C_{\mathrm{ij}k} \qquad \textit{and} \qquad C_{\mathrm{ij}k} = \{r_{\mathrm{ij}k1}, r_{\mathrm{ij}k2}, ..., r_{\mathrm{ij}kn}\} \tag{4.4}$$

where $C_{\mathrm{ij}k}$ is the k_{th} configuration of T_{ij} and contains $n(n \geq 1)$ roof parts. However, this exhaustive list has redundancies that we seek to omit to achieve better performance (e.g., faster search). We ignore configurations that are affine transformations of other configura-

tions (e.g., translation, flip, mirror, etc.). Further, we split the image into grids (setting grid size to 2 pixels) and iterate parameters in the grid space. Additionally, according to our preeliminary analysis of our used portion of SpaceNet [52], roughly 90% of building shapes are covered by I, L, T, U, and Z (see Figure 4.9) . Hence, we only consider the families T_{11}, T_{21}, T_{32} and T_{43}. In summary, we support 4520 configurations in total.

To search for the configuration of the current building shape family t_{i}^{D}, we apply the following strategies. Since the configurations only consists of canonical instances without those generated by transformations, we apply a series of image processing methods to s_{i}; e.g., cropping the s_{i}, resizing the s_{i} to size $(120, 120)$, centering the s_{i} with 4 pixels margin, and then applying transformations including flipping s_{i} horizontally or vertically, and rotating s_{i} (e.g., 90, 180, 270 degrees) before searching for the configuration. Eventually, we find the best match $c_{\mathrm{i}}^{D} = \{r_{\mathrm{i}1}, ..., r_{\mathrm{ij}}, ...r_{\mathrm{i}q_{\mathrm{i}}}\}$ using intersection-of-union (IOU) (the best IOU of transformed s_{i} and the generated footprint image based on a configuration). The matched c_{i}^{D} will be passed to the roof ridge detection component for subsequent processing.

Synthetic Data Creation and Training

While assuming building image inputs (e.g., b_{i}) is an option, it requires to manually annotate a large set of real-world training images for both building footprints and roof ridges. Instead, we leverage synthetic dataset to avoid labor-intensive annotations and thus can do self-supervised training more easily.

Regarding the creation of a synthetic dataset, we consider building structure regularities. Buildings exhibit properties such as straight walls, parallel walls, walls meeting at one of a set of predetermined angles (e.g., 90 or 135 degrees), symmetrical arrangements, and other features. For the sake of simplicity, we focus on straight walls, parallel walls and right angle regularities meaning a rectangle represents each single roof part. Our synthetic dataset consists of all the configurations of building shape families discussed previously. Additional types can be added to our dataset easily.

Moreover, in order to handle noisy and irregular building footprint segmentation, aside from typical data augmentation techniques (e.g., flip, translation, rotation, etc.), we add

noise to our clean/regularized synthetic images (see Figure 4.13 (a)). We apply random occlusion or bumps (e.g., different shapes and sizes) around the footprint boundaries. Based on preliminary experiments, we furthermore include different levels of transformations (e.g., low-noisy, medium-noisy and high-noisy: see Figure 4.13 ii) from left to right) when training the classifier.

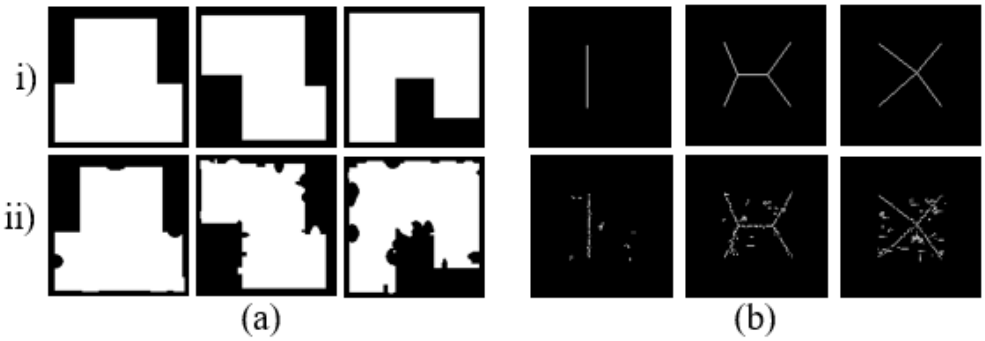


Figure 4.13. *Data Transformation.* We show (a) building footprints, and (b) roofs. For each, i) clean and regularized synthetic images. ii) Images after corresponding transformations.

4.2.4 Roof Ridge Detection Component

In the following, we provide details about estimating procedural roof parameters, creating synthetic roof dataset, and training the roof family classifier. The component R (see Figure 4.14) also has three parts: the previously used PBSR, a roof type classifier, and a roof ridge configuration recognizer.

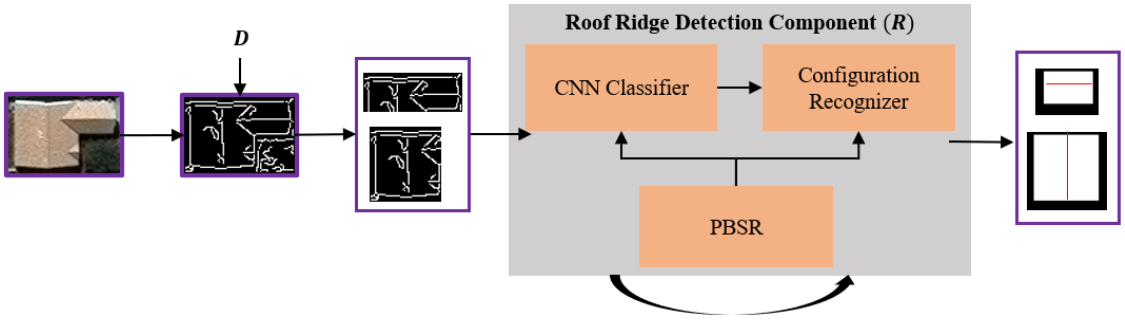


Figure 4.14. *Roof ridge detection component.*

Classifier

Given a building image b_i, we first apply histogram equalization to adjust color contrast, and then make use of an edge detector ([122] as default) to generate an initial edge map m_i. m_i is subsequently cropped into an edge set $\{e_{i1}, ...e_{ij}, ...e_{iq_i}\}$ following the aforementioned configuration c_i^D of b_i. For any e_{ij}, the classifier predicts the roof family type for the corresponding roof part (or node). The classifier network is a ResNet [121] with modification of the last fully-connected layer to the number of supported roof family types. We train the classifier with 119,500 synthetic images and achieve 95.6% classification accuracy when testing on edge maps e_{ij}.

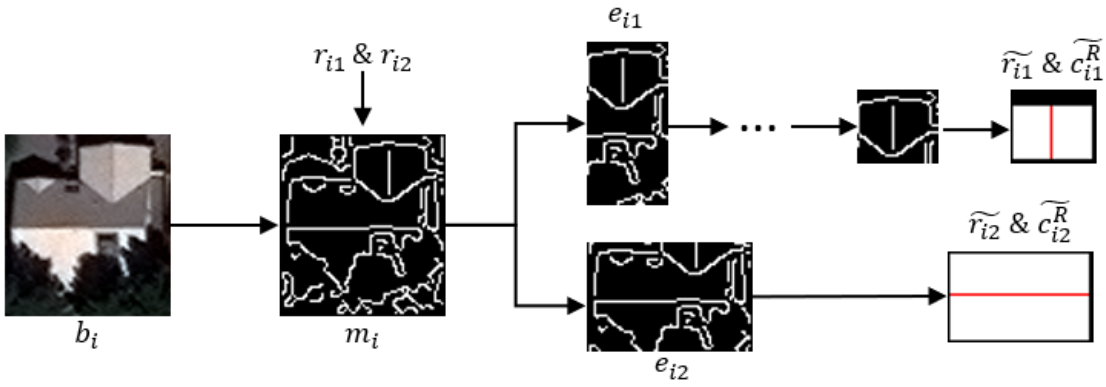


Figure 4.15. *Roof Processing Step.* We show an example to illustrate how our recognize roof ridges and refine roof parts.

Configuration Recognizer

Given an edge set $\{e_{i1}, ...e_{ij}, ...e_{iq_i}\}$ of b_i, the goal is to detect and estimate the roof structures for the whole building. For example, if $q_i = 1$, b_i consists of a single roof. We recognize the roof family type and find the best matched configuration c_{i1}^R by maximizing supporting points in e_{i1} and the candidate ridge coverage. The supporting points are those edge points of e_{i1} whose distance to the candidate ridge line is smaller than a threshold value (e.g., setting to 5% of the length of the candidate). If $q_i \geq 2$, we need to refine the sizes of roof parts (see roof part r_{i1} in Figure 4.15). We start with recognizing the roof family type for each e_{ij}. Based on the result, we decide the main roof part (e_{i2} in Figure 4.15). Afterwards,

we focus on iteratively refining the rest of the roof parts. We start with decreasing the overlap area by half each time (binary search by following the direction perpendicular to the main roof). For each iteration, we predict the roof family type and and find the corresponding best matched configuration until we find the best matching (e.g., setting candidate ridge coverage to 90% and maximizing the number of supporting points). In the end, it leads to the final roof set $\tilde{c^D}_{\mathrm{i}} = \{\tilde{r}_{\mathrm{i}1}, ..\tilde{r}_{\mathrm{ij}}, ...\tilde{r}_{\mathrm{i}q_{\mathrm{i}}}\}$ with their corresponding ridge configuration set $\{\tilde{c^R}_{\mathrm{i}1}, ..\tilde{c^R}_{\mathrm{ij}}, ...\tilde{c^R}_{\mathrm{i}q_{\mathrm{i}}}\}$.

Finally, we collect the roof set and roof ridge configurations of each b_{i}, and combine them into procedural output for B.

Synthetic Data Creation and Training

We generate synthetic roof images to support the roof family types. For the purpose of representing noisy and irregular edge maps e_{ij}, aside from typical data augmentation techniques, we further transform the synthetic roof images by adding random noisy curve lines and randomly removing small parts of the edges (see Figure 4.13 (b)). During training, similarly we apply different levels of transformations.

We quantitatively and qualitatively evaluate our outputs and compare to state-of-the-art methods. Please see the evaluations in Section 7.2.2.

5. FACADE SCALE

A portion of this chapter was previously published by Springer International Publishing [123].

Automatic satellite-based reconstruction enables large and widespread creation of urban areas. However, satellite imagery is often noisy and incomplete, and is not suitable for reconstructing detailed building facades. In this chapter, we present a machine learning-based inverse procedural modeling method to automatically create synthetic facades from satellite imagery. Our key observation is that building facades exhibit regular, grid-like structures. Hence, we can overcome the low-resolution, noisy, and partial building data obtained from satellite imagery by synthesizing the underlying facade layout.

Our approach takes as input 3D building models obtained from point-clouds (e.g., [98]), as well as satellite image fragments projected onto the faces of the building models. The image fragments are used together with trained deep networks to find a representative sample of a facade with minimal noise, and infer its style and procedural parameters. The parameters are then used to complete the rest of the facade, and potentially other non-observed facades of a building. In the end, our approach produces complete facade layouts applied to building models. Figure 5.1 shows example results of our approach. Since we have a procedural output (instead of an image), we can zoom-in to any part of the facade and still have a crisp result, as observed in the close-up views.

Our results yield improvements over other methods applied to the same data. Over our six test areas, each spanning 1-2 km^2, our method is consistently better than the prior work we compare to quantitatively and qualitatively, and the average accuracy of several performance metrics is 85.4% despite significant occlusions, noise, and strong blurriness. Further, our deep networks are trained on a new dataset of rectified satellite facade views with ground truth segmentation that we also offer as a contribution. As far as we know, our work is the first pipeline to handle façade reconstruction based on satellite imagery despite the occlusions and resolution limitations of such imagery.

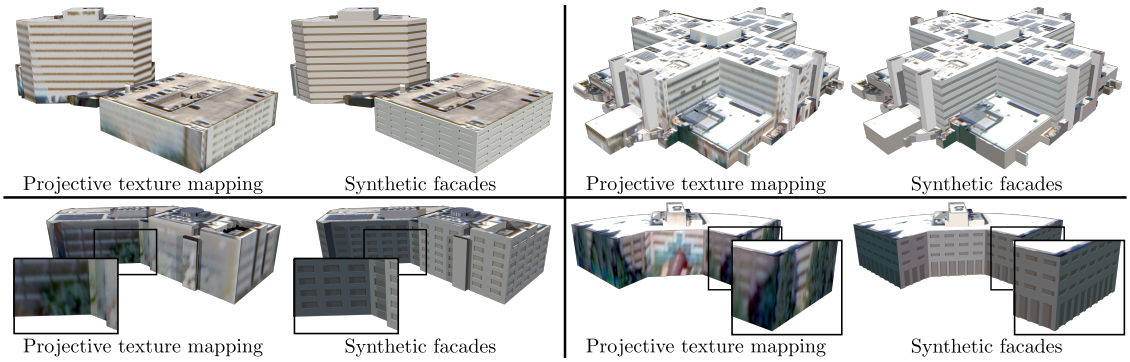


Figure 5.1. *Examples of facade synthesis and completion.* Our method automatically creates procedural facades from satellite-based images despite noise, occlusions, and incomplete coverage.

5.1 Related Work

Almost all facade reconstruction methods use ground or aerial imagery, typically rectified and rectangular. Many approaches have been followed (e.g., using dynamic programming [124], using lattices [125], using matrix approximations [126], and inferring grammars from pre-labelled segments [101], [127], [128]). However, these methods do not perform well for our very under-sampled facades. For example, see our comparisons in the results section.

More recently, deep learning based facade parsing has obtained excellent results for ground-level imagery. For example, Liu et al. [129] and Fathalla et al. [130] perform facade segmentation but assume high-resolution frontal views. Nishida et al. [33] further assumes hand-specified building silhouettes and their facade stage depends on having clear boundaries between floors and between columns. Further, none of these account for the significant occlusions in satellite-based facades. Kelly et al. [45] could automatically and realistically decorate buildings by synthesizing geometric details/textures. However, their work requires style references (e.g., façade and roof textures, window layouts) and such references from satellite would be very low-resolution and heavily occluded. Kozinski et al. [101] (and partially Mathias et al. [131]) include provisions for occlusions but depend on many assumed structural priors for numerous object classes and SIFT feature vectors. On average the facades we encounter are only 20x90 pixels in size (often significantly worse) and thus make it prohibitive to determine such detailed structure. Image-to-image translation, such as Isola

et al. [40] and Zhu et al. [132], has been proposed but does not support all of regularity, occlusions, and satellite data. From the semantic segmentation point of view, facade parsing could also be considered as a segmentation task. Many papers (e.g. DeepLabv3+ [42], EncNet [41], etc.) have shown great success with segmentation, but none of them use satellite facade data. Thus we trained those neural networks from scratch using our created satellite facade dataset (see Results section) and observe that these state-of-the-art segmentation neural networks also suffer from the low-quality of satellite facade data and cannot generate crisp facades.

Filling-in missing pixels of an image, often referred as image in-painting or completion, is an important task in computer vision. Deep learning and GAN-based approaches (e.g., DeepFill [133], PICNet [134]) have achieved promising results in this task. However, image in-painting is ill-suited for resolving shadows and occlusions in satellite facade images. First, detection of these areas is a very challenging problem, especially for satellite data. Second, even assuming these areas could be detected automatically, image in-painting approaches cannot infer correctly due to the low quality of satellite facade data. We also show in the Results section comparisons to these approaches.

5.2 Facade Synthesis

As mentioned in Section 1.4, while there might be multiple satellite images observing the same building, there is usually not a high quality satellite observation of every facade on a building due to shadows, foliage/occlusions, and limited resolution. Directly projecting the satellite images to the buildings is inadequate. Further, such texture mapping depends on very accurate image-to-image registration, geometric modeling, and complete coverage of all building facades. Our approach attempts to overcome these issues by synthesizing procedural facades using a selected subset of the available satellite imagery, and then applying these facades across the entire building. This approach has the following advantages:

- *Crisp Results.* The produced facade details will be crisp and visible at any resolution.

- *Exploits Best Observations.* Without relying on accurate RPCs and image registration, we choose the best, potentially fragmented, observations of each building and use it to obtain facade details.

- *Completes Missing Fragments.* Even if a facade/fragment is missing, we can fill-in the facade with details from a partial observation (or in worst case with details from neighboring facades).

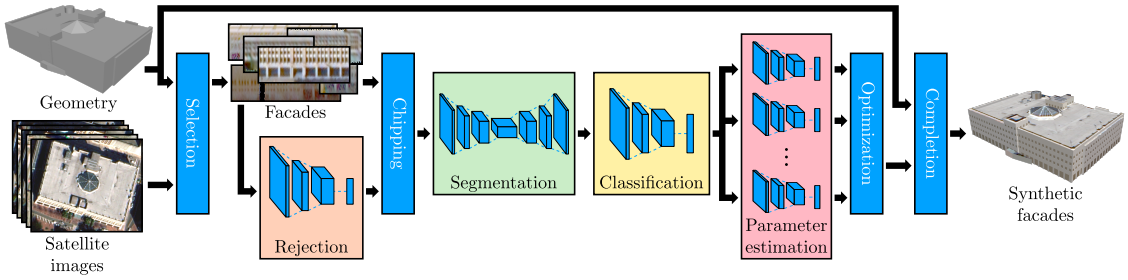


Figure 5.2. *Pipeline.* The pipeline of our multi-stage approach for facade completion and synthesis.

We provide an overview of the proposed procedural facade approach in Figure 5.2 and in the following we describe the pipeline starting with our selection method, followed by our deep-learning based facade style classification and parameter estimation, and finally our facade and building completion.

5.2.1 Selection

In a first stage, we choose the satellite image that has low grazing angle and does not have much dark pixels as the best view of the facade, and the resultingimage is used as input to the rest of the pipeline. In many cases, even the best observation of a facade is not useful due to noise, shadows, trees, and occlusions. Thus we employ a deep-learning based rejection model to prevent further processing of any such facades. Rejected facades will not undergo classification or parameter estimation, but can still receive synthetic facade layouts as part of the completion phase (Section 5.2.3).

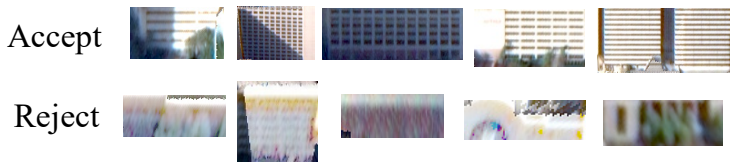


Figure 5.3. *Accept or Reject.* The first row shows facades that our rejection model will accept. The second row shows facades that will be rejected.

Our rejection network is based on a pre-trained ResNet [121] model, in which we modify the last fully connected layer to two classes: one for "good" facades to be accepted and the other for "bad" facades to be rejected. We used 120 examples of "good" facades from our facade data set and 120 examples of "bad" facades, resulting in 1920 training images in total after applying data augmentation such as flip, rotation, random crop and intensity variations. The model performs with 92% accuracy when tested on 200 test images. Figure 5.3 shows some examples of accepted and rejected facades.

5.2.2 Classification and Parameter Estimation

In a second stage, our approach estimates the style and parameters of an equivalent procedural facade representation. Our method extracts a "chip" from the selected facade image because i) satellite-based images often suffer from occlusions and thus assuming a full facade view would be prohibitive, and ii) otherwise the parameter space would be unnecessarily large as the number of floors/windows may vary significantly yet the spacing between floors and windows is regular. The procedural representation for the entire facade is obtained from the chip and then used during the next stage to complete each facade.

Chip Extraction. To choose the best chip to extract, we divide the original facade image into a set of N tiles each of size 6x6 meters. Each chip is formed by selecting a tile as the center and then varying the chip size to 6, 12, or 18 meters and varying the aspect ratio (e.g., 1:1, 1:2, or 2:1). In total, $9N$ different candidate chips are produced for each facade. Please see Figure 5.4 for a visual depiction. We evaluate each chip by passing it through our rejection network and evaluating its rejection score. The chip with the lowest rejection score

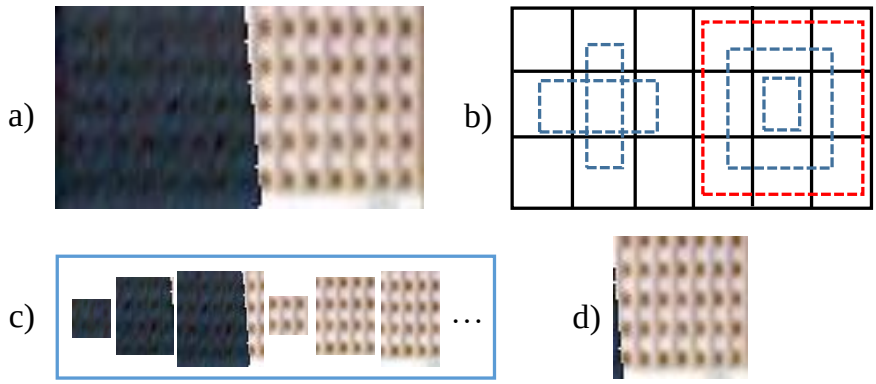


Figure 5.4. *Chip Extraction.* a) Original facade. b) Division of a) into tiles and demonstration of how chips are formed. c) Apply b) to a). d) The best chip.

is considered to be the cleanest chip found for the facade, and is selected to represent this facade further in the pipeline.

Segmentation. During segmentation, we only label each pixel as belonging to window/-door or non-window/non-door since other facade classes are usually not visible in satellite imagery. During development, we experimented with several state-of-the-art deep-network based semantic segmentation models (e.g., DeepLabv3+ [42], EncNet [41], and Pix2Pix [40]). Please see Segmentation models in the Results section for quantitative and qualitative comparisons among these architectures. We found that the architecture of Pix2Pix [40] performs among the best ones, and in particular we specify the generator architecture to consist of ResNet blocks, the discriminator architecture to be 34x34 PatchGAN, and the input image size to be 96x96. We train the segmentation network from scratch using our own manually created satellite facade dataset. Specifically, we train with 120 facade images (960 after applying the aforementioned data augmentation) along with ground truth from our dataset.

After segmentation, we have binary segmented chip facades with two labels: one representing windows and doors (black), and one representing the building wall (white). Using a binary representation eases the burden for deep-network based recognition and parameter estimation. In addition, we apply some image processing techniques to further refine the segmented image. First we perform a small amount of dilation (e.g. rectangular dilation with a kernel size of 3 pixels) to reduce some of the noisy black window/door pixels. Next,

since some facades are not perfectly rectified (due to errors in image registration and/or geometry), we perform a global image rotation computed automatically to force rows of windows/doors to be horizontal. Further, each window/door is replaced by a filled-in version of its rectangular bounding box. The end result is a binary image with rectangular windows and doors representing the facade, and serves as the input to our recognition and estimation networks.

Grammar Classification and Estimation. We represent a synthetic facade by one of six possible grammars each with a number of parameters, defined in a systematic fashion. While a single grammar with many parameters might be able to express more facades we found its generality to result in overall lower quality given the low-resolution nature of our facade imagery. For our grammar classification, a facade may contain doors and windows, or only windows. Further, the windows can be arranged as a grid of disjoint windows, as columns of vertically abutting windows, or as rows of horizontally abutting windows (see Figure 1.2 and Figure 5.5). Since window shapes are hard to differentiate with satellite data, we treat all windows as rectangles.

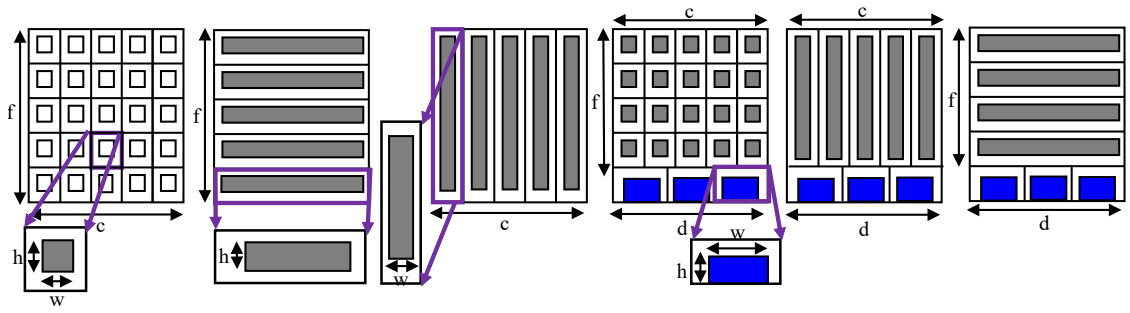


Figure 5.5. *Grammars.* Our grammars of (1-3) three styles of only windows and (4-6) three styles with doors at the base. "f" stands for the number of floors. "c" is the number of column boundaries. "d" is the number of doors. "h" is the relative height and "w" is the relative width. Please see the close-ups for additional parameters in the different grammars.

Which grammar a facade belongs to, along with the parameters for said grammar, is determined with a set of deep networks based on ResNet [121]. There is a classification network, which determines the grammar, followed by six parameter estimation networks, for determining the parameters specific to each grammar. The classification network is a

ResNet [121] with modification of the last fully-connected layer to the number of grammars. The final output layer of this network yields confidence values for each of the aforementioned grammars. After classifying a facade via this network, the segmented facade chip is then sent through the parameter estimation network that corresponds to the highest confidence value in the classification output.

To robustly find the procedural parameters for the classified grammar, we use a separate deep network for each individual grammar, all of which are also based on ResNet [121]. They differ only in the last fully-connected layer, where we modify the number of parameters to match that of the grammar. We also use mean squared error as the loss function for our estimation networks. The predicted parameters (e.g., window rows, columns, relative size, etc.) altogether yield a synthetic facade that is similar to the input image.

To train the estimation networks by systematically iterating over possible facade parameter configurations, we synthesized 200,000, 20,000, 20,000, 400,000, 50,000, and 50,000 facades from grammars 1 to 6 in Figure 5.5, respectively, based on the different number of parameters for each. We also perform data augmentation accounting for noise and errors in the segmentation (i.e., up to 10% noise such as perturbation of boundaries in windows/doors) and randomly remove up to 10% of windows/doors. To train the classification network, we collected 108,000 images in total from the aforementioned training images, distributed evenly among all six grammars.

Optimization. After recognition and parameter estimation, we perform a coarse-to-fine refinement for each chip. Segmentation suffers from noise, shadows, trees, and occlusions. Fortunately, our parameter estimation network is able to recover a procedural facade that

fills-in occluded content though there might be an overall translation or scale error. Thus, we define an objective function, using F1 score [135], as:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{ALL} \\ Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F1 &= 2 * \frac{Precision * Recall}{Precision + Recall} \\ P^* &= \underset{P}{\text{argmax}} F1 \end{aligned} \tag{5.1}$$

with true positives TP, false positives FP, true negatives TN, and false negatives FN for windows/doors and non-windows/non-doors.

In the above, P stands for the grammar parameters in Figure 5.5, P^* is the optimal parameter set.

Our optimizer tries to maximize this function using Monte Carlo stochastic optimization (e.g. altering P such as the number of floors, windows and window size) so as to create a synthetic facade that improves the $F1$ score with respect to the segmentation result. Please see Optimization in Results section for details and comparisons.

5.2.3 Completion

In a third and final stage, our method applies the estimated procedural parameters to all facades and generates windows and doors with the estimated sizes and spacing. Although the prior step determined parameters for rectangular chips, the actual facades on the buildings are not limited to rectangles but instead may have irregular shapes. To this end, we logically divide a building facade into a set of horizontally-adjacent rectangular sections. Since doors only appear at the bottom of a facade, we partition each rectangular section, that touches ground level, into two subsections: a door subsection extending from the bottom of the facade up to the door height, and a window subsection covering the remainder. Doors are placed horizontally-centered in the door subsections and sized according to the estimated

parameters. The window subsections are then further subdivided into window cells, also sized and spaced according to the estimated parameters, with one window placed into each cell. The tallest window subsections determine vertical window placement such that building floors are level across all sections.

Since each chip's parameters are estimated independently, neighboring facades will in general have different door/window sizes and spacing, and potentially different grammars. To remedy this issue, we first group facades together based on similar heights. All facades within each group are then forced to use the grammar of the highest scoring facade in the group, scored according to the grammar classification confidence value from the previous stage, with parameter values averaged over matching grammars in the group.

The resulting facades have windows and doors, which are colored according to the average window/door color as determined by the segmentation. Similarly, the facade wall is colored according to the average non-window color.

For evaluation details, we refer you to Section 7.3.

6. OTHER APPLICATIONS

We have discussed the applications of our framework to reconstruct different scales of urban structures (e.g., cities, buildings and facades). Theoretically, our framework can be generalized to other urban structures and applications. In the following sections, we present two different applications: (1) enhancing segmentations of urban structures, and (2) footprint completion of archaeological sites.

6.1 Enhancing Urban Segmentation

A version of this section is pending publication in Computer Vision and Image Understanding, 2022.

Image segmentation is a fundamental task that has benefited from recent advances in machine learning. One type of segmentation, of particular interest to computer vision, is that of urban segmentation. Although recent solutions have leveraged on deep neural networks, approaches usually do not consider regularities appearing in facade structures (e.g., windows are often in groups of similar alignment, size, or spacing patterns) as well as additional urban structures such as building footprints and roofs. Moreover, both satellite and street-view images are often noisy and occluded, thus getting the complete structure segmentation from a partial observation is difficult. Our key observations are that facades and other urban structures exhibit regular structures, and additional views are often available. In this paper, we present a novel framework (**RFCNet**) that consists of three modules to achieve multiple goals. Specifically, we propose **Regularization** to improve the regularities given an initial segmentation, **Fusion** that fuses multiple views of the segmentation, and **Completion** that can infer the complete structure if necessary. Experimental results show that our method outperforms previous state-of-the-art methods quantitatively and qualitatively for multiple facade datasets. Furthermore, by applying our framework to other urban structures (e.g., building footprints and roofs), we demonstrate our approach can be generalized to various pattern types.

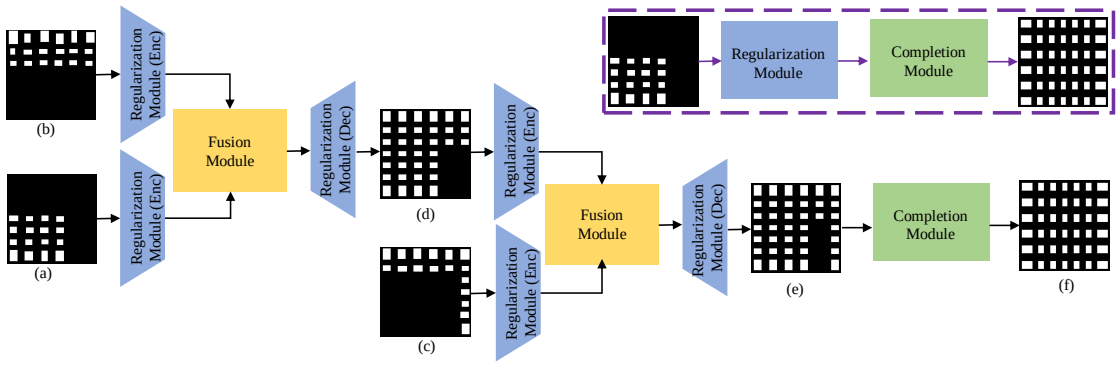


Figure 6.1. *RFCNet.* Two scenarios are shown. For single image facade segmentation (dashed box), it will be processed by Regularization and Completion. For multi-view facade segmentation, Fusion will combine the pairwise latent vector of inputs. (a-c) Multi-view facade segmented images. (d-e) Intermediate fused facade images. (f) Final synthetic output of RFCNet.

In this following, we describe our characterization of pattern regularity and generation of pattern styles for training, present the overall architecture of our RFCNet, and then detail each module of our architecture. Finally, we describe RFCnet implementation.

6.1.1 Related Work

Our work builds on procedural and inverse procedural modeling (see Chapter 1), facade segmentation, image fusion and image completion.

Image Fusion. It merges information from multiple images of the same scene taken from various sensors at different positions and/or different times, hopefully collecting complementary information. However, most recent approaches (e.g., [136]–[141]) focus on multi-modal fusion (i.e., combining information from different domains). In our case, we are focusing on fusing same-domain images from different viewpoints. Our fusion is more similar to [142] which aggregates multiple volumetric latent representations of the same object, and then applies a simple channel-wise averaging operation to obtain a fused representation. A problem with their basic averaging strategy is that feature maps are fused together without measuring the usefulness of each feature vector; hence, useless and useful features might be mixed together.

Image Completion. Filling-in missing pixels of an image, often referred to as image completion or image in-painting, is an important task. Deep learning and GAN-based approaches (e.g., [133], [134], [143]–[145]) have achieved promising results in this task. Compared to real-image completion, segmented-image completion is more challenging due to the lack of color and contextual information. We show comparisons to these approaches in the result section.

6.1.2 Pattern Regularity and Styles

In the following, we describe our assumptions about the pattern structural regularity and styles. We focus on the case of facades to illustrate the details of our method; details for other patterns are defined in an analogous way.

Structural Regularity. We characterize the structural regularity of facades by the arrangement of their features. *Windows and doors* are the predominant features visible in both street-level and satellite-based facade segmentation. Nonetheless, we also support additional labels for street-level observations (e.g., balconies). The placement of windows/doors can be described by their *alignment* (A), *size* (S) and *spacing* (P) as shown in Figure 6.2. Without loss of generality, a window b_i is defined by rectangle $\{x_\mathrm{i}, y_\mathrm{i}, w_\mathrm{i}, h_\mathrm{i}\}$ where $(x_\mathrm{i}, y_\mathrm{i})$ is its top-left corner and $(w_\mathrm{i}, h_\mathrm{i})$ is its size (S). A group of windows is left aligned (A_l) when the x coordinates of the top-left corners are equal. Right (A_r), top (A_t) and bottom (A_b) alignments are defined similarly. The horizontal spacing (P_h) between two horizontally adjacent windows (b_j, b_k) is defined by $x_k - (x_\mathrm{j} + w_\mathrm{j})$ (assuming b_k is at the right of b_j). Thus a group of windows has the same horizontal spacing when the computed horizontal spacing among those windows is equal. The vertical spacing (P_v) is defined analogously.

Style Generation. Within a facade there can be one or more groups of windows/doors exhibiting different combinations of the aforementioned structural characteristics $A_l/A_r/A_t/A_b$, S, and P_h/P_v. A particular combination of characteristics defines a *facade style*. For example, the facade style (a) in Figure 6.3 is based on the combination of constraints: A_l and A_r for each column of windows, A_t and A_b for each floor, windows of same size S, and same P_h and P_v spacing. (b) in the same figure differs from (a) by having more than one group of

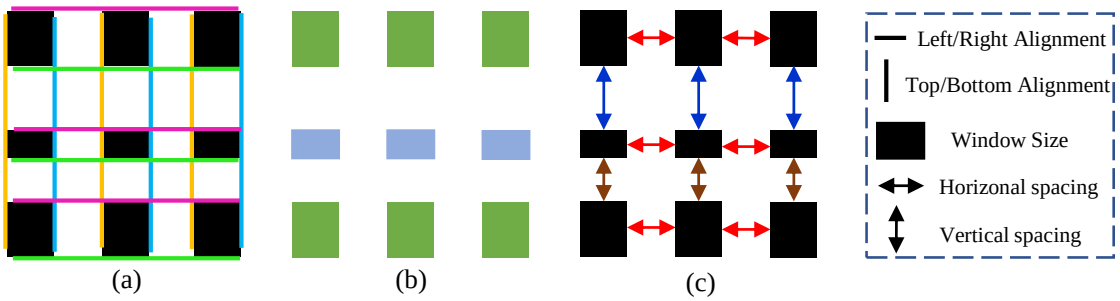


Figure 6.2. *Illustration of Structural Regularity.* (a) Left, right, top and bottom alignments are in different colors. (b) Different window sizes are in different colors. (c) Horizontal and vertical spacing are in different colors.

windows of the same S. The facade style (c) differs from (a) by having 2 groups of P_h in the facade.

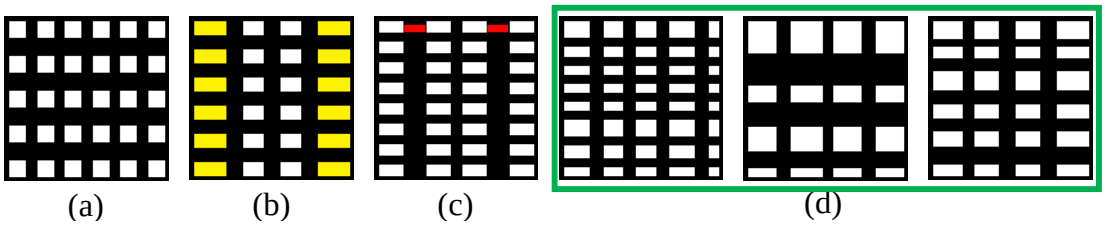


Figure 6.3. *Example Facade Styles.* (a-d) show progressively more general facade styles, with (d) being supported by our approach.

In the specific case of facade segmentation, prior work assumes specific structural constraints. For instance, our previous facade work addresses facade segmentation and completion but only for facades of style (a). [33] defines a specific set of 16 synthetic facade styles, similar to styles (a-c). For each proposed facade style, they generated enough synthetic training images to train a set of deep networks. For published facade datasets like CMP [146] or ECP [147], the supported facades are limited to the styles observed in the cities where the facades were collected. Further, they assume facades are captured at sufficiently high resolution and completeness to observe the supported stylistic details. Training based on those datasets will fail when handling other facade styles or satellite facade images.

Unlike the aforementioned methods, our approach is based on a much more general set of assumptions. For facades, our approach works as long as the facade satisfies the near

minimal constraints of basic alignment (i.e., A_l and A_r for groups of columns of windows, A_t and A_b for groups of floors as seen in real facade images, and groups with same size S and groups with similar spacing P_h and/or P_v. During training, we will generate a very large number of training examples that essentially will include all the prior sets of specific styles as well as other more general facade styles like (d) of Figure 6.3. To show the facade generality of our set of assumptions, we test several models against the styles in Figure 6.3.

6.1.3 Architecture

Our RFCNet, as illustrated in Figure 6.1, contains three sequential modules: Regularization module (R), Fusion module (F), and Completion module (C). RFCNet takes as input a segmentation from one or more viewpoints. For a single viewpoint, the input segmentation passes through Regularization and Completion. For the multi-view scenario, inputs will be successively combined by Fusion in pairs. In both cases, the output is a well-regularized, crisp and complete synthetic structure. Moreover, the whole network is trained in an end-to-end manner.

Regularization. The design of our Regularization module is mainly based on a Convolutional Autoencoder **Jonathan**. Our Regularization consists of two main blocks as shown at the left of Figure 6.4: an encoder part E^R and a decoder part D^R. To be specific, E^R takes the raw (un-regularized) segmentation $I^R \in \mathbb{R}^{H\times W\times C}$ as input, and first passes through a spatial transform network (STN) [148] which predicts a global affine transformation T to align the facade I^R to $I^R_t = T(I^R)$. I^R_t will be downsampled by a series of 2D convolutions layers into a lower dimensional latent representation $Z^R \in \mathbb{R}^{H'\times W'\times C'}$ that contains the informative content of the facade. D^R is trained to up-sample and reconstruct a regularized and synthetic output $\tilde{I}^R \in \mathbb{R}^{H\times W\times C}$ from Z^R. Each layer is either a convolutional layer (by default) or a residual block [121]. We also add an attention module CBAM [149] to bias allocation of the most informative feature expressions and simultaneously suppress the less useful ones.

Fusion. In order to fuse multiple facade segmentations $\{..., I^R_j, I^R_k, ...\}$, our Fusion module takes a pair of encoded latent representations $(Z^R_j, Z^R_k) \in \mathbb{R}^{H'\times W'\times C'}$ from E^R as inputs

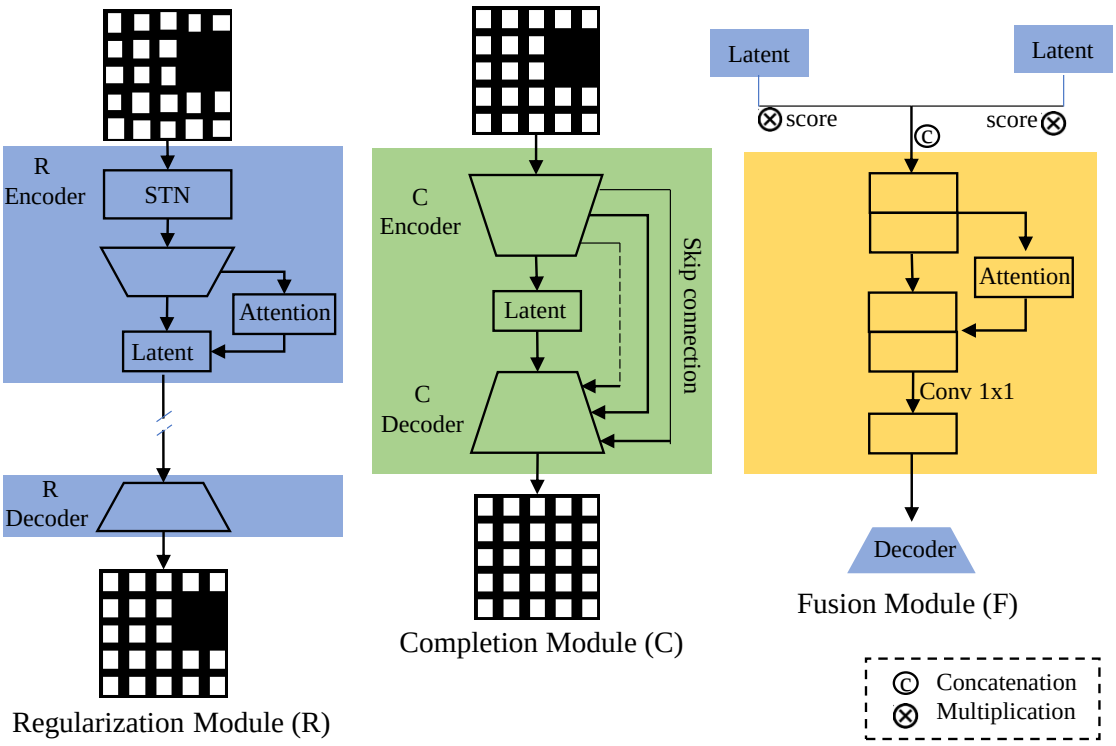


Figure 6.4. *Modules.* Structural details of Regularization, Completion and Fusion modules.

each time and generates an accumulated representation Z^F_{jk}. Then Z^F_{jk} passes through D^R for subsequent processing. The Fusion module is shown at the right of Figure 6.4. Our Fusion naturally extends to an arbitrary number of inputs in this way (e.g., an example of 3 inputs shown in Figure 6.1). Unlike the work [142] using basic averaging-based fusion (or, max, min, or addition), our proposed Fusion not only incorporates all those basic fusion strategies, but also learns the weights of how to fuse. In the beginning, (Z^R_j, Z^R_k) are scaled by their confidence values (or score) (w^F_j, w^F_k), and then they are concatenated together to form a deeper representation $Z^F_c = concat(w^F_j \times Z^R_j, w^F_k \times Z^R_k)$ and $Z^F_c \in \mathbb{R}^{H' \times W' \times 2C'}$. The confidence value corresponds to the quality of the input facade (e.g., the user may provide one of 1.0, 0.75 or 0.5 that correspond to high, medium and low quality images respectively). This confidence value especially helps when we use satellite based segmentation. Next, the concatenated representation Z^F_c is compressed by passing through a 1×1 convolutional layer to reduce the depth of channels to the original size C'. An attention module CBAM is also added to measure the usefulness of each feature vector, and filter out less important features.

Completion. Our Completion module takes the partially viewed and well-regularized segmentation $I^C \in \mathbb{R}^{H \times W \times C}$ as input, and then generates a well-regularized and complete synthetic output $\tilde{I}^C \in \mathbb{R}^{H \times W \times C}$ as shown in the middle of Figure 6.4. The Completion architecture is similar to Regularization but with several notable differences. STN is not necessary since the input has been aligned during Regularization. Since completion is more about propagating feature information globally, the bottleneck latent representation is fully connected to the previous layer. Skip connections [36] are added between the layer in the encoder and the corresponding layer in the decoder.

We rigorously and qualitatively evaluate our outputs and compare to state-of-the-art methods across multiple datasets. Please see the evaluations in Section 7.4.1.

6.2 Building Contour Completion

A version of this section is pending publication in Computer Graphics International, 2022.

Image/sketch completion is a core task that addresses the problem of completing the missing regions of an image/sketch with realistic and semantically consistent content. We address one type of completion which is producing a tentative completion of an aerial view of the remnants of a building structure. The inference process may start with as little as 10% of the structure and thus is fundamentally pluralistic (e.g., multiple completions are possible). We present a novel pluralistic building contour completion framework. A feature suggestion component uses an entropy-based model to request information from the user for the next most informative location in the image. Then, an image completion component trained using self-supervision and procedurally-generated content produces a partial or full completion. In our synthetic and real-world experiments for archaeological sites in Turkey, with up to only 4 iterations, we complete building footprints having only 10-15% of the ancient structure initially visible. We also compare to various state-of-the-art methods and show our superior quantitative/qualitative performance. While we show results for archaeology, we anticipate our method can be used for restoring highly incomplete historical sketches and for modern day urban reconstruction despite occlusions.

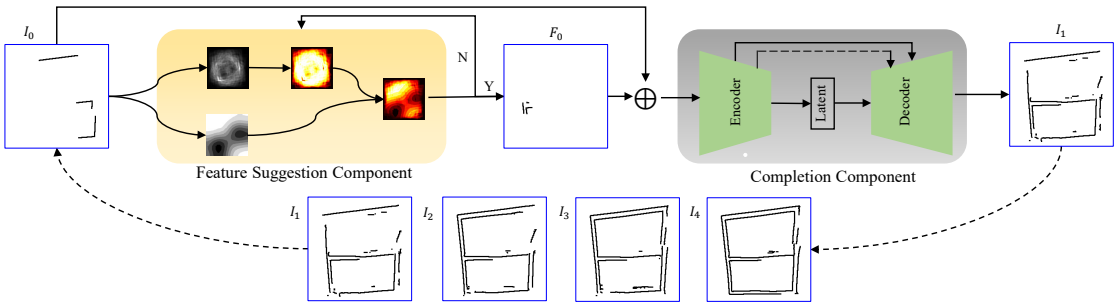


Figure 6.5. *Pipeline.* Iterates over a Feature Suggestion Component and a Completion Component.

6.2.1 Related Work

Sketch and contour completion works attempt to complete images containing lines/contours (e.g., black ink on a white background). The presence of structured content lacking color and texture introduces additional challenges. SketchGAN [150] uses a cascade Encode-Decoder network to complete the input sketch iteratively, and employs an auxiliary sketch recognition task to recognize the completed sketch. However, its incomplete input sketch is already 60% $\sim$ 90% complete, there are no large missing parts, and the sketch is most likely targeted to only one complete result. Ghosh et al. [151] propose interactive GAN-based sketch-to-image translation to generate full images given only sparse user strokes. However, it requires the user to choose the target object type. More recently, SketchBERT [152] and SketchHealer [153] perform the task by considering that a sketch is stored as a sequence of data points (e.g., vector format), rather than a photo-realistic image of pixels. ShadowDraw [154] is an earlier work that does not perform sketch completion per-se but rather simultaneously shows various potential more complete drawings to assist in drawing. The system is trained with many sketch-line drawings from a given set of categories. In general, these methods do not control pluralistic completions nor start with only a 10% complete sketch. Nonetheless, in our results section we compare to sketch completion.

We describe the main components of our approach. First, we describe our initial dataset (and motivating archaeological region), and our procedural generation method to generate

data for our self-supervised network training. Second, we describe our entropy-based feature suggestion component. Third, we describe our image completion component.

6.2.2 Procedural Data Generation

Our initial dataset is based on Bogsak Island in southern coastal Turkey which is an area of heavy archaeological investigation containing stone structures from the fourth to ninth century AD. It provides a prime location for us to explore building contour completion. Numerous similar other sites exist in the general region of Cilicia as well as other locations across the globe. Our investigator team includes archaeologists who have studied the site during the last decade [REF-omitted-for-anonymous-submission]. They are extremely excited about being able to "complete" the many partially preserved buildings in this site and then expand to other sites in order to better understand the past settlements. This dataset consists of aerial imagery spanning approximately 70,000 square meters at 5 cm per pixel. In these images, we detect the building walls using the state-of-the-art edge detection network DexiNed [155].

To enable our guided pluralistic building contour completion approach, we generate a large synthetic dataset of building contours spanning the observed style of the building structures in the general region. After inspecting the subset of buildings already studied in Bogsak (approximately 70 buildings), we classify them into three types of buildings: single, split, and T-shape. In our current system, we focus on building walls and leave the treatment of windows and door details for future work. The already studied buildings include archaeologist-inferred completions. As in urban procedural modeling, we define each style procedurally yielding random building variations (see Figure 6.6). Starting with the complete and synthetic building contour images, we then progressively mask-out (randomly picking either corners or wall edges) portions of each building producing 7 levels of incompleteness, with level 7 being the most incomplete (e.g., only 6% $\sim$ 13% of the structure remains) (Figure 6.7).

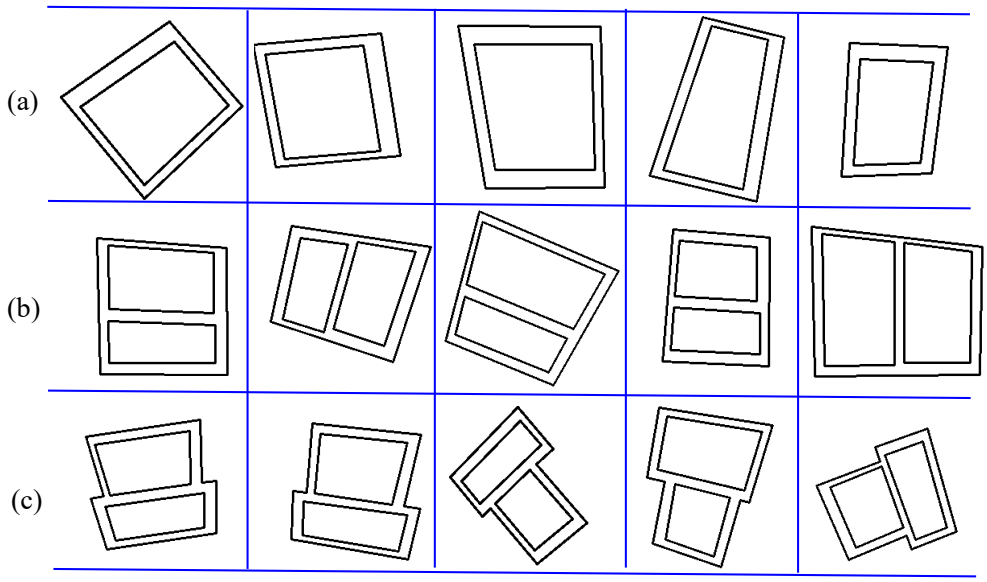


Figure 6.6. *Synthetic Dataset.* We show (a) single room, (b) split room, and (c) T room footprints used for training.

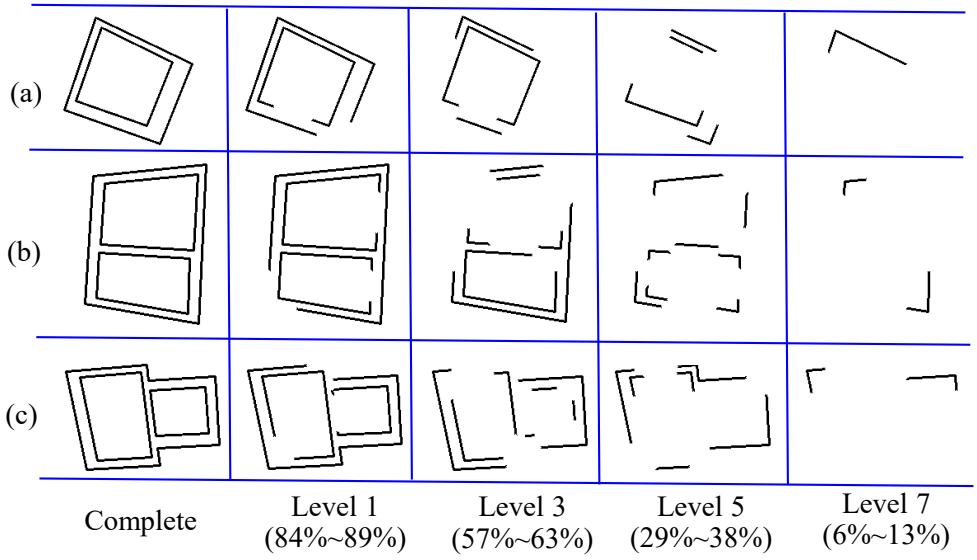


Figure 6.7. *Incompleteness Levels.* We show different levels of incompleteness for (a) single rooms, (b) split rooms, and (c) T rooms. **Note:** Percentage represents completion level.

6.2.3 Feature Suggestion Component

Given an incomplete building footprint (Figure 6.8a), our feature suggestion component iteratively provides the next best location where it would be most beneficial to have additional data. The additional data is an user-provided indication of the existence, or agreement, of there being more underlying structure: the user can "uncover" near said location "in the field" or sketch a small fragment of the believed building structure at, or near, the provided location.

To determine the location (x_0, y_0) that maximizes the information gain towards completion, we use a weighted information entropy model. Let the incomplete building footprint image be I_0 and the ground-truth complete image be I_C. For any (x, y) in the 2D image grid, $I_0(x, y) = 1$ if there is a building structure at (x, y); otherwise, $I_0(x, y) = 0$ – this is directly the output of edge detection. Next, consider the space of all complete buildings B. Given a complete footprint $I_\mathrm{i} \in B$, I_i is said to be consistent with I_0 if $I_\mathrm{i}(x, y) = 1$ for all (x, y) such that $I_0(x, y) = 1$. However, there are an infinite number of $I_\mathrm{i} \in B$ that are consistent with I_0, such as different building types, poses, and scales, and any of them can be a reasonable completion of I_0, thus I_C is not unique and instead there are a plurality of possible completions. We represent such uncertainty with information entropy given by

$$H(X) = \sum_{\mathrm{i}} p(I_\mathrm{i}) \log p(I_\mathrm{i}) \tag{6.1}$$

and is depicted in Figure 6.8 (c). During each iteration of the feature suggestion component, we provide information at a position (b_x, b_y) and the information gain G of such input is the difference between the original entropy and the conditional entropy after revealing such information

$$G = H(X) - H(X \mid (b_x, b_y)) \tag{6.2}$$

$$\begin{aligned} H(X \mid (x, y)) = {} & P(I(x, y) = 1) H(I_\mathrm{i} \mid I(x, y) = 1) \\ & + P(I(x, y) = 0) H(I_\mathrm{i} \mid I(x, y) = 0) \end{aligned} \tag{6.3}$$

Intuitively, the goal of the feature suggestion component is to identify the next 2D location (b_x, b_y) that maximizes the information gain so as to more quickly arrive at a complete footprint. The plurality of completions is addressed by the progressive identification of each (b_x, b_y) which gradually steers the completion process until only a single completion is possible, at which point the iterative suggestions are no longer needed.

A practical computational approach to the above is to use a large sample N_S of the possible complete footprints of different building types, poses, and scales, as described in Section 6.2.2. Given an incomplete footprint I_0, we measure the likelihood of a complete footprint I_i being a possible completion of I_0 using a masked $L2$ distance measure given by

$$d(I_0, I_\text{i}) = \sum_{(x,y)} \|I_0(x,y) - I_\text{i}(x,y)\|_2^2 \cdot I_0(x,y) \tag{6.4}$$

Hence, complete footprints with a smaller masked L2 distance are more likely to be a possible completion of I_0. By averaging the top N_F footprints we obtain P_0 (see Figure 6.8 (b)), where $P_0(x,y)$ approximates the probability of $I_C(x,y) = 1$. Therefore, at each iteration we search the 2D image and suggest the location (b_x, b_y) that maximizes the information gain for the subsequent completion component. Experimentally, we found that using $N_S = 3000$ random footprint samples and $N_F = 50$ top footprints yields a good trade-off between performance and accuracy, and is what we use for our reported results. Practically, we also estimate a distance field (see Figure 6.8 (d)) to avoid proposing feature locations near known structures in Figure 6.8 (a).

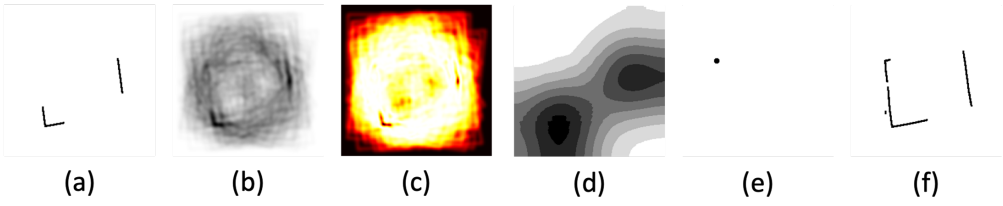


Figure 6.8. *Feature Suggestion Component.* (a) Incomplete footprint. (b) Average of the matched footprints. (c) Heatmap showing the uncertainty. (d) Distance field. (e) Proposed feature. (f) (Partially) completed footprint after one iteration.

6.2.4 Completion Component

Our completion component progressively produces an improved or completed building contour using the incomplete building image I_0 and a feature image F_0 having feature information at/near the location (b_x, b_y) provided by the feature suggestion component. However unlike typical completion tasks, the needed completion level of our footprint images ranges from a small missing portion to missing most of the footprint (e.g., 94% missing in layer 7). To accommodate this level of missing data, the design of our completion component considered the following three aspects.

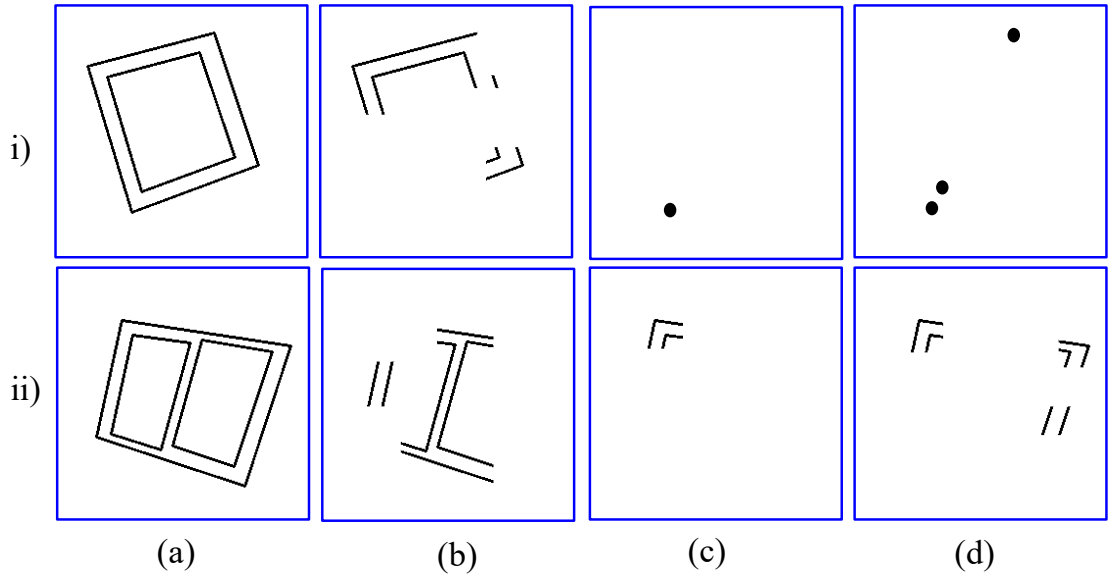


Figure 6.9. *Feature Suggestions.* We show (a) complete footprints, (b) incomplete footprints, (c) single feature images, and (d) multiple-feature images. For each, i) dot-style features and ii) line-style features are shown.

Single vs. Multiple Features. There are two fundamental methods for providing features (see Figure 7.30). A single-feature method accepts a building footprint image and only one feature in the feature image (e.g., Figure 6.9 (c)). This method is simpler in terms of training cases. But, completion error is accumulated throughout the feature suggestion iterations. A multiple-feature method can avoid error accumulation by always using the original incomplete image I_0 and adding up all previously proposed features $F_{\mathrm{i}} = \sum_{\mathrm{j}=0}^{\mathrm{i}-1} F_{\mathrm{j}}$

(e.g., Figure 6.9 (d)). However, it is much more difficult to train because of the exponential increase in number of training cases.

Feature Styles. We experimented with the performance of several feature styles and converged to two styles: dot-style and line-style (as shown in Figure 6.9 i) and ii)). Dot-style features represent corners in the footprint, and the presence of a dot in the feature image implies a missing building corner in the incomplete image at the given location (b_x, b_y). Line-style features are more informative because they illustrate both corner features and wall-edge features. The presence of small line segments in the feature image implies the presence of missing walls or corners (e.g., small L shape is for a corner, and short straight line segments are for missing walls.). We also experimented, for example, with using "thick lines" to represent walls (where the thickness of the line corresponds to observed thickness of the wall). We found this style to under-perform line-style so we did not pursue it any further.

Completion Level. Another design aspect is how much to complete, during training, a footprint given a feature image. Recall that an incomplete building fragment might support a plurality of completions. The goal is to find the balance between too aggressive completion causing ambiguity/noise/deterministic completion and too conservative completion resulting in many iterations. We performed several experiments using 25%, 50% and 100% completion to determine the best level (see Figure 7.31).

After experimenting with the aforementioned design considerations, we found multiple feature, line-type style, 50% completion to yield the best performance. Further, combining line-type features with multiple-features is actually equivalent to a single-feature style but at a higher-level of completion – in other words, we are seemingly doing multiple feature completion by thinking of it as a single feature completion using slightly more complete building footprints. Thus, the training time is very tractable. This configuration is extremely practical for our archaeology-setup because archaeologists can readily complete building footprints with only a few iterations of additional work. In general, we found by using our test data that 50% completion generates the best balance between number of suggestion iterations and image completion. In the results section, we showcase the effect of the aforementioned

design aspects. Additionally, We have comprehensive evaluates on our outputs and compare to state-of-the-art methods. Please see the evaluations in Section 7.4.2.

7. EXPERIMENTS AND RESULTS

In the following sections, we conduct experiments and demonstrate results for aforementioned applications of our framework. We quantitatively and qualitatively evaluate our approach on different scales of urban structures. Further, we compare our approach to several state-of-the-art methods and our approach consistently creates better results.

7.1 City Scale

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 2.4GHz XEON computer with several NVIDIA GTX TITAN XP boards each with 12GB 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 minutes 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 hours for the classifier network and 12-15 hours 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 hours to do a global optimization for 64 km^2 and 5-12 minutes 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 8x8 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.

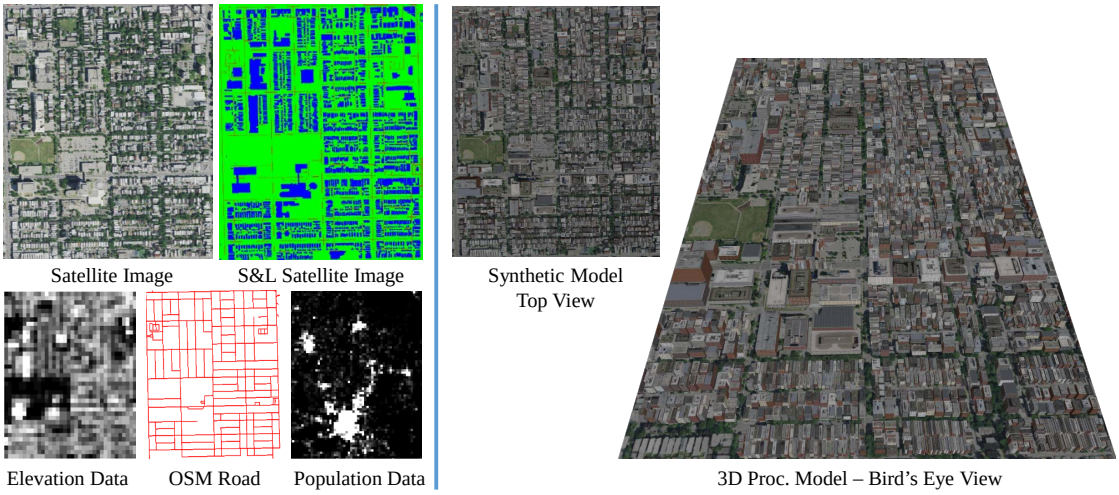


Figure 7.1. *Visual Pipeline.* Our method uses a satellite image and its segmented version, together with OSM, elevation and population data, to create a 3D procedural model.

7.1.1 Visual Results

Figures 7.1, A.1 and A.2 qualitatively show our results. First, Figure 7.1 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 A.1 and A.2 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.

7.1.2 Numerical Results

Figures 7.2 and 7.3, and 7.4 show numerically the improvement with various forms of our model optimization. Figure 7.2 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

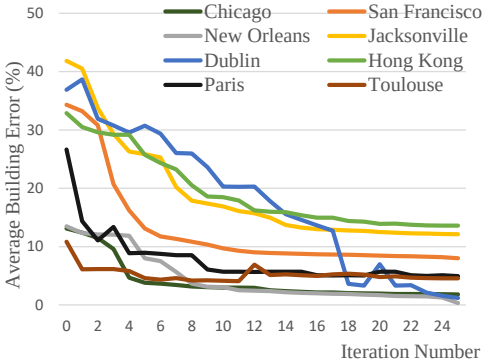


Figure 7.2. *Global Optimization.* The progressive reduction of our error function during optimization of the calibration parameters c_0, resulting in improved parcels and buildings.

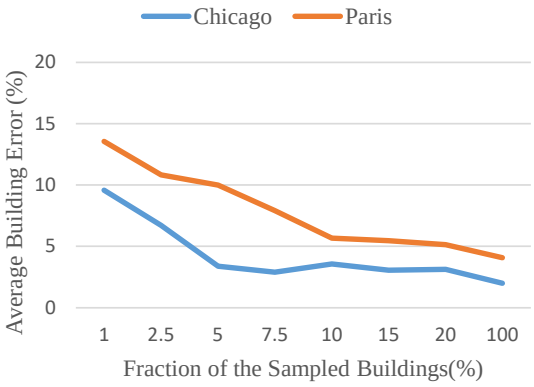


Figure 7.3. *Partial Knowledge of Average Building Footprint Area.* The graph shows how knowing the overall average building area for a random subset of the buildings affects the final building error after optimization. For Paris and Chicago, between 5 and 10% is a good trade off point.

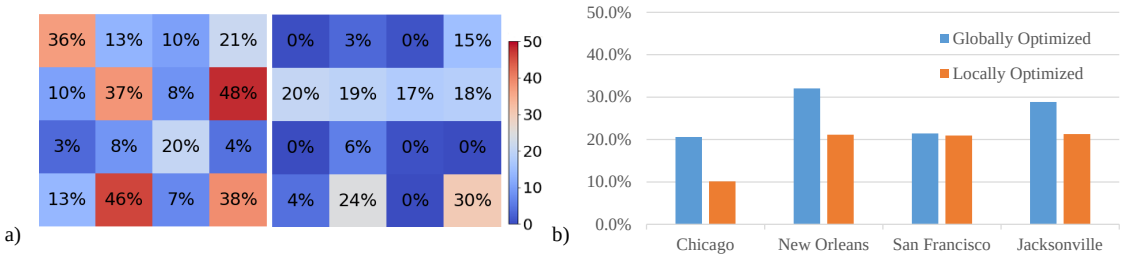


Figure 7.4. *Local Optimization.* Using a subset of our cities, we show how a local optimization (e.g., calibration parameters c_{2k}) further reduces our error function. a-left) Depicts the building error difference as a heatmap over the urban area (Chicago) using global optimization. a-right) Depicts 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.

building-area difference and relative building-count difference over the entire region (called *average building error*). The error reduction is also summarized in Table 7.1.

Table 7.1. *Global Optimization.* Initial vs Optimized Average Building Error.

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**

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 7.3 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.

Figure 7.4 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 7.4a 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 7.4 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 7.4 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.

7.1.3 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 7.5 and Table 7.2 compare distributions of building area using *Kolmogorov-Smirnov Testing*. Table 7.3 performs a *t-test* to show similarity of values over space.

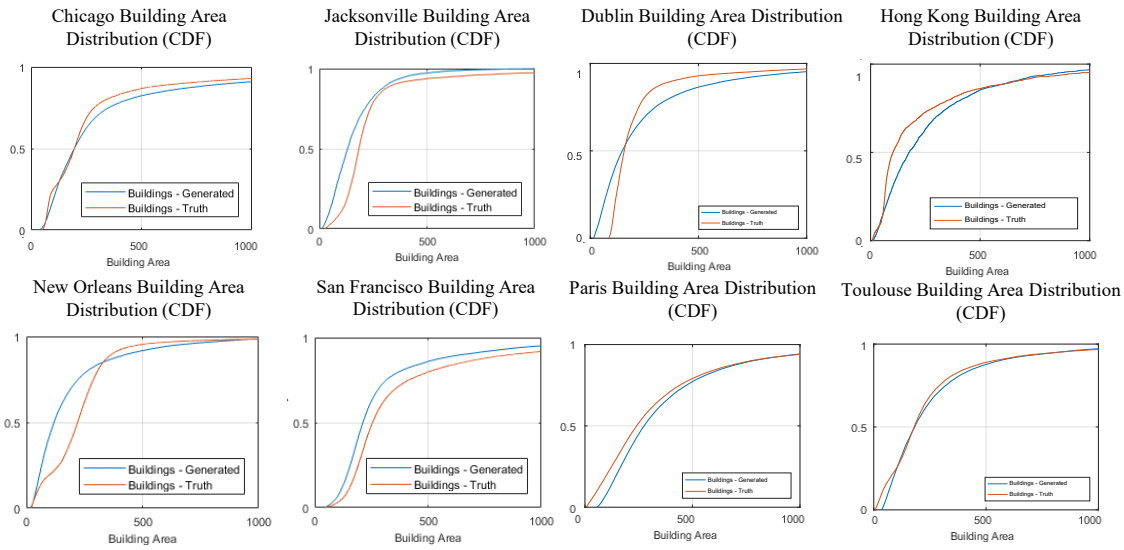


Figure 7.5. *Cumulative Distribution Function (CDF) Comparison.* Observe the similarity between the CDF for the synthetic models and for the ground truth of our cities.

Figure 7.5 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 different sizes are considered equal. Thus, we seek to have CDFs considered similar yet produced from using as many bins as possible.

Table 7.2 quantitatively measures distribution similarity. For example, at significance level $\alpha = 0.05$ (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 $26m^2$ for building areas ranging up to $2500m^2$.

Table 7.3 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

Table 7.2. *Building Area Distribution Similarity Test.* It shows 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.

	$\alpha = 0.05$		$\alpha = 0.01$	
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**

Table 7.3. *Building Number and Area Similarity Test.* 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.

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**

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.

7.1.4 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 7.4 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 7.4 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 7.6.

Table 7.4. *Segmentation Comparison.* 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%).

Chicago	Count	Area	Count Err	Area Err	Error
Truth	49461	452.8	–	–	–
Segm.	10387	1667.8	79%	268%	174%
Ours	**41765**	**460.2**	**16%**	**2%**	**8.6%**

Toulouse	Count	Area	Count Err	Area Err	Error
Truth	26045	315.8	–	–	–
Segm.	12392	469.1	52%	49%	51%
Ours	**22954**	**320.9**	12%	2%	**6.8%**

7.1.5 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).

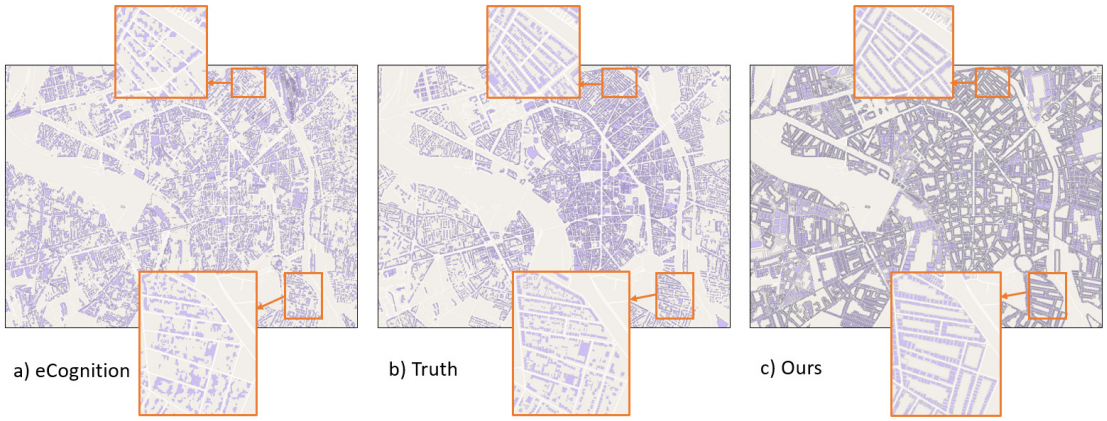


Figure 7.6. *Segmentation Comparison Images.* We compare the results of our approach to directly extruding the segmented and labelled satellite image provided as input. a) Segmentation and labeling result of using eCognition. b) Ground truth segmentaion and labeling. c) Our produced procedural model with labeling.

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 kilometers at about a 45^o downward looking viewing direction. At each distance, we created an image for 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.

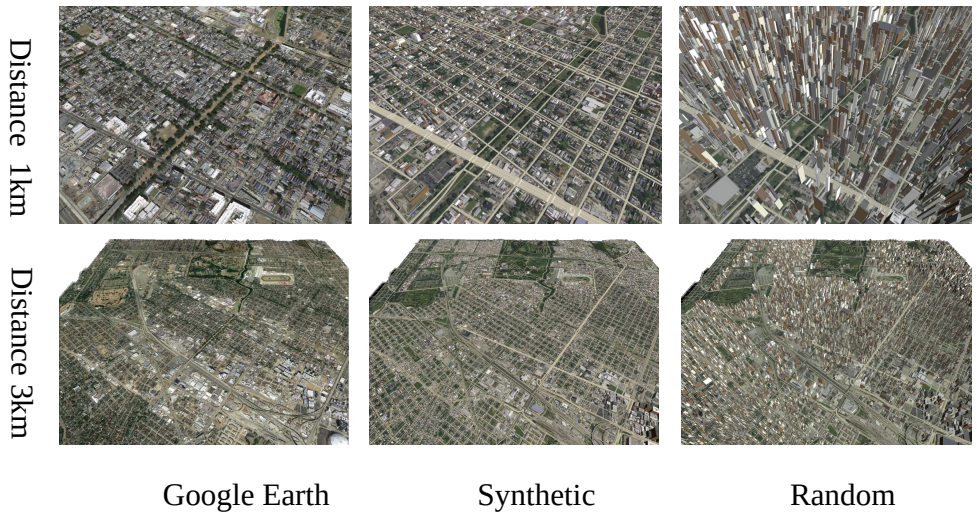


Figure 7.7. *Realism User Study.* First row is viewing from 1km distance. Second row is a distance 3km where our Synthetic image and Google Earth are considered almost equally realistic but the synthetic image with random building heights and areas is still notably less realistic.

Figures 7.7 and 7.8 show a summary of user study results. In Figure 7.8, 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 3 closest images of the random synthetic rendering to obtain a confidence value for each user (e.g., 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 7.7 for representative images at the different distances. Overall, from close-up users prefer Google Earth but from about 2.8 kilometers 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

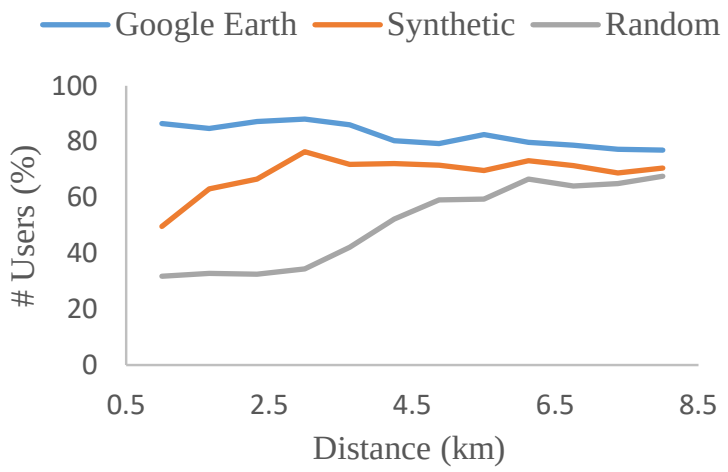


Figure 7.8. *Realism User Study Responses.* Based on worker responses observing images of Chicago, Dublin and New Orleans in random order, we show how realistic the different images are considered at increasingly farther distances.

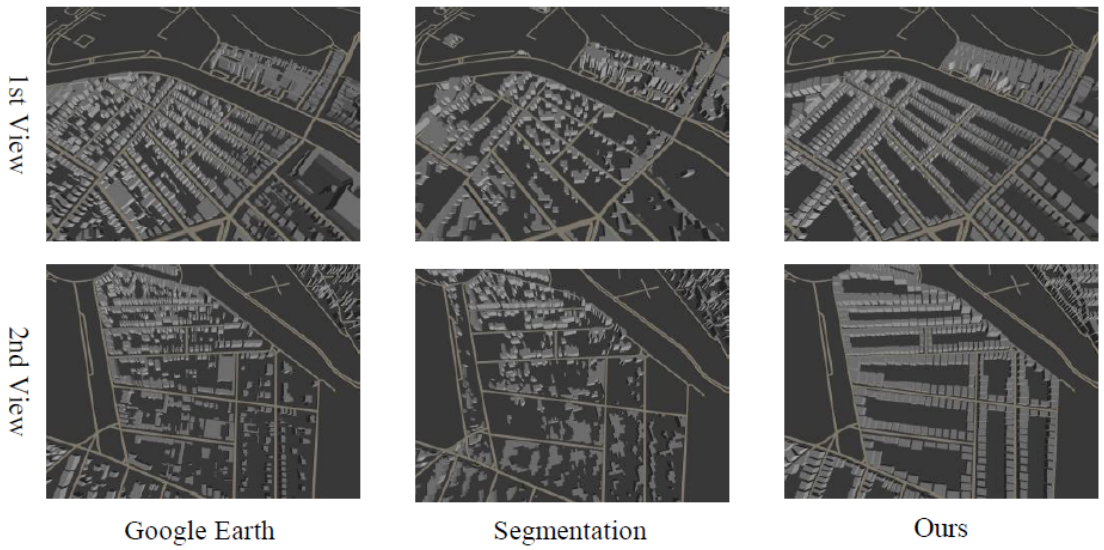


Figure 7.9. *Preference User Study Images.* We show two views of Google Earth images, extruded building-segmentation images and our synthetic images.

segmented satellite images (e.g., 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 7.6, created 6 image pair 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.

Figure 7.9 shows images from this user study and detailed responses of the user study are in Table 7.5. 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 (e.g., p-value = 0.064).

Table 7.5. Preference User Study Responses. 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.

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%

7.1.6 Application Examples

As applications, we show three tentative uses. Figure 7.10 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 minutes. Table 7.6 demonstrates an example computation of urban morphology values useful for urban planning/urban climate

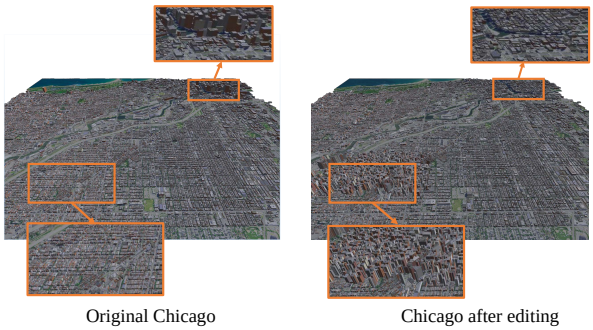


Figure 7.10. *Urban content generation.* This figure shows how we can quickly generate and edit plausible city-scale models based on real-world cities.

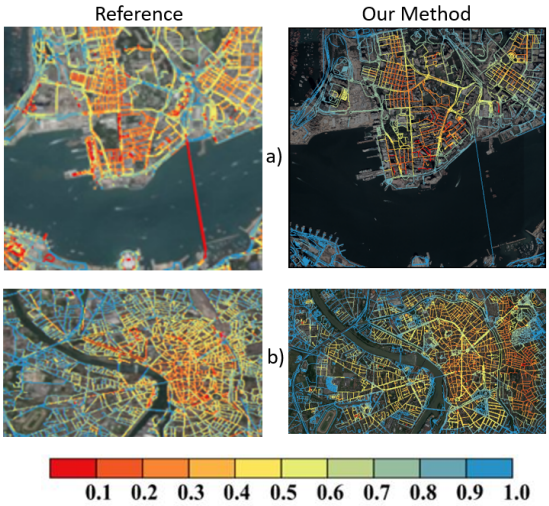


Figure 7.11. *Sky-View Factors.* We show for Hong Kong (top) and Toulouse (bottom) the similarity of our automatically computed sky-view factor to that obtained by the lengthy semi-automatic method of using [156] and Google Earth Street View images.

Table 7.6. *Urban Morphology Distribution Test.* Our ks-test shows that our two urban morphology values pass the test at significance levels α=0.05 and 0.01. For area weighted building height A_h the test passes with a granularity of at most $6.8m$ for α=0.05 and $4.1m$ for α=0.01. Similarly, for building surface to plan area ratio A_r, the test passes with a granularity of at most 0.18 for α=0.05 and 0.04 for α=0.01.

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

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,

we show that our computed urban morphology values yield a distribution of values that is statistically similar to ground truth at either α=0.05 and 0.01.

Figure 7.11 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 [156]. 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.

7.2 Building Scale

In the following sections, we conduct experiments and demonstrate results for both aforementioned methods.

7.2.1 Multi-view Satellite Images

Our method is implemented using OpenCV, OpenGL, and QtUrban, and it runs on an Intel i7 workstation with a NVIDIA GTX 1080. We have applied our method to two test areas in the United States captured by WorldView3 satellite images: a portion of (A1) Jacksonville, Florida (1.9 km^2) and (A2) UC San Diego, California (1 km^2). Collectively, the areas have a few hundred buildings and medium to tall buildings have from 20 to a few hundred windows/doors each. The 2.5D point cloud dataset we use was produced by an implementation based on [157], [158]. Our entire method runs automatically yielding 14 buildings per minute.

Regularization

Figure 7.12 shows examples of the progression of models through our pipeline including applying regularization. As you can see, the corner regularization will make the corner angles into typical angles (e.g., 90), the parallel regularization will make the almost parallel

wall segments parallel, the symmetry regularization will make the model hold the symmetry property, and the alignment regularization will remove misalignment between layers. In our experiments, we found the symmetry and curved-wall regularization to occur the least amongst these 5 metrics.

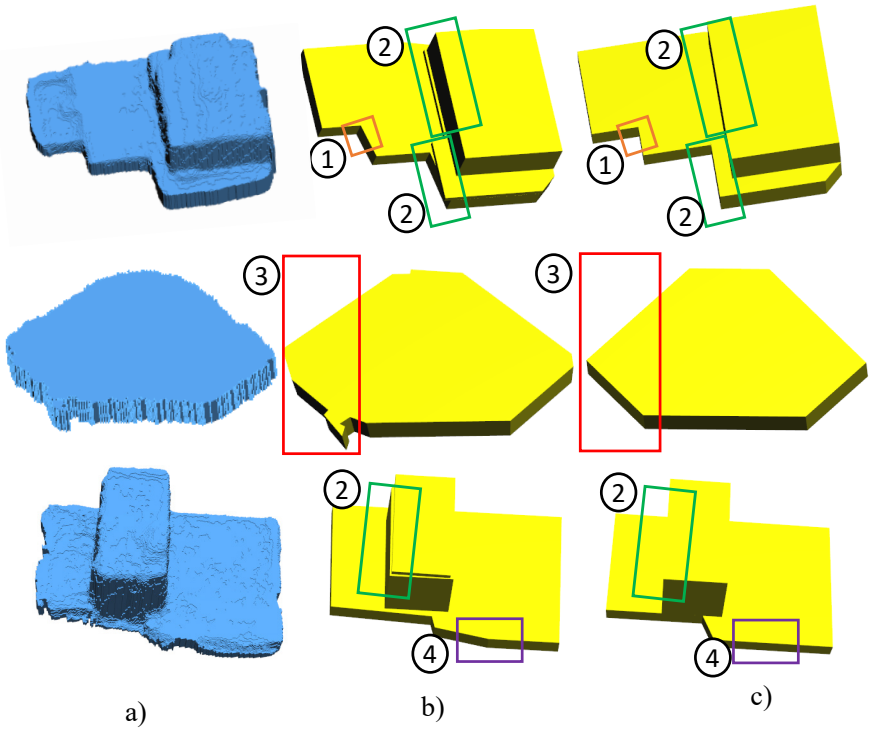


Figure 7.12. *Metrics.* We show examples of the metrics used in our regularization. a) Incoming satellite-based point cloud, b) the output models before regularization, and c) the output models after applying specific regularization terms. "1" is corresponding to the corner regularization. "2" is corresponding to the alignment regularization. "3" is corresponding to the symmetry regularization. "4" is corresponding to the parallel regularization.

Geometry Synthesis

Table 7.7 reports statistics about the synthesized building geometries. Table 7.8 contains the globally averaged accuracy of our produced buildings in terms of 2D and 3D completeness and correctness using an implementation of the testing method of [159]. Our models are compared against a manually-refined high-resolution aerial LIDAR capture of the test area. The overall accuracy is above 90%. Further, we compare our approach to a similar set of

prior methods as in the recent paper by [160]. In particular, we show in Figure 7.13 a visual comparison between Poisson surface reconstruction, dual contouring [95], Polyfit [94] and our method. We also compare to a surface simplification method QSlim [161] to demonstrate that general polygonal simplification does not maintain the expected geometric and architectural properties. Overall, our approach produces the best crisp and regularized models.

Table 7.7. *Building Complexity.* Average number of vertices, edges, and faces in buildings by our method.

Zone	#Vertices	#Edges	#Faces
A1	164.6	67.2	183
A2	224.24	90.93	248.5

Table 7.8. *Geometric Accuracy.* Accuracies for our areas in terms of the metric by [159].

Zone	2D Correctness	2D Completeness	3D Correctness	3D Completeness
A1	0.93	0.90	0.92	0.92
A2	0.95	0.80	0.92	0.83

[100] focused on building reconstruction from satellite-based point clouds (in fact, also Worldview 3 based). They developed neural networks to do semantic segmentation and then to find roof points. They extract roof shape primitives by applying RANSAC [90]. The final results are refined by the boundary and the continuity of the model. Nonetheless, as you can see in Figure 7.14 (copied with permission from their paper), our work essentially extends such an approach to further produce crisp and lightweight building models. Although these are not the same urban areas in the figure, the quality of our solution is notably cleaner and crisper. In addition, [100] report geometric accuracy for several areas also using [159]. Their average values for the same terms as in Table 7.8 are 0.905, 0.73, 0.895, 0.75. As seen, our approach is consistently more accurate by 7% on average which visually amounts to a significant spread.

Finally, we show in Figure B.1 many close-ups of reconstructed buildings from both areas, textured with projected satellite images.

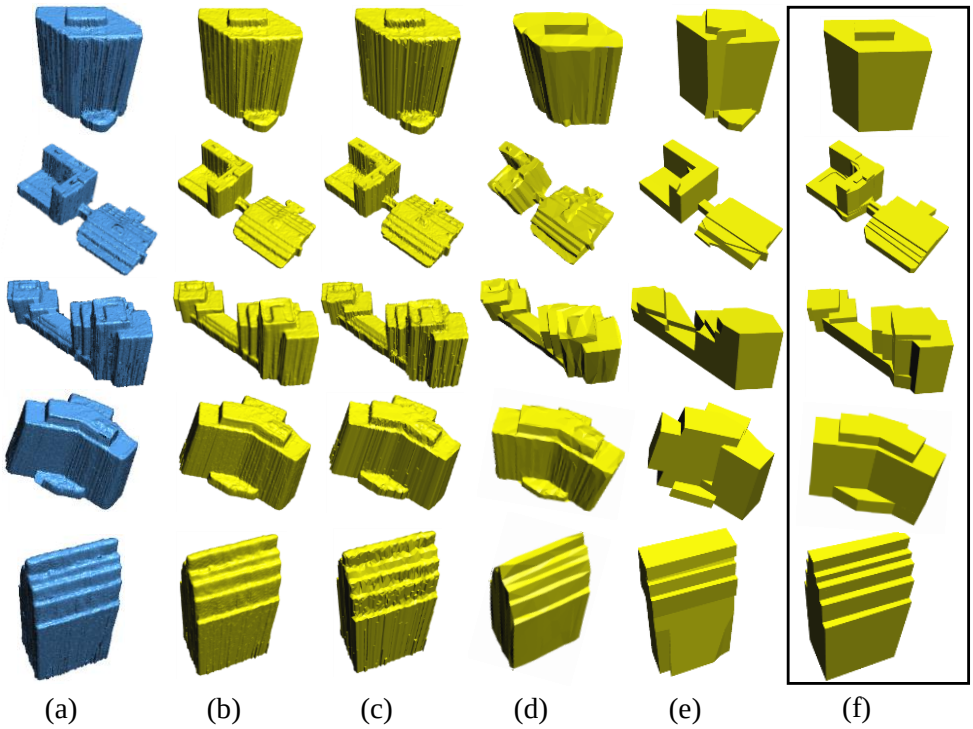


Figure 7.13. *Geometry Comparison.* a) Incoming satellite-based point cloud, b) Poisson surface reconstruction, c) 2.5D dual contouring, d) QSlim [161] of b), e) PolyFit, and f) Our method.

7.2.2 A Single Satellite Image

Our method is implemented in Python and we train our neural network models using PyTorch. The weights of our classifiers are trained by the SGD optimizer where initial learning rate is set to 1e-3. Our typical input image sizes are $(H, W, C) = (128, 128, 1)$. It runs on an Intel i9 workstation with NVIDIA RTX 2080 8GB cards. We quantitatively and qualitatively evaluate our approach on multiple satellite datasets.

Evaluation Metrics

We rigorously evaluate our outputs from both building decomposition component and roof ridge detection component. In particular, we evaluate both pixel-wise correctness and structure regularization for building footprint. For roof ridges, we assess the performance in terms of a hit ratio, completeness, and correctness metric.

Building Footprint

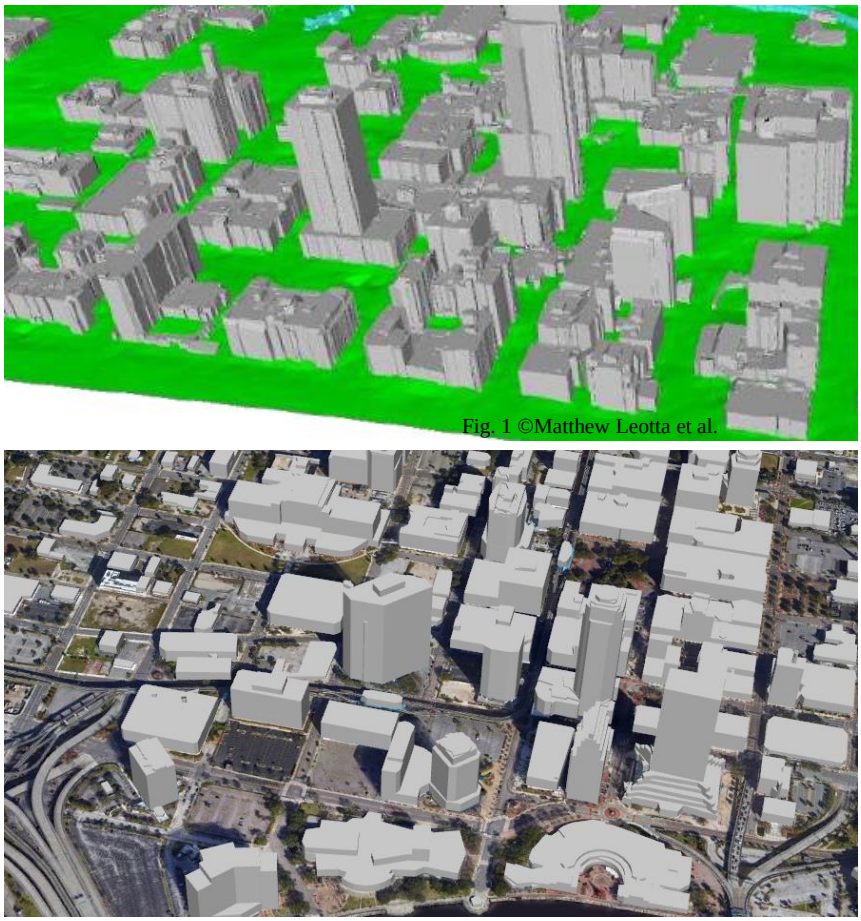


Figure 7.14. *Comparison.* The top row is a result image cropped from [100]. The bottom row is generated by our system.

For pixel-wise correctness evaluation, we use the following statistical measures: Accuracy, Precision, Recall and F1 (see 5.1).

For building footprint regularization evaluation, we follow the observation that building walls are typically parallel or meet at corners of predetermined angles (90, 45 or 135 degrees). Hence, for the polygonal outline of a building footprint, we compute the interior angles in degrees (within $[0, 180]$) for each vertex. Then, we cluster corners of similar angles into a group g_{i} and all groups form part of the set G. The regularization error of E_r is defined as:

$$E_r = \sum_{g_{\text{i}}} \frac{stdvar(g_{\text{i}})}{scale(g_{\text{i}})} + w_r \|G\|, \tag{7.1}$$

where $stdvar(g_{\mathrm{i}})$ measures the standard deviation of angles in g_{i}, $scale(g_{\mathrm{i}})$ is used to approximately normalize the error – we set $scale(g_{\mathrm{i}})$ to 5 in our experiments. $\|G\|$ is the number of corner groups. We add $\|G\|$ to encourage fewer and thus larger groups. w_r is a weight that balances the two aforementioned terms – we set $w_r = 0.1$ in our tests. It's easy to recognize that a rectangular building footprint whose walls are parallel and corners are all 90 degrees has $E_r = 0.1$ since the $stdvar(g_{\mathrm{i}}) = 0$ and $\|G\| = 1$.

Roof Ridges

With regard to roof ridge evaluation, we adapt and modify the relevant definitions of correctness and completeness from [162]. Correctness represents the percentage of the predicted roof ridge which lies within a rectangular buffer around the ground truth ridge. We set the buffer width to be 0.1 times the length of ground truth roof ridge (which results in a width of typically 2-4 pixels). Using a similar strategy, completeness is the percentage of the ground truth which lies within the buffer around the predicted ridge. Hence, we define correctness and completeness for roof ridges as follows:

$$Correctness = \frac{length\ of\ matched\ prediction}{length\ of\ prediction}, \tag{7.2}$$

$$Completeness = \frac{length\ of\ matched\ ground\ truth}{length\ of\ ground\ truth}, \tag{7.3}$$

In order to be consistent with footprint evaluation, we also evaluate ridges per entire building. Since a single building commonly contains more than one roof ridge, we apply weights to balance the importance of the multiple ridges based on their length. For easy illustration, we assume the building has a set of ridge lines $\{l_1, ..., l_{\mathrm{i}}, ...\}$ and the corresponding weight set is $\{w_1, ..., w_{\mathrm{i}}, ...\}$. We define:

$$w_{\mathrm{i}} = \begin{cases} \frac{\|l_{\mathrm{i}}\|}{\sum_{\mathrm{j}} \|l_{\mathrm{j}}\|} & \text{if i}_{th} \text{ ridge is found (predicted)} \\ 0 & \text{otherwise} \end{cases}$$

where $\|l_i\|$ is the length of the ridge l_i. "Found" means that both correctness and completeness of the ridge are bigger than a threshold (setting to 0.3 in our experiments). Hence, the correctness and completeness per building is

$$\begin{aligned} Correctness\ per\ building &= \sum_i w_i * correctness(l_i) \\ Completeness\ per\ building &= \sum_i w_i * completeness(l_i) \end{aligned} \tag{7.4}$$

Additionally, we define another term to represent how many of the ridges have been "hit" (or found) by our method. This term also considers the relative importance of each ridge, and thus is computed as a percentage by summing the aforementioned weights of the (found) ridges:

$$Hit\ Ratio = \sum_i \|w_i\|, \tag{7.5}$$

Building Footprint Comparison

Although our approach aims to generate procedural roofs, regularized and parameterized procedural building footprint are produced by our building decomposition component as an intermediate output. We compare this intermediate result to the outputs of other methods. We selected tiles at random but that are at least almost orthorectified. To be specific, we test on 7 random tiles of SpaceNet (which corresponds to 180 buildings), 10 random tiles of CrowdAI (resulting in another 65 buildings), and 1 random tile of Urban3D (producing another 415 buildings). We compare our generated building footprints to the initial segmentations of each dataset and a state-of-the-art building footprint delineation method ASIP [108]. The initial segmentation for each of the datasets is in the first row in each group of Table 7.9. For ASIP, we set $\beta = 10^{-3}$ and $\lambda = 10^{-5}$ as recommended by their paper and apply their tool to generate results in a polygon format.

As shown in Table 7.9, our method is slightly less in accuracy and precision, but always achieves better recall (meaning our results are more complete) and F1 score performance (Note: only F1 of CrowdAI is not the best) for all three datasets compared to the best model

Table 7.9. *Quantitative Comparison.* We compare our building footprints with the initial footprint segmentations and the ASIP method [108] for SpaceNet, CrowdAI and Urban3D datasets. For footprint correctness, higher is better. For regularization error, lower is better. **Note:** our E_r of our approach is 0.1.

Dataset	Method	Footprint Correctness				E_r
		Acc.	Pre.	Rec.	F1	
SpaceNet	Mask R-CNN	**89.5%**	**95.1%**	86.9%	90.4%	—
	ASIP	89.0%	94.6%	86.4%	90.0%	1.79
	Ours	89.0%	94.0%	**88.6%**	**90.8%**	**0.1**
CrowdAI	Mask R-CNN	**94.4%**	**92.3%**	89.8%	**90.8%**	—
	ASIP	93.9%	92.0%	88.7%	90.0%	1.52
	Ours	93.1%	88.9%	**91.9%**	90.2%	**0.1**
Urban3D	DeepLabv3+	**86.4%**	**81.0%**	85.3%	81.6%	—
	ASIP	85.8%	80.3%	84.2%	80.7%	1.60
	Ours	85.5%	79.4%	**88.1%**	**81.8%**	**0.1**

in terms of footprint correctness (e.g., for SpaceNet, our accuracy and precision is **0.5%** and **1.1%** lower, but our recall and F1 score is improved by **1.6%** and **0.4%**). Regarding regularization error E_r defined in Equation 7.1, we compare to the polygonal output of the ASIP method. For the initial segmentation (e.g., Mask R-CNN or DeepLabv3+) and the corresponding ground truth, the polygonal representations don't exist and simply computing the E_r term would provide a very large error for those methods. Nevertheless, it is obvious to recognize that there is no regularization for the segmentation (Figure 7.15 (c)), and the ground truth (Figure 7.15 (b)) is regularized and its E_r is close to 0.1. As clearly observed, our results significantly improve building footprint regularization (e.g., for SpaceNet, the regularization error is reduced by **94.4%**). Further, as illustrated in Figure 7.15, the outputs of our method are visually appealing as well.

Roof Ridge Comparison

We compare our procedural roofs to three methods which approximately perform the same task as us – these are the most similar works we could find that operate on a single image, though two of these use aerial images at 6 times higher resolution. Since annotations

Figure 7.15. *Qualitative Comparison.* (a) Real images. (b) Ground truth footprints. (c) Initial building footprint segmentations. (d) ASIP results. (e) Our results.

of roof ridges for our test datasets are not available, we manually create them by using an image annotator tool VIA [163]. We randomly chose 83, 21 and 26 buildings from SpaceNet, CrowdAI, and Urban3D, respectively, and annotated the roof ridges. We compare our predicted roof ridges to the state-of-the-art method Conv-MPN which predicts building edges [107] (we only evaluate the roof ridges in Conv-MPN for fairness). As shown in Table 7.10, our method consistently achieves better performance compared to Conv-MPN (e.g., hit ratio improved by **31.8%**, correctness improved by **27.7%**, and completeness improved by **45.4%** for SpaceNet). Yet more, as demonstrated in Figure 7.16, our results are qualitatively preferable.

In addition, we compare to the methods in [113], [114]. These approaches reconstruct roofs from a single aerial image at a 5 cm spatial resolution using the Potsdam dataset provided by [164]. We compare to these methods by first down-sampling the aerial image to

Table 7.10. *Quantitative Comparison.* We compare our predicted ridges with Conv-MPN method [107] for our SpaceNet, CrowdAI and Urban3D datasets. For all three metrics terms, higher is better. **Note:** since Conv-MPN is not trained on Urban3D originally, we only show our performance for this dataset.

Dataset	Method	Hit Ratio	Correctness	Completeness
SpaceNet	Conv-MPN	63.2%	58.7%	44.8%
	Ours	**95.0**%	**86.4**%	**90.2**%
CrowdAI	Conv-MPN	63.4%	61.6%	58.6%
	Ours	**96.3**%	**90.9**%	**92.9**%
Urban3D	**Ours**	87.9%	73.5%	81.0%

30 cm resolution (– the resolution of our tested satellite images –) and then apply our method. In addition, we also manually annotate roof ridges. In terms of hit ratio, correctness, and completeness, we obtain **97.9%**, **92.9%** and **96.8%** respectively.

However, [113], [114] use a different correctness and completeness term to evaluate their roof ridge and other urban structures. For [113], it outputs 43.4% completeness and 4% correctness for just roof ridges (their completeness and correctness is higher when you also consider the building footprint pixels). In [114], they improved results to 57.7% completeness and 81.3% correctness for roof ridges (using the same metrics as [113]). At satellite-level resolutions, the correctness and completeness term they provide does not seem suitable. Nonetheless, we did compute the values using their method and obtained 35.5% completeness and 34.3% correctness at 30 cms per pixel, as opposed to their values at 5 cms per pixel. While our terms are lower than [114], our method operates at 6 times lower resolution because we used satellite images.

More Results

We show our procedural urban generations for three large areas in Figure C.1. Moreover, since we have a procedural output (instead of an image), we can zoom-in to any part of the area and still have a high-quality result.

Figure 7.16. *Qualitative Comparison.* (a) Real images. (b) Ground truth ridge annotations. (c) Conv-MPN results. (d) Ours results. **Note:** The red lines in (c) are considered as roof ridges.

7.3 Facade Scale

Our method is implemented using OpenCV, OpenGL, and PyTorch, and it runs on an Intel i7 workstation with NVIDIA GTX 1080 cards. We have applied our method to six test areas in the United States captured by WorldView3 satellite images: a portion of (A1) Jacksonville, Florida (2.0 km^2), (A2) UC San Diego, California (1 km^2), (A3) San Fernando, California (1 km^2), (A4) Omaha, Nebraska (2.2 km^2), (A5) San Diego, California (1.2 km^2) and (A6) USC, California (2 km^2). Collectively, the areas have a few hundred buildings

and medium to tall buildings and have from 20 to a few hundred windows/doors each. Our method runs automatically yielding facades for 14 buildings per minute. The training time for our classification network is about 12 hours, and the training time for our estimation networks from grammars 1) to 6) is about 20 hours, 3 hours, 3 hours, 36 hours, 8 hours, and 8 hours, respectively.

7.3.1 Dataset

In order to train our neural network models, evaluate our method, and compare with other methods, we present a dataset of real satellite facades, which includes about 400 rectified images of facades from the aforementioned six areas, which have been manually annotated with two different labels: one for windows/doors and the other for the walls. Because of the low-quality of these facades, even humans can't precisely do the segmentation. Thus, mis-segmentation and misalignment always exist. Further, we carefully refine the annotations for 61 facade images and use those facades as a test data set for evaluating models/methods.

7.3.2 Pipeline Steps

We show example pipeline steps in Figure 7.17 which includes chip extraction results, segmentation results, image processing results and our final facade completion results.

Table 7.11. *Segmentation Quantitative Comparison.* Pixel Accuracy, precision, recall and F1 metrics evaluated on 61 facades for models from b) to g). Those terms are defined in Optimization Section.

Model	Accuracy	Precision	Recall	F1
b)	0.843665	0.756	0.747	0.742
c)	0.8482	0.795	0.712	0.742
d)	0.866343	0.836	0.741	0.771
e)	0.846425	0.802	0.696	0.732
f)	0.849911	0.776	0.725	0.740
g)	0.870966	0.864	0.709	0.766

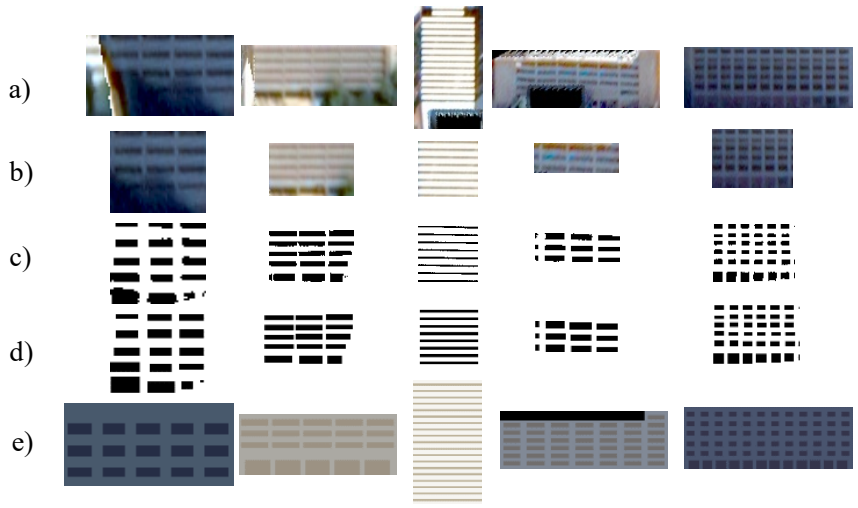


Figure 7.17. *Pipeline Steps.* a) Selected facade images. b) Facade chips. c) Results of using our segmentation model b). d) Images after applying dilation, rotation and replacement of windows/doors with filled-in rectangular bounding boxes and then being fed to our neural networks. e) Synthesized facades.

7.3.3 Segmentation models

We test satellite facade segmentation on three state-of-the-art neural network architectures: Pix2Pix [40], Deep Labv3+ [42] and EncNet [41]. We train these architectures from scratch using our data set and also customize the hyper-parameters to fit our segmentation problem. For Pix2Pix we also try different generator and discriminator architectures which could support different sizes of input images. Please see Table 7.11 for quantitative comparisons. Based on this comparison, we perceive Pix2Pix_96 to work best and it is the segmentation model we use in our approach.

Table 7.12. *Optimization Quantitative Comparison.* Pixel accuracy, precision, recall, F1 and blob accuracy evaluated on 61 facades for models c) and d) in Figure 7.18.

Method	Accuracy	Precision	Recall	F1	Blob
c)	0.725	0.556	0.673	0.597	0.810
d)	0.880	0.818	0.834	0.815	0.923

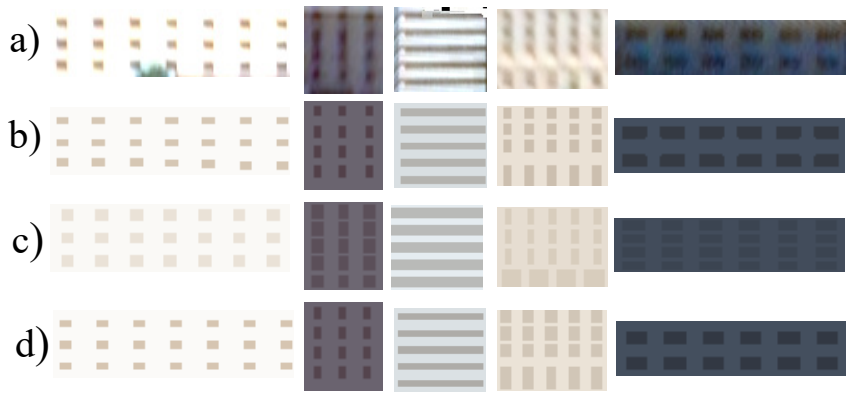


Figure 7.18. *Optimization Qualitative Results.* a) Original facades. b) Manually created ground truth. c) Our results without optimization. d) Our results with optimization.

7.3.4 Optimization

We evaluate 61 facade images using both our method without optimization and our method with optimization. Thus we show that we improve pixel accuracy, precision, recall, F1 and blob accuracy by perturbing grammar parameters. The blob accuracy is the window count accuracy defined as:

$$Blob = 1 - \frac{|Our\_Window\_Count - Ground\_Truth\_Window\_Count|}{Ground\_Truth\_Window\_Count}, \tag{7.6}$$

Please see Figure 7.18 and Table 7.12 for qualitative and quantitative comparisons. In summary, with optimization our metrics improve from 0.69 to 0.85, an improvement of 16% on average.

Table 7.13. *Facade Quantitative Comparison.* We evaluate Mean Absolute Error (MAE) and Mean Relative Error (MRE) of the number of floors and the number of windows per floor on 61 facades for c) and d) in Figure 7.19.

	MAE		MRE	
Method	#floors	#windows	#floors	#windows
c)	0.770	0.770	15.8%	12.1%
d)	0.246	0.164	4.2%	3.9%

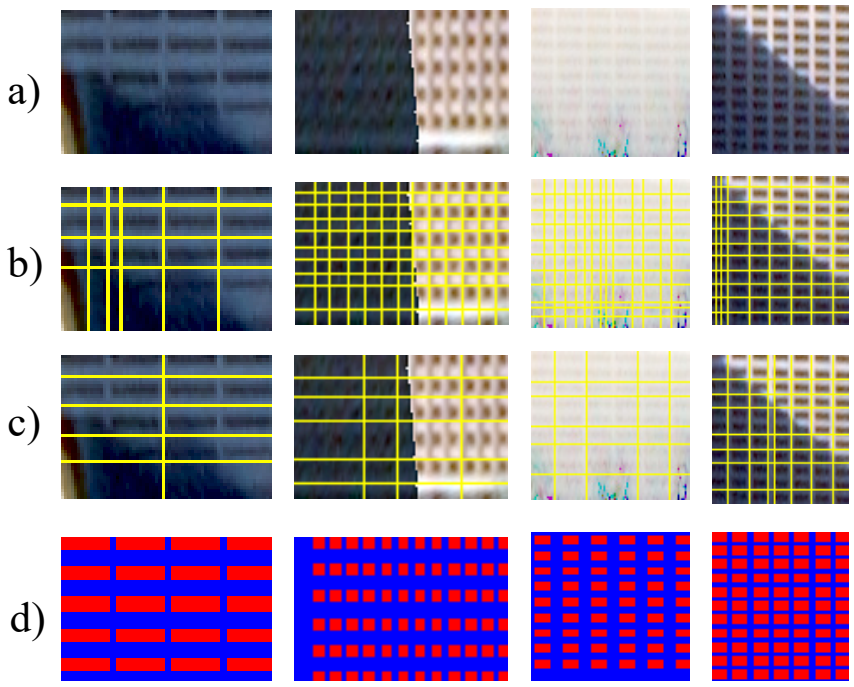


Figure 7.19. *Facade Subdivision Comparison.* We provide a) satellite-based facades to b) an image-based approach, c) Nishida et al. [33], and d) Ours.

7.3.5 Comparisons

We compare our approach to several state-of-the-art methods. First, in Figure 7.19 we show a visual comparison between the facade subdivision of b) an image-gradient-based approach (e.g., [10]), c) Nishida et al. [33] (retrained using the same training set as our approach), and d) our method. We highlight that Nishida et al. [33] (and also Teboul et al. [147]) essentially make use during their processing pipeline of an image-gradient based method similar to [10] (thus we include the image-gradient comparison). We also include facade quantitative comparisons in Table 7.13.

Second, we test two state-of-the-art neural network architectures for image inpainting/-completion: DeepFill [133] and PICNet [134]. With DeepFill determining which part to "fill" is an unaddressed challenge and thus for this comparison we manually select occluded, shadowed and/or tree-covered areas. In PICNet, we use the random rectangular mask generation method they provide (e.g., select a sufficient number of rectangles within the image to most likely performed all necessary in-filling). Please see Figure 7.20 for visual results.

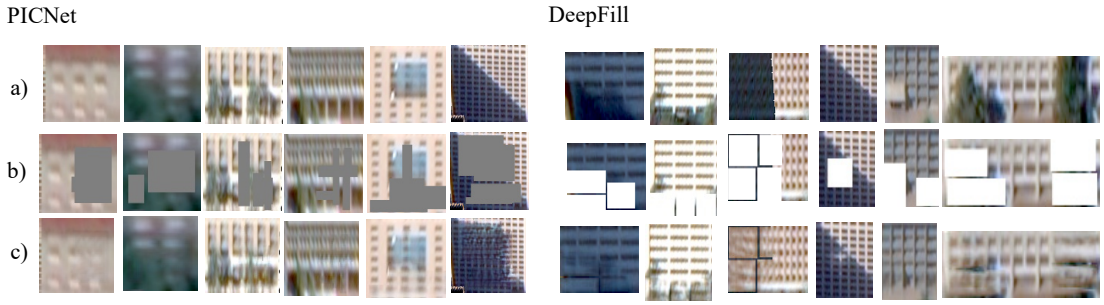


Figure 7.20. *Image In-painting.* a) Original facades. b) Rectangular areas to be filled-in. c) Results after inpainting.

While the methods are able to place content in the occluded areas, there are still significant artifacts which will hinder subsequent facade process.

To evaluate the facade processing ability directly using the segmentation model and image in-painting model, we evaluate performance using our 61 test images qualitatively and quantitatively. To be specific, for the segmentation model, we choose the aforementioned Pix2Pix_96 and apply it to the facade images directly. Then, we dilate each window/door to occupy a rectangular bounding box. For the image in-painting model, we choose DeepFill [133] and complete the facade images with manually selected masks. Then we apply the segmentation model to the completed facade images and we also use a version of the windows/doors dilated to rectangles. The quantitative metrics include pixel accuracy, precision, recall, and blob accuracy. In Figure 7.21 and Table 7.14, we show details of comparing our method to the segmentation model and the image in-painting model.

Table 7.14. *Quantitative comparison.* Pixel accuracy, precision, recall, F1 and blob accuracy evaluated for models from c) to e) in Figure 7.21. We evaluated c) and e) on 61 facades in the left table. However the right table shows applying d) to 22 facades (22 out of 61 facades are occluded and suitable for image in-painting.) and we manually set the mask as best as possible.

Method	Accuracy	Precision	Recall	F1	Blob
c)	0.835	0.695	0.868	0.758	0.891
e)	0.880	0.818	0.834	0.815	0.923

Method	Accuracy	Precision	Recall	F1	Blob
c)	0.802	0.705	0.797	0.728	0.840
d)	0.806	0.803	0.612	0.677	0.875
e)	0.843	0.768	0.828	0.783	0.918

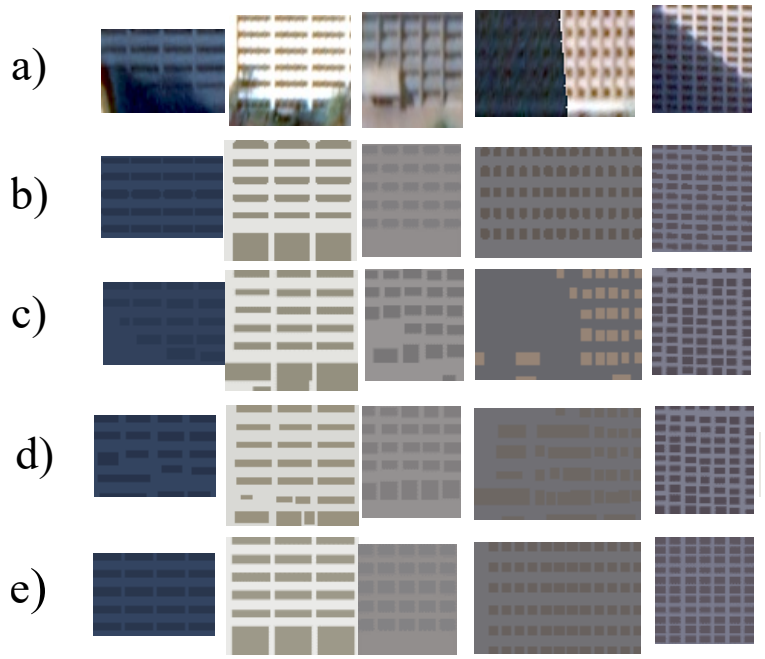


Figure 7.21. *Facade Comparisons.* Comparison to SOTA methods on facade parsing. a) Input satellite facades. b) Manually created ground truth. c) The results of applying Pix2Pix_96 to a). d) The results of applying Pix2Pix_96 to image completed by DeepFill [133]. e) Ours.

7.3.6 Examples

Finally, we show in Figure 7.22 many close-ups of reconstructed buildings as well as an overall view of one area (A1). Views of our additional areas (A2) and more buildings are in Figure D.1.

7.4 Other Applications

In the following sections, we conduct experiments and demonstrate results for other applications.

Figure 7.22. *Examples.* We show a view of a reconstructed area A1 within Google Earth and close-ups of our buildings.

7.4.1 Enhancing Urban Segmentation

We quantitatively and qualitatively evaluate our approach on building facades, for which we have ample ground truth. We also show preliminary qualitative results for building footprints and roofs.

Facade Evaluation Metrics

We rigorously evaluate our RFCNet in the case of facade segmentation. In particular, we evaluate two ways: facade correctness and facade regularization. Facade correctness focuses on the pixel accuracy, precision, and recall (completion) of the facade. Facade regularization measures the regularities of the facade.

Correctness. For facade correctness evaluation, we use statistical measures: Accuracy, Precision and Recall (see definitions in 5.1).

Regularization. Three metric error terms are defined to measure the regularization of a facade layout: alignment (E_a), size (E_s) and spacing (E_p). The errors measure the deviation from having groups of perfect alignment, groups of equal size, and groups of equal spacing. We adapt and modify the relevant definitions of [165].

- **Group:** We use a threshold t to split the window elements into a set of groups $G = \{..., g_{\mathrm{i}}, ...\}$ for the regularization error terms. A candidate group $g_{\mathrm{i}} = (E_{\mathrm{i}}, V_{\mathrm{i}})$

contains a set of window elements E_{i} that share the regularization term, and a set of values V_{i} (e.g., it is a set of x coordinate of left-corner from E_{i} for left alignment) that will be used to compute the corresponding regularization error.

- **Alignment Error:** E_a is defined as:

$$E_a = \sum_{A_{\text{i}}} (\sum_{g_{\text{j}}} \frac{stdvar(g_{\text{j}})}{scale(g_{\text{j}})} + w_a \|G\|), \tag{7.7}$$

where A_{i} stands for one alignment type among top, bottom, left and right alignments. g_{j} is a candidate group of A_{i}. $stdvar(g_{\text{i}})$ measures the standard deviation of V_{i} in g_{i}. $scale(g_{\text{i}})$ is used to scale the error. For left and right alignment, it is equal to the minimal width of window elements E_{i} in g_{i}. For top and bottom alignment, it is equal to the minimal height of E_{i}. $\|G\|$ is the number of candidate groups of A_{i}. We add $\|G\|$ to encourage fewer and larger groups. w_a is a weight that balances the two terms (e.g., $w_a = 0.01$).

- **Size Error:** E_s is defined as:

$$E_s = \sum_{g_{\text{j}}} \frac{stdvar(g_{\text{j}})}{scale(g_{\text{j}})} + w_s \|G\|, \tag{7.8}$$

where g_{j} is a candidate group that has the same window size. $stdvar(g_{\text{i}})$ measures the standard deviation of V_{i} in g_{i}. $scale(g_{\text{i}})$ is equal to minimal height or width of E_{i} in g_{i}. $\|G\|$ is the number of groups. w_s is a weight that balances the two terms (e.g., $w_s = 0.01$).

- **Spacing Error:** E_p is defined as:

$$E_p = \sum_{P_{\text{i}}} (\sum_{g_{\text{j}}} \frac{stdvar(g_{\text{j}})}{scale(g_{\text{j}})} + w_p \|G\|), \tag{7.9}$$

where P_{i} stands for horizontal or vertical spacing. g_{j} is a candidate group of P_{i}. $stdvar(g_{\text{i}})$ measures the standard deviation of V_{i} in g_{i}. $scale(g_{\text{i}})$ is equal to minimal spacing of V_{i} in g_{i}. $\|G\|$ is the number of candidate groups of P_{i}. w_p is a weight that balances the two terms (i.e., we usually set $w_p = 0.01$).

Facade Comparison

We evaluate RFCNet on the aforementioned ECP, WVS and GSV datasets with our defined evaluation metrics. To verify effectiveness of our method on enhancing initial segmentation, we compared our method to three segmentation models, state-of-the-art image completion models, and IPM methods both qualitatively and quantitatively. The comparison exhibits our approach shows a significant improvement. In the following sections, we only show comparison with one initial segmentation model.

Regularization. We apply our Regularization to initial segmentations for each of our three datasets and in all cases achieve better performance. The initial segmentation for each of the datasets is the first row in each group of Table 7.15 (e.g., for WVS, we improve the accuracy of the initial segmentation of Pix2Pix by **6.2%** and reduce alignment error E_a by **60.8%**). In addition, we retrain models in IPM methods using corresponding datasets for fair comparison. It shows our method improves facade correctness compared to them (e.g., accuracy is improved by **30.2%** for ECP, **30.1%** for GSV, and **4%** for WVS). Moreover, as shown in Figure 7.23, our module generates visually pleasant facade structures.

Table 7.15. *Regularization Quantitative Comparison.* We compare our Regularization (R) with the initial facade segmentation and IPM methods for ECP, WVS and GSV datasets. For facade correctness, higher is better. For facade regularization error, lower is better. **Note:** IPM methods generate regularized outputs, so regularization error is close to 0 but correctness is lower than others.

Dataset	Method	Facade Correctness			Facade Regulariz. Error		
		Acc.	Pre.	Rec.	E_a	E_s	E_p
ECP	DeepLabv3+	96.4%	84.7%	94.6%	1.01	0.05	0.18
	Nishida et al.	69.0%	50.8%	53.3%	—	—	—
	R	**99.2%**	**95.3%**	**99.4%**	**0.32**	**0.04**	**0.12**
WVS	Pix2Pix	87.2%	70.1%	91.9%	0.74	0.12	0.27
	Zhang et al.	89.4%	80.5%	84.6%	—	—	—
	R	**93.4%**	**81.6%**	**96.8%**	**0.29**	**0.08**	**0.16**
GSV	U-Net	90.5%	75.0%	90.0%	0.76	0.05	0.22
	Nishida et al.	68.0%	50.2%	51.6%	—	—	—
	R	**98.1%**	**92.8%**	**99.5%**	**0.25**	**0.04**	**0.13**

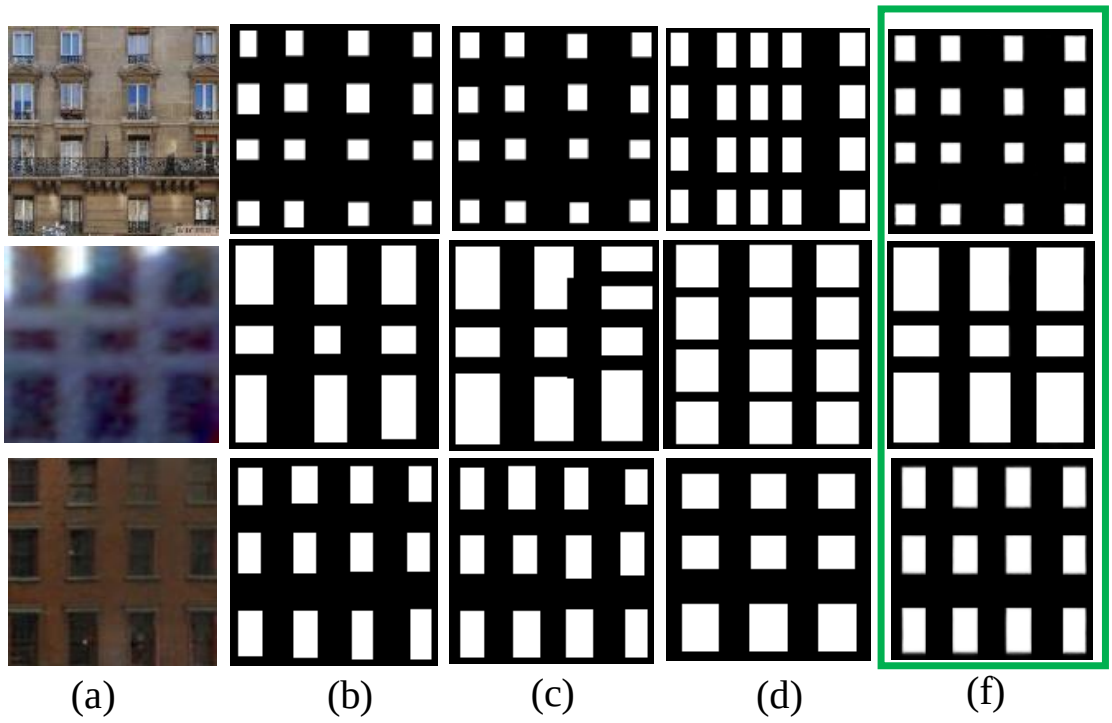


Figure 7.23. *Regularization Qualitative Comparison.* (a) Facade images from ECP, WVS and GSV respectively. (b) Ground Truth. (c) Initial Segmentation. (d) IPM results. (f) Our Regularization.

Regularization and Completion. For incomplete or partially occluded facade segmentation, as shown in Table 7.16, our Regularization and Completion (R & C) not only significantly improves the facade correctness and regularization metrics (e.g., for WVS, we improve the accuracy of the initial segmentation of Pix2Pix by **8.4%** and reduce alignment error E_a by **72.4%**), but also achieves better performance compared to the initial segmentation augmented by the image in-painting method DeepFill [133] (e.g., accuracy improved by **5.4%** and alignment error E_a reduced by **74.7%** for WVS). For a fair comparison, we refine DeepFill using our facade synthetic dataset. In addition, our method improves IPM methods with respect to facade correctness (e.g., accuracy improved by **29.5%** for ECP, **26.5%** for GSV, and **3.1%** for WVS). Further, our R & C outperforms the single Regularization module in terms of accuracy and recall. As illustrated in Figure 7.24, the outputs of our Regularization and Completion are visually appealing as well.

Table 7.16. *Regularization and Completion Quantitative Comparison.* After applying Regularization and Completion (R & C) to the initial segmentation, we compare our results to the segmentation, segmentation after completion using DeepFill, IPM methods and our Regularization (R). **Note:** Pix2Pix → DeepFill means DeepFill takes the initial segmentation of Pix2Pix as inputs.

Dataset	Method	Facade Correctness			Facade Regulariz. Error		
		Acc.	Pre.	Rec.	E_a	E_s	E_p
ECP	DeepLabv3+	91.0%	74.0%	74.8%	0.77	0.05	0.18
	DeepLabv3+ → DeepFill	93.2%	73.5%	95.8%	0.80	0.04	0.21
	Nishida et al.	67.4%	50.4%	54.4%	—	—	—
	R	95.3%	**92.8%**	78.4%	0.27	0.03	**0.12**
	R & C	**96.9%**	84.6%	**99.7%**	**0.24**	**0.03**	0.15
WVS	Pix2Pix	84.8%	77.2%	65.9%	0.87	0.07	0.32
	Pix2Pix → DeepFill	87.8%	75.5%	84.1%	0.95	0.07	0.33
	Zhang et al.	90.1%	82.3%	87.7%	—	—	—
	R	89.9%	**88.3%**	74.3%	**0.21**	0.05	**0.14**
	R & C	**93.2%**	83.2%	**94.7%**	0.24	**0.05**	0.19
GSV	U-Net	82.7%	74.7%	55.3%	0.71	0.06	0.14
	U-Net → DeepFill	87.3%	75.5%	77.5%	0.89	0.07	0.26
	Nishida et al.	67.1%	50.1%	51.3%	—	—	—
	R	88.5%	**91.6%**	65.4%	**0.20**	**0.04**	**0.07**
	R & C	**93.6%**	83.8%	**93.4%**	0.26	0.06	0.12

RFCNet. For partially-occluded facade segmentation with additional views, as shown in Table 7.17, our RFCNet achieves better facade results when evaluated for facade correctness and regularization as compared to the segmentation of the first view (e.g., for WVS, improves the segmentation accuracy by **9.1%** and reduces alignment error E_a by **60.8%**) and the segmentation completed using DeepFill (e.g., for WVS, improves accuracy by **4.4%** and reduces E_a by **70.1%**). Moreover, our RFCNet obtains better performance compared with only applying our Regularization and Completion to the first view. In addition, our method improves facade correctness compared to IPM methods (e.g., accuracy improved by **28.4%** for GSV and **5.6%** for WVS). What's more, our RFCNet improves accuracy and recall compared to both R & C and R. As demonstrated in Figure 7.25, our RFCNet results are qualitatively preferable.

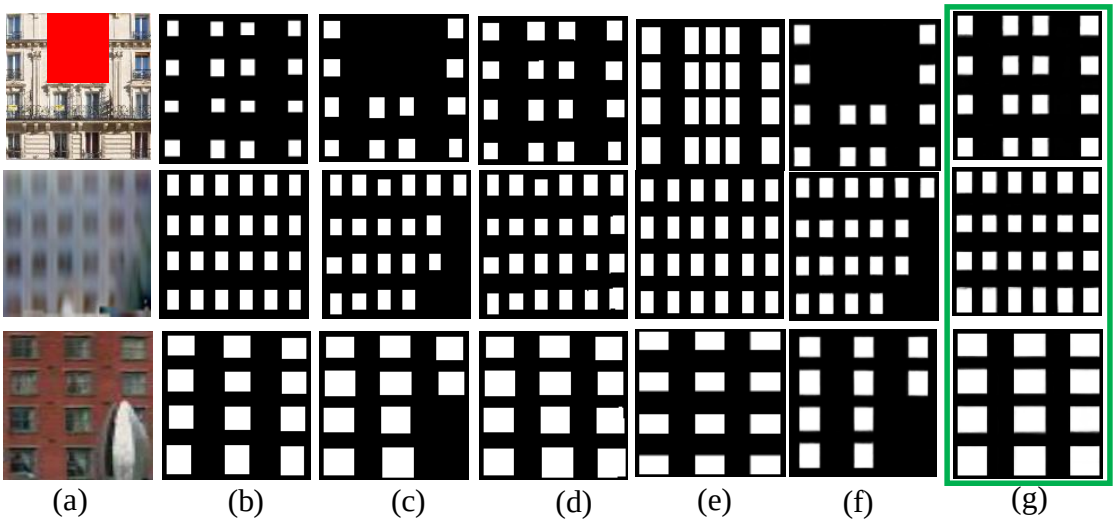


Figure 7.24. *Regularization and Completion Qualitative Comparison.* (a) Occluded facade images from ECP, WVS and GSV respectively. (b) Ground Truth. (c) Initial Segmentation. (d) Segmentation completed by DeepFill. (e) IPM results. (f) Our Regularization. (g) Our Regularization and Completion. **Note:** We manually mask ECP facade images shown in red box.

Table 7.17. *RFCNet Quantitative Comparison.* We compare the initial facade segmentation, the segmentation completed by DeepFill, IPM methods, and the outputs after applying our Regularization (R) and our R & C to the segmentation for the first view in WVS and GSV. Further, we evaluate the output after fusing additional views by applying our whole RFCNet.

Dataset	Method	Facade Correctness			Facade Regulariz. Error		
		Acc.	Pre.	Rec.	E_a	E_s	E_p
WVS	Pix2Pix	85.9%	86.1%	65.8%	0.51	0.09	0.27
	Pix2Pix → DeepFill	90.6%	85.1%	85.2%	0.67	0.13	0.43
	Zhang et al.	89.4%	80.7%	86.2%	—	—	—
	R	90.1%	**94.7%**	72.8%	**0.19**	**0.06**	**0.14**
	R & C	93.5%	85.5%	95.1%	0.20	0.10	0.15
	RFC	**95.0%**	88.7%	**96.0%**	0.20	0.09	0.15
GSV	U-Net	83.6%	80.0%	53.8%	0.71	0.07	0.15
	U-Net → DeepFill	87.2%	79.2%	71.1%	0.89	0.11	0.19
	Nishida et al.	66.9%	50.3%	50.7%	—	—	—
	R	89.7%	**94.0%**	67.6%	0.20	0.06	0.11
	R & C	91.6%	82.2%	88.1%	0.28	**0.06**	0.14
	RFC	**95.3%**	86.4%	**95.2%**	**0.15**	0.06	**0.09**

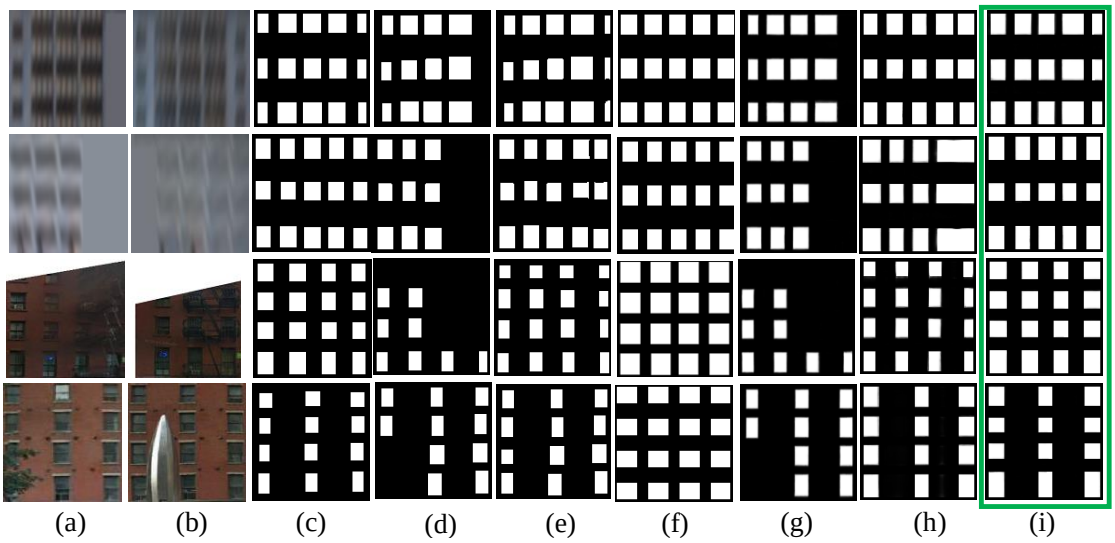


Figure 7.25. *RFCNet Qualitative Comparison.* (a) First view of facade images from WVS and GSV. (b) Additional views. (c) Ground Truth. (d) Segmentation of the first view. (e) Segmentation completed by DeepFill. (f) IPM results. (g) Our Regularization. (h) Our R & C. (i) Our entire RFCNet.

7.4.2 Building Contour Completion

Iterative Completion. In order to demonstrate the progression of completion, Figure 7.27 expands upon one of the footprints shown in Figure 7.26 (e.g., the middle footprint at layer 5). We show the footprint's completion behavior in incompleteness layer 1, 3, 5, and 7. As can be seen, as the incompleteness layer increases so does the number of iterations, requiring up to 3 iterations for convergence. The figure also shows, for comparison, the result of iterative naive image completion of the same footprint (e.g., call image completion recursively several times). Our approach produces the most complete footprints especially in the upper layers.

Archaeological Site. In Figure 7.28 we use our approach to complete several real-world sites. We show the aerial images, initial edges, our completion result, and the ground truth completion published by expert archaeologists.

Comparisons. Furthermore, we choose examples from Figure 7.26 and compare our method (GPBC) to four recent methods Pix2Pix [40], GLCIC [166], PIC [134], and SketchBERT [152] (we retrain all four models using our dataset for fairness). The implementation

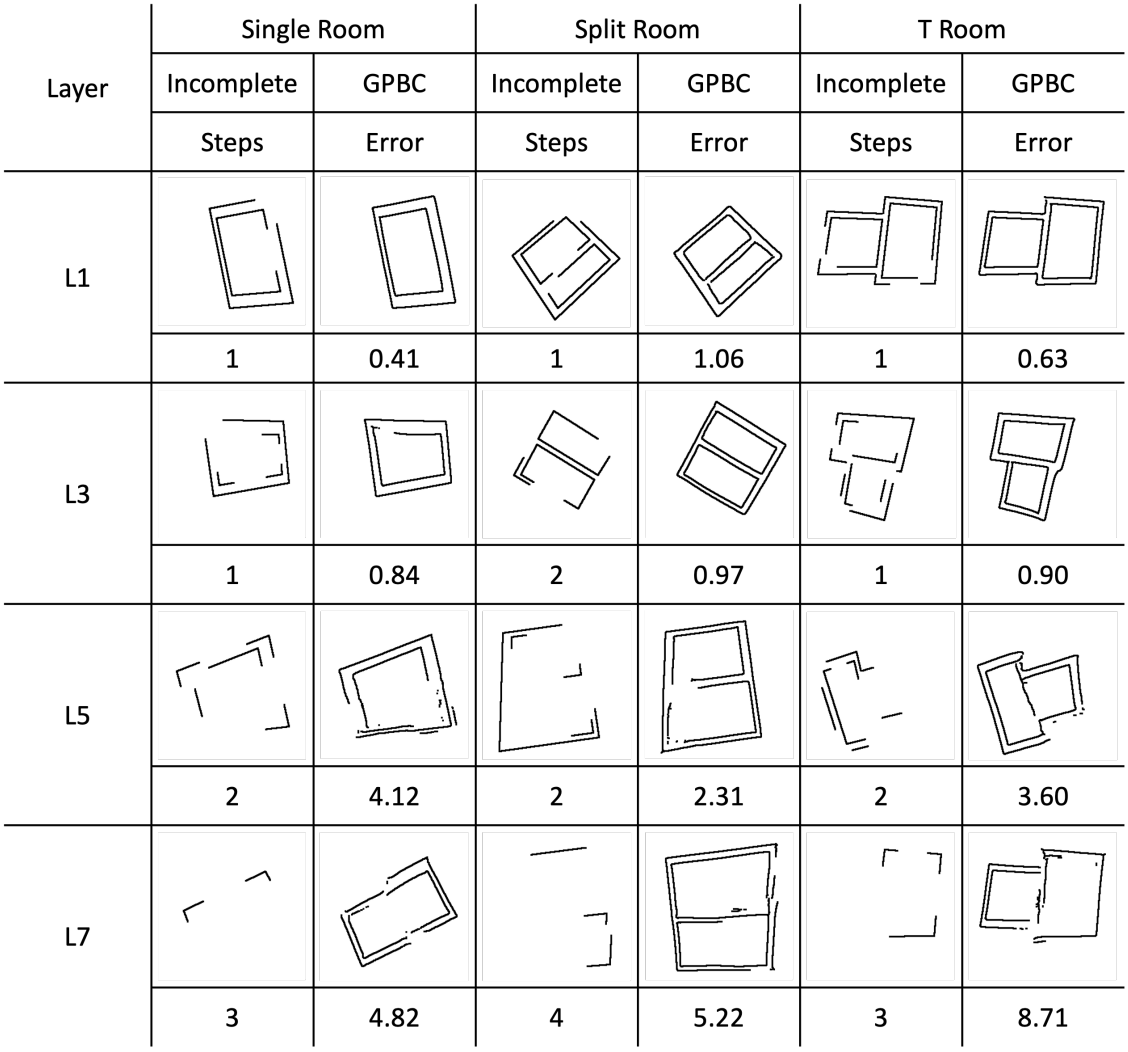


Figure 7.26. *Qualitative Analysis.* We show the visual results of our approach for different levels of initial incompleteness and for different building types.

of SketchBERT provided by the authors does not allow us to explicitly provide the incomplete input, but we can make the level of incompletion consistent with ours. As shown in Figure 7.29, our method consistently achieves better performance both qualitatively and quantitatively. Specifically, our results are more complete and clean than others (especially in L5 and L7). We improve the L2 pixel-wise errors significantly. The shown Layer 5 output from our method is improved by 4.8x (i.e., 4.8 times lower error), 1.8x, 2.8x, and 3.8x respectively as compared to Pix2Pix, GLCIC, PIC, and SketchBERT. Further, our layer 7 output is better by 2.1x, 1.7x, 2.2x, and 2.5x as compared to the same set of methods. For

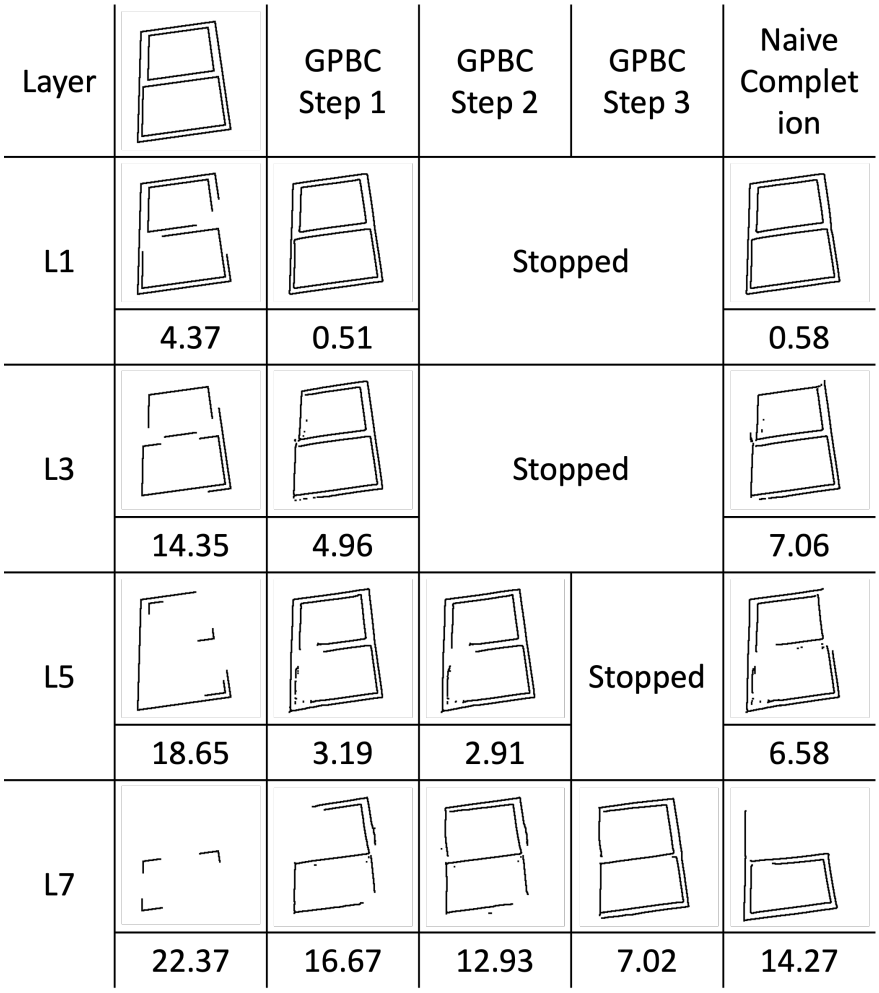


Figure 7.27. *Iterative Completion.* Step-by-step results of our proposed model and naive image completion on a split-room building.

instance, SketchBERT performs reasonably with single room cases, but is much worse for split or T-room cases.

Design Analysis. Our approach is the result of a variety of early experiments which ultimately led to the proposed design.

- We explored the single feature vs multiple feature methods (see Figure 7.30). Repetitive applications of single feature based completion tended to propagate errors to the final answer and thus multiple features seems to work best.

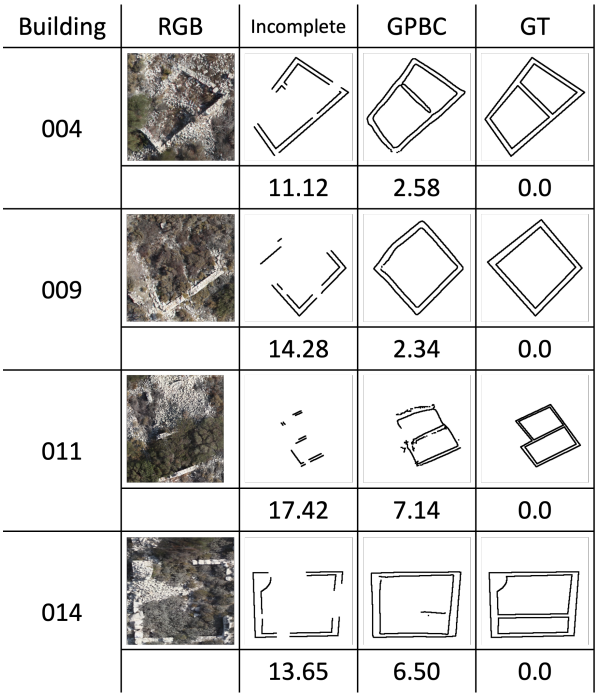


Figure 7.28. *Real-world Sites.* We use our method to complete images from actual archaeological sites on Bogsak Island.

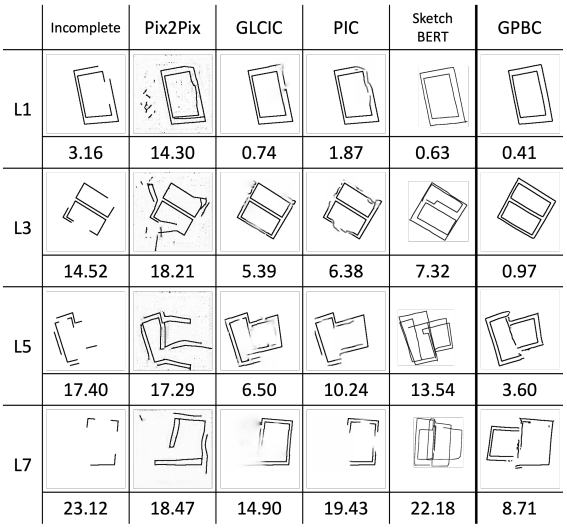


Figure 7.29. *Comparisons.* Our model GPBC outperforms previous state-of-the-art methods both visually and numerically.

- We experimented training with different levels of completion (see Figure 7.31). Having a 25% completion provided little new content and having 100% completion lead to improper contours, thus leaving 50% as a good compromise.
- We also investigated training with different amounts of positional perturbations of the features (e.g., during training, perturb the feature locations but keep the same output). Generally, we found training with such perturbations benefited lower-levels of incompleteness but had little, or worse, effect on high-levels of incompleteness, so we did not train with perturbations.
- For the dot-style features, we tested several dot sizes and Gaussian falloff rates, but the performance of these options was similar.

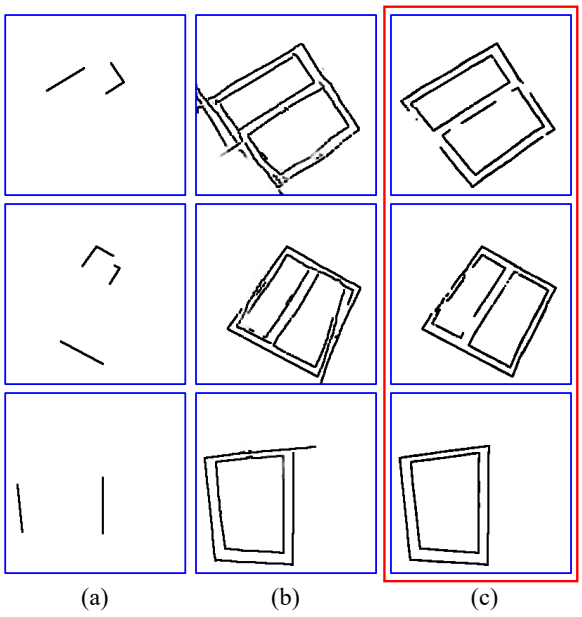


Figure 7.30. *Single vs. Multiple Features.* We show (a) incomplete footprints, (b) completion results by SF method, (c) completion results by MF method.

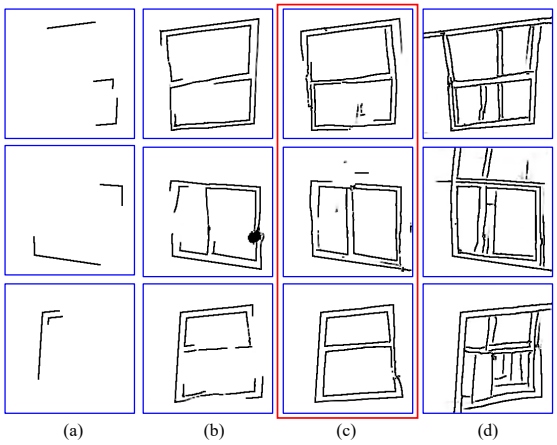


Figure 7.31. *Different Completion Levels.* We show (a) incomplete footprints, (b-d) completion results generated by the completion model trained under 25%, 50% and 100% completion levels, respectively.

8. CONCLUSIONS AND FUTURE WORK

8.1 Summary

We have shown our framework can be applied to automatically and procedurally reconstruct both different scales of urban structures and work for multiple resolutions of satellite images. Our work is exploring and exploiting the regularities of urban structures to conquer the difficulties of data noise, data sparsity and data uncertainty of satellite images. Our outputs are regularized, crisp, complete and parameterized models. Our procedural outputs for the urban environments have been used in many applications: improving flooding [167], helping archaeologist to quickly recover the structures of sites 6.2, providing researchers in urban planning and environmental science to quickly conduct experiments [168]–[171], etc.

8.2 Limitations

Although we support a very wide range of styles for urban structures, there are always exceptions. However, our approach has some limitations. First, our modeling process did not support non-rectangular shapes. Second, for urban structures whose styles are outside our defined grammars, we could give our best guess (see Figure 8.1 and Figure 8.2). Third, We do not support non-rectified satellite images (see Figure 8.2 iii)). In theory, our framework can handle these failure scenarios by adding more urban structure types to our synthetic training datasets. This is discussed in future work.

8.3 Future work

Our work has several avenues of future work discussed as short-term goals and long-term goals.

Short-Term Future Research

We would like to support more types/shapes of urban structures (e.g., non-right-angle buildings, not orthorectified satellite image, mansard roof type, circular or oval windows, etc.). In theory, our framework can handle these scenarios by adding more urban structure

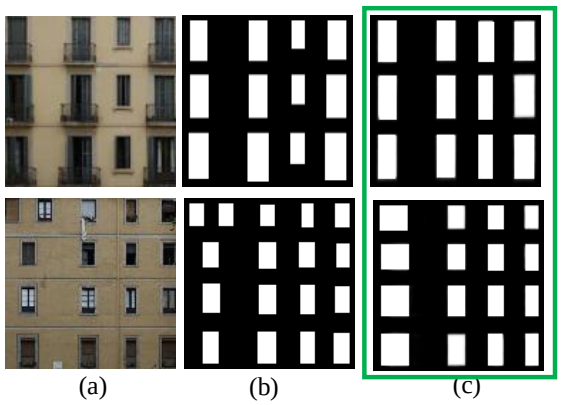


Figure 8.1. *Failure Examples.* (a) Facade images. (b) Initial segmentation. (c) Our results.

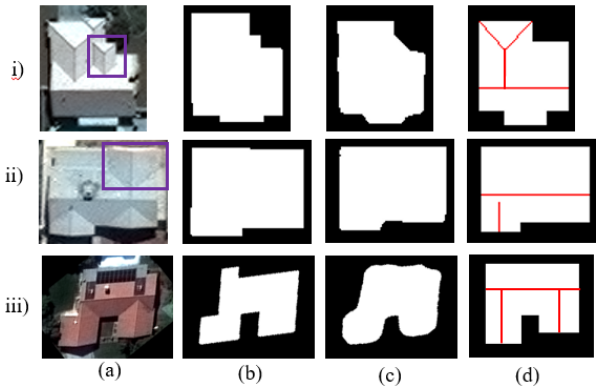


Figure 8.2. *Failure Examples.* (a) Real images. (b) Ground truth footprints. (c) Initial footprint segmentations. (d) Ours results.

types to our synthetic training datasets. Additionally, We can explore more comprehensive representations (patterns) for those aforementioned non-regular shapes. Further, we can also output additional confidence values (e.g., how confident or possible our generated models are correct) to our reconstructed urban models.

We would like to capture finer details (e.g., dormers, chimneys, etc.) of urban areas. For now, our framework output city models, building models, and facade models. However, more details can be reconstructed. For this purpose, we can take advantages of deep learning or even 3D information (e.g., DSM, DTM, LiDAR, etc.) to detect and extract more features.

We would like to extend the evaluations of our framework. Currently, our framework is mostly compared to state-of-the-art theoretical papers. Since our framework has been or (can be easily) extended to real life applications, we should consider the evaluations in those applications and show how efficient our approaches are and how much effort we could save for other researchers.

Long-Term Future Research

As stated by Warren Weaver in his 1948 historical address, the problems modern science faces going forward are those of large organized complexity – the problems of interest in

cities, humans, and the environment are not random but rather the consequence of a very complex set of interactions. Understanding and improving these problems is the objective of modern science. Towards that, I will pursue expanding our framework to consider the dimensions of efficiency, large-scale, and data heterogeneity.

Efficient AI: Recent advances in Artificial Intelligence (AI), particularly in deep learning, have provided significant improvements in the ability to understand visual content. However, mainstream computer vision/graphics research has given little consideration to speed or computation time, data acquisition, and even less to constraints such as computation power, GPU memory and model size. Nevertheless, addressing all of these metrics is critical if advances in computer vision are going to be widely applied to embedded systems (e.g., mobile, AR/VR devices, autonomous driving). Specific topics include: learning from limited data (zero-shot, one-shot, few-shot learning), learning from imperfect data, learning from synthetic data, self-supervised (or unsupervised) deep learning, efficient architecture search algorithm, representation learning, neural network compression (sparsification, binarization, pruning), etc.

Large-Scale AI: Research in this area for me is motivated by incorporating and enhancing AI techniques in the realm of climate change (environmental science) to push the academic and scientific frontier, and is also driven by the urgent need for a fast-reacting and holistic system to simulate and predict climate change on a large-scale. Although a considerable number of isolated Climate System features have been analyzed with AI techniques (e.g., local climatic impacts prediction), more generic application to understand better the full climate system has not occurred. This demands innovation and collaboration among the computer vision, AI disciplines (e.g., machine learning, deep learning), and the remote sensing community to boost automated interpretation of Earth Observation (EO) big data. It implies the need for applications in the following research topics: multiple inference tasks, data fusion (multi-resolution, multi-temporal, and multi-modality fusion), large-scale surveillance, 3D urban modelling and reconstruction, navigation systems, natural hazard forecast and response, climate change monitoring, etc.

AI Cities: Cities contain heterogeneous sets of data that can be integrated with AI techniques as a strategic approach to sustainability. The post-pandemic era has brought a

new global trend of "greenhouse gas pollution reduction" and this concept adds new dimensions to urbanization which requires quickly upgrading existing cities. Transportation as one of the largest segments in the city can benefit from actionable insights derived from the mission. Among traffic, signaling systems, transportation systems, infrastructure, and transit, the opportunity for insights from these sensors to make transportation systems smarter is immense. Unfortunately, there are several reasons why these potential benefits have not yet materialized. Poor data quality, broad data heterogeneity, lacking of data labels, and missing high-quality models that can convert the data into actionable insights are some of the biggest impediments to unlocking the value of the data. AI cities intend to bridge the gap between real world city-scale problems and the cutting edge research and development in intelligent techniques. The AI cities will specifically focus on a broad diverse set of problems such as anomaly detection (detecting anomalies such as lane violation, wrong-direction driving, etc.), traffic congestion (smart parking systems, city road layout modeling), urban flooding and efficient urban mobility & public transport.

REFERENCES

[1] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[2] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing and Modeling*, 3rd. Orlando, FL, USA: Academic Press, Inc., 2003.

[3] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes, "A survey on procedural modelling for virtual worlds," *Comp. Graph. Forum*, vol. 33, no. 6, pp. 31–50, 2014, ISSN: 1467-8659. DOI: 10.1111/cgf.12276. [Online]. Available: http://dx.doi.org/10.1111/cgf.12276.

[4] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, 2001, pp. 301–308. DOI: http://doi.acm.org/10.1145/383259.383292.

[5] G. Chen, G. Esch, P. Wonka, Pascal, Müller, and E. Zhang, "Interactive procedural street modeling," *ACM Trans. Graph.*, vol. 27, no. 3, p. 35, 2007. DOI: http://doi.acm.org/10.1145/1278780.1278822.

[6] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin, "Procedural generation of roads," *Comp. Graph. Forum*, vol. 29, no. 2, pp. 429–438, 2010.

[7] C. A. Vanegas, T. Kelly, B. Weber, J. Halatsch, D. G. Aliaga, and P. Müller, "Procedural generation of parcels in urban modeling," *Comput. Graph. Forum*, vol. 31, no. 2pt3, pp. 681–690, May 2012, ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.03047.x. [Online]. Available: https://doi.org/10.1111/j.1467-8659.2012.03047.x.

[8] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky, "Instant architecture," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 669–677, 2003. DOI: http://doi.acm.org/10.1145/882262.882324.

[9] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. v. Gool, "Procedural modeling of buildings," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 614–623, Jul. 2006, ISSN: 0730-0301. DOI: 10.1145/1141911.1141931. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141931.

[10] P. Müller, G. Zeng, P. Wonka, and L. v. Gool, "Image-based procedural modeling of facades," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007, ISSN: 0730-0301. DOI: 10.1145/1276377.1276484. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276484.

[11] C.-H. Shen, S.-S. Huang, H. Fu, and S.-M. Hu, "Adaptive partitioning of urban facades," *ACM Trans. Graph.*, vol. 30, no. 6, 184:1–184:10, Dec. 2011, ISSN: 0730-0301. DOI: 10.1145/2070781.2024218. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024218.

[12] F. Bao, M. Schwarz, and P. Wonka, "Procedural facade variations from a single layout," *ACM Trans. Graph.*, vol. 32, no. 1, 8:1–8:13, Feb. 2013, ISSN: 0730-0301. DOI: 10.1145/2421636.2421644. [Online]. Available: http://doi.acm.org/10.1145/2421636.2421644.

[13] H. Zhang, K. Xu, W. Jiang, J. Lin, D. Cohen-Or, and B. Chen, "Layered analysis of irregular facades via symmetry maximization," *ACM Trans. Graph.*, vol. 32, no. 4, 121:1–121:13, Jul. 2013, ISSN: 0730-0301. DOI: 10.1145/2461912.2461923. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461923.

[14] M. Lipp, D. Scherzer, P. Wonka, and M. Wimmer, "Interactive modeling of city layouts using layers of procedural content," *Comp. Graph. Forum*, vol. 30, no. 2, pp. 345–354, 2011, ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2011.01865.x. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2011.01865.x.

[15] T. Gwenola and S. Donikian, "Modelling virtual cities dedicated to behavioural animation," *Comp. Graph. Forum*, vol. 19, no. 3, pp. 71–80, Sep. 2000.

[16] B. Weber, P. Mueller, P. Wonka, and M. Gross, "Interactive geometric simulation of 4d cities," *Comp. Graph. Forum*, Apr. 2009. [Online]. Available:

.

[17] C. A. Vanegas, D. G. Aliaga, B. Benes, and P. A. Waddell, "Interactive design of urban spaces using geometrical and behavioral modeling," pp. 1–10, 2009. DOI: http://doi.acm.org/10.1145/1661412.1618457.

[18] I. Garcia-Dorado, D. G Aliaga, and S. V Ukkusuri, "Designing large-scale interactive traffic animations for urban modeling," in *Comp. Graph. Forum*, Wiley Online Library, vol. 33, 2014, pp. 411–420.

[19] C.-H. Peng, Y.-L. Yang, F. Bao, *et al.*, "Computational network design from functional specifications," *ACM Trans. Graph.*, vol. 35, no. 4, p. 131, 2016.

[20] C. A. Vanegas, D. G. Aliaga, P. Wonka, P. Müller, P. Waddell, and B. Watson, "Modelling the appearance and behaviour of urban spaces," in *Comp. Graph. Forum*, Wiley Online Library, vol. 29, 2010, pp. 25–42.

[21] D. G. Aliaga, İ. Demir, B. Benes, and M. Wand, "Inverse procedural modeling of 3d models for virtual worlds," in *ACM SIGGRAPH 2016 Courses*, ser. SIGGRAPH '16, Anaheim, California: ACM, 2016, 16:1–16:316, ISBN: 978-1-4503-4289-6. DOI: 10.1145/2897826.2927323. [Online]. Available: http://doi.acm.org/10.1145/2897826.2927323.

[22] O. Šťava, S. Pirk, J. Kratt, *et al.*, "Inverse procedural modelling of trees," *Comp. Graph. Forum*, vol. 33, no. 6, pp. 118–131, 2014, ISSN: 1467-8659. DOI: 10.1111/cgf.12282. [Online]. Available: http://dx.doi.org/10.1111/cgf.12282.

[23] D. G. Aliaga, C. A. Vanegas, and B. Benes, "Interactive example-based urban layout synthesis," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–10, 2008. DOI: http://doi.acm.org/10.1145/1409060.1409113.

[24] O. Šťava, B. Benes, R. Měch, D. G. Aliaga, and P. Krištof, "Inverse procedural modeling by automatic generation of L-systems," *Comp. Graph. Forum*, vol. 29, no. 2, pp. 665–674, 2010, ISSN: 1467-8659. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01636.x.

[25] O. Stava, S. Pirk, J. Kratt, *et al.*, "Inverse procedural modelling of trees," vol. 33, no. 6, pp. 118–131, Sep. 2014, ISSN: 0167-7055. DOI: 10.1111/cgf.12282. [Online]. Available: https://doi.org/10.1111/cgf.12282.

[26] T. Hädrich, B. Benes, O. Deussen, and S. Pirk, "Interactive modeling and authoring of climbing plants," *Computer Graphics Forum*, vol. 36, no. 2, pp. 49–61, 2017. DOI: https://doi.org/10.1111/cgf.13106. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13106. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13106.

[27] M. Bokeloh, M. Wand, and H.-P. Seidel, "A connection between partial symmetry and inverse procedural modeling," pp. 1–10, 2010. DOI: http://doi.acm.org/10.1145/1833349.1778841.

[28] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun, "Metropolis procedural modeling," *ACM Trans. Graph.*, vol. 30, 11:1–11:14, 2 Apr. 2011, ISSN: 0730-0301. DOI: http://doi.acm.org/10.1145/1944846.1944851. [Online]. Available: http://doi.acm.org/10.1145/1944846.1944851.

[29] C. A. Vanegas, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and P. Waddell, "Inverse design of urban procedural models," *ACM Trans. Graph.*, vol. 31, no. 6, 168:1–168:11, Nov. 2012, ISSN: 0730-0301. DOI: 10.1145/2366145.2366187. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366187.

[30] I. Demir, D. G. Aliaga, and B. Benes, "Coupled segmentation and similarity detection for architectural models," *ACM Trans. Graph.*, vol. 34, no. 4, 104:1–104:11, Jul. 2015, ISSN: 0730-0301. DOI: 10.1145/2766923. [Online]. Available: http://doi.acm.org/10.1145/2766923.

[31] I. Demir, D. G. Aliaga, and B. Benes, "Proceduralization for editing 3d architectural models," in *Intl. Conf. on 3D Vision (3DV)*, 2016, pp. 194–202. DOI: 10.1109/3DV.2016.28.

[32] G. Nishida, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and A. Bousseau, "Interactive sketching of urban procedural models," *ACM Trans. Graph.*, vol. 35, no. 4, 130:1–130:11, Jul. 2016, ISSN: 0730-0301. DOI: 10.1145/2897824.2925951. [Online]. Available: http://doi.acm.org/10.1145/2897824.2925951.

[33] G. Nishida, A. Bousseau, and D. G. Aliaga, "Procedural Modeling of a Building from a Single Image," *Computer Graphics Forum*, Eurographics, vol. 37, no. 2, 2018. [Online]. Available: https://hal.inria.fr/hal-01810207.

[34] T. Kelly, J. Femiani, P. Wonka, and N. J. Mitra, "Bigsur: Large-scale structured urban reconstruction," *ACM Trans. Graph.*, vol. 36, no. 6, Nov. 2017. DOI: 10.1145/3130800.3130823. [Online]. Available: https://doi.org/10.1145/3130800.3130823.

[35] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[36] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[37] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *ICLR*, 2015.

[38] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Dec. 2017.

[39] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *TPAMI*, 2017.

[40] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[41] H. Zhang, K. Dana, J. Shi, *et al.*, "Context encoding for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[42] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[43] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-scnn: Gated shape cnns for semantic segmentation," *ICCV*, 2019.

[44] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR*, 2016.

[45] T. Kelly, P. Guerrero, A. Steed, P. Wonka, and N. J. Mitra, "Frankengan: Guided detail synthesis for building mass-models using style-synchonized gans," *ACM Transactions on Graphics*, vol. 37, no. 6, 2018.

[46] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *ICLR*, 2018.

[47] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[48] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *(CVPR)*, Jun. 2018.

[49] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *(CVPR)*, Jun. 2019.

[50] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation," in *ECCV*, 2020.

[51] Y. Qian, H. Zhang, and Y. Furukawa, "Roof-gan: Learning to generate roof geometry and relations for residential houses," *CoRR*, vol. abs/2012.09340, 2020. [Online]. Available: https://arxiv.org/abs/2012.09340.

[52] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, "SpaceNet: A Remote Sensing Dataset and Challenge Series," *arXiv e-prints*, arXiv:1807.01232, arXiv:1807.01232, Jul. 2018. arXiv: 1807.01232 [cs.CV].

[53] S. P. Mohanty, J. Czakon, K. A. Kaczmarek, *et al.*, "Deep learning for understanding satellite imagery: An experimental survey," *Frontiers in Artificial Intelligence*, vol. 3, 2020.

[54] O. C. Ozcanli, Y. Dong, J. L. Mundy, H. Webb, R. Hammoud, and V. Tom, "A comparison of stereo and multiview 3-d reconstruction using cross-sensor satellite imagery," in *IEEE Computer Vision and Pattern Recognition Workshops*, 2015, pp. 17–25.

[55] R. Qin, "Automated 3d recovery from very high resolution multi-view satellite images," in *ASPRS (IGTF) annual Conference*, 2017, p. 10.

[56] E. Zheng, K. Wang, E. Dunn, and J.-M. Frahm, "Minimal solvers for 3d geometry from satellite imagery," in *IEEE International Conference on Computer Vision*, 2015, pp. 738–746.

[57] X. Zhang, A. Shehata, B. Benes, and D. Aliaga, "Automatic deep inference of procedural cities from global-scale spatial data," *ACM Trans. Spatial Algorithms Syst.*, vol. 7, no. 2, Jun. 2021, ISSN: 2374-0353. DOI: 10.1145/3423422. [Online]. Available: https://doi.org/10.1145/3423422.

[58] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer, "A survey of urban reconstruction," in *Computer Graphics Forum*, vol. 32, 2013.

[59] T. Schöps, J. L. Schönberger, S. Galliani, *et al.*, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *CVPR*, vol. 3, 2017.

[60] C. A. Vanegas, D. G. Aliaga, and B. Benes, "Building reconstruction using manhattan-world grammars," *CVPR*, pp. 358–365, 2010. DOI: http://doi.ieeecomputersociety.org/10.1109/CVPR.2010.5540190.

[61] Q. Shan, C. Wu, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz, "Accurate geo-registration by ground-to-aerial image matching," in *Proceedings of the 2014 2Nd International Conference on 3D Vision - Volume 01*, ser. 3DV '14, Washington, DC, USA: IEEE Computer Society, 2014, pp. 525–532, ISBN: 978-1-4799-7000-1. DOI: 10.1109/3DV.2014.69. [Online]. Available: http://dx.doi.org/10.1109/3DV.2014.69.

[62] Y. Hou, J. Peng, Z. Hu, P. Tao, and J. Shan, "Planarity constrained multi-view depth map reconstruction for urban scenes," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 139, pp. 133–145, 2018, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.03.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271618300601.

[63] L. Duan and F. Lafarge, "Towards large-scale city reconstruction from satellites," in *Proc. of the European Conference on Computer Vision (ECCV)*, Amsterdam, Netherlands, 2016.

[64] C. Yi, Y. Zhang, Q. Wu, *et al.*, "Urban building reconstruction from raw lidar point data," *Computer-Aided Design*, vol. 93, pp. 1–14, 2017, ISSN: 0010-4485. DOI: https://doi.org/10.1016/j.cad.2017.07.005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010448517301331.

[65] C. Poullis and S. You, "Automatic reconstruction of cities from remote sensor data," in *CVPR*, IEEE, 2009, pp. 2775–2782.

[66] B. Bonczak and C. E. Kontokosta, "Large-scale parameterization of 3d building morphology in complex urban landscapes using aerial lidar and city administrative data," *Computers, Environment and Urban Systems*, vol. 73, pp. 126–142, 2019, ISSN: 0198-9715. DOI: https://doi.org/10.1016/j.compenvurbsys.2018.09.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971518300176.

[67] Y. Park and J.-M. Guldmann, "Creating 3d city models with building footprints and lidar point cloud classification: A machine learning approach," *Computers, Environment and Urban Systems*, vol. 75, pp. 76–89, 2019, ISSN: 0198-9715. DOI: https://doi.org/10.1016/j.compenvurbsys.2019.01.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971518304733.

[68] L. Zhang, Z. Li, A. Li, and F. Liu, "Large-scale urban point cloud labeling and reconstruction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 138, pp. 86–100, 2018, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.02.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271618300376.

[69] Q.-Y. Zhou and U. Neumann, "2.5 d building modeling by discovering global regularities," in *CVPR*, IEEE, 2012, pp. 326–333.

[70] Y. Verdie, F. Lafarge, and P. Alliez, "LOD Generation for Urban Scenes," *ACM Transactions on Graphics*, vol. 34, no. 3, p. 15, 2015. [Online]. Available: https://hal.inria.fr/hal-01113078.

[71] P. K. Agarwal, A. Beutel, and T. Mølhave, "Terranni: Natural neighbor interpolation on 2d and 3d grids using a gpu," *ACM Trans. Spatial Algorithms Syst.*, vol. 2, no. 2, 7:1–7:31, Jun. 2016, ISSN: 2374-0353. DOI: 10.1145/2786757. [Online]. Available: http://doi.acm.org/10.1145/2786757.

[72] G. Facciolo, C. de Franchis, and E. Meinhardt-Llopis, "Automatic 3d reconstruction from multi-date satellite images," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul. 2017, pp. 1542–1551. DOI: 10.1109/CVPRW.2017.198.

[73] E. Rupnik, M. Pierrot-Deseilligny, and A. Delorme, "3d reconstruction from multi-view vhr-satellite images in micmac," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 139, pp. 201–211, 2018, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.03.016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271618300819.

[74] M. D. Robles-Ortega, L. M. Ortega, and F. R. Feito, "Efficient visibility determination in urban scenes considering terrain information," *ACM Trans. Spatial Algorithms Syst.*, vol. 3, no. 3, 10:1–10:24, Nov. 2017, ISSN: 2374-0353. DOI: 10.1145/3152536. [Online]. Available: http://doi.acm.org/10.1145/3152536.

[75] J. O. Vollmer, M. Trapp, H. Schumann, and J. Döllner, "Hierarchical spatial aggregation for level-of-detail visualization of 3d thematic data," *ACM Trans. Spatial Algorithms Syst.*, vol. 4, no. 3, 9:1–9:23, Sep. 2018, ISSN: 2374-0353. DOI: 10.1145/3234506. [Online]. Available: http://doi.acm.org/10.1145/3234506.

[76] S. Paisitkriangkrai, J. Sherrah, P. Janney, and A. Van-Den Hengel, "Effective semantic pixel labelling with convolutional networks and conditional random fields," eng, IEEE, Jun. 2015, pp. 36–43, ISBN: 978-1-4673-6759-2.

[77] J. Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," *CoRR*, vol. abs/1606.02585, 2016. arXiv: 1606.02585. [Online]. Available: http://arxiv.org/abs/1606.02585.

[78] J. Huang and S. You, "Point cloud labeling using 3d convolutional neural network," in *Pattern Recognition (ICPR)*, IEEE, 2016, pp. 2670–2675.

[79] M. Volpi and D. Tuia, "Dense semantic labeling of subdecimeter resolution images with convolutional neural networks," eng, *Geoscience and Remote Sensing, IEEE Trans. on*, vol. 55, no. 2, pp. 881–893, Feb. 2017, ISSN: 0196-2892.

[80] M. Baatz, U. Benz, S. Dehghani, *et al.*, "Ecognition user guide," *Definiens Imaging GmbH, Munich, Germany*, 2004.

[81] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[82] K. G. Nikolakopoulos, "Evaluating alos aw3d30 data," in *Fifth International Conference on Remote Sensing and Geoinformation of the Environment (RSCy2017)*, International Society for Optics and Photonics, vol. 10444, 2017, p. 1 044 408.

[83] E. A. Bright, A. N. Rose, and M. L. Urban, "Landscan 2012," 2012.

[84] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, ser. NIPS'12, Lake Tahoe, Nevada, 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257.

[85] I. D. Stewart and T. R. Oke, "Local climate zones for urban temperature studies," *Bulletin of the American Meteorological Society*, vol. 93, no. 12, pp. 1879–1900, 2012.

[86] X. Zhang, C. May, G. Nishida, and D. Aliaga, "Progressive regularization of satellite-based 3d buildings for interactive rendering," in *Symposium on Interactive 3D Graphics and Games*, 2020, pp. 1–9. DOI: 10.1145/3384382.3384526.

[87] R. Wang, J. Peethambaran, and D. Chen, "Lidar point clouds to 3-d urban models: A review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 2, pp. 606–627, Feb. 2018, ISSN: 2151-1535. DOI: 10.1109/JSTARS.2017.2781132.

[88] D. S. Chen, "A data-driven intermediate level feature extraction algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 749–758, Jul. 1989, ISSN: 1939-3539. DOI: 10.1109/34.192470.

[89] P. J. Besl and R. C. Jain, "Segmentation through variable-order surface fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 2, pp. 167–192, Mar. 1988, ISSN: 1939-3539. DOI: 10.1109/34.3881.

[90] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692. [Online]. Available: http://doi.acm.org/10.1145/358669.358692.

[91] J. Chen and B. Chen, "Architectural modeling from sparsely scanned range data," *International Journal of Computer Vision*, vol. 78, pp. 223–236, Jul. 2008. DOI: 10.1007/s11263-007-0105-5.

[92] M. Li, P. Wonka, and L. Nan, "Manhattan-world urban reconstruction from point clouds," vol. 9908, Oct. 2016, pp. 54–69, ISBN: 978-3-319-46492-3. DOI:
.

[93] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, 2007, ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2007.01016.x.

[94] L. Nan and P. Wonka, "Polyfit: Polygonal surface reconstruction from point clouds," 2017.

[95] Q.-Y. Zhou and U. Neumann, "2.5d dual contouring: A robust approach to creating building models from aerial lidar point clouds," in *European Conference on Computer Vision*, 2010.

[96] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen, "Smartboxes for interactive urban reconstruction," *ACM Trans. Graph.*, vol. 29, no. 4, 93:1–93:10, Jul. 2010, ISSN: 0730-0301. DOI: 10.1145/1778765.1778830. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778830.

[97] J. Xiao and Y. Furukawa, "Reconstructing the world's museums," in *ECCV*, 2012.

[98] W. Nguatem and H. Mayer, "Modeling urban scenes from pointclouds," in *IEEE International Conference on Computer Vision*, 2017, pp. 3837–3846.

[99] Y. Verdie, F. Lafarge, and P. Alliez, "Lod generation for urban scenes," *ACM Transactions on Graphics*, vol. 34, no. 3, 2015.

[100] M. J. Leotta, C. Long, B. Jacquet, *et al.*, "Urban semantic 3d reconstruction from multiview satellite imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2019.

[101] M. Kozinski, R. Gadde, S. Zagoruyko, G. Obozinski, and R. Marlet, "A mrf shape prior for facade parsing with occlusions," in *IEEE Computer Vision and Pattern Recognition*, 2015, pp. 2820–2828.

[102] D. Marcos, D. Tuia, B. Kellenberger, *et al.*, English, in *Proceedings - 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 31st Meeting of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018 ; Conference date: 18-06-2018 Through 22-06-2018, IEEE computer society, Dec. 2018, pp. 8877–8885. DOI: 10.1109/CVPR.2018.00925.

[103] D. Cheng, R. Liao, S. Fidler, and R. Urtasun, "Darnet: Deep active ray network for building segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[104] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *INTERNATIONAL JOURNAL OF COMPUTER VISION*, vol. 1, no. 4, pp. 321–331, 1988.

[105] Z. Li, J. D. Wegner, and A. Lucchi, "Topological map extraction from overhead images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[106] N. Nauata and Y. Furukawa, "Vectorizing world buildings: Planar graph reconstruction by primitive detection and relationship inference," in *European Conference on Computer Vision*, 2020.

[107] F. Zhang, N. Nauata, and Y. Furukawa, "Conv-mpn: Convolutional message passing neural network for structured outdoor architecture reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[108] M. Li, F. Lafarge, and R. Marlet, "Approximating shapes in images with low-complexity polygons," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[109] H. Arefi and P. Reinartz, "Building reconstruction using dsm and orthorectified images," *Remote Sensing*, vol. 5, no. 4, pp. 1681–1703, 2013, ISSN: 2072-4292. [Online]. Available: https://www.mdpi.com/2072-4292/5/4/1681.

[110] Y. Zheng and Y. Zheng, "A hybrid approach for three-dimensional building reconstruction in indianapolis from lidar data," *Remote Sensing*, vol. 9(4), Mar. 2017. DOI: 10.3390/rs9040310.

[111] L. Li, J. Yao, J. Tu, X. Liu, Y. Li, and L. Guo, "Roof plane segmentation from airborne lidar data using hierarchical clustering and boundary relabeling," *Remote Sensing*, vol. 12, no. 9, 2020, ISSN: 2072-4292. [Online]. Available: https://www.mdpi.com/2072-4292/12/9/1363.

[112] M. S. Y. Ywata, A. P. Dal Poz, M. H. Shimabukuro, and H. C. de Oliveira, "Snake-based model for automatic roof boundary extraction in the object space integrating a high-resolution aerial images stereo pair and 3d roof models," *Remote Sensing*, vol. 13, no. 8, 2021, ISSN: 2072-4292. [Online]. Available: https://www.mdpi.com/2072-4292/13/8/1429.

[113] F. Alidoost, H. Arefi, and F. Tombari, "2d image-to-3d model: Knowledge-based 3d building reconstruction (3dbr) using single aerial images and convolutional neural networks (cnns)," *Remote Sensing*, vol. 11, no. 19, 2019, ISSN: 2072-4292. [Online]. Available: https://www.mdpi.com/2072-4292/11/19/2219.

[114] F. Alidoost, H. Arefi, and M. Hahn, "Y-shaped convolutional neural network for 3d roof elements extraction to reconstruct building models from a single aerial image," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. V-2-2020, pp. 321–328, 2020. DOI: 10.5194/isprs-annals-V-2-2020-321-2020. [Online]. Available: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-2-2020/321/2020/.

[115] J. Mahmud, T. Price, A. Bapat, and J.-M. Frahm, "Boundary-aware 3d building reconstruction from a single overhead image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[116] Y. Wang, S. Zorzi, and K. Bittner, "Machine-learned 3d building vectorization from satellite imagery," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 1072–1081.

[117] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[118] Y. Zheng and Q. Weng, "Model-driven reconstruction of 3-d buildings using lidar data," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 7, pp. 1541–1545, 2015. DOI: 10.1109/LGRS.2015.2412535.

[119] T. Partovi, H. Huang, T. Krauß, H. Mayer, and P. Reinartz, "Statistical building roof reconstruction from worldview-2 stereo imagery," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-3/W2, pp. 161–167, Mar. 2015. DOI: 10.5194/isprsarchives-XL-3-W2-161-2015.

[120] T. Partovi, F. Fraundorfer, S. Azimi, D. Marmanis, and P. Reinartz, "Roof type selection based on patch-based classification using deep learning for high resolution satellite imagery," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-1/W1, pp. 653–657, May 2017. DOI: 10.5194/isprs-archives-XLII-1-W1-653-2017.

[121] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[122] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[123] X. Zhang, C. May, and D. G. Aliaga, "Synthesis and completion of facades from satellite imagery," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 573–588, ISBN: 978-3-030-58536-5.

[124] A. Cohen, A. G. Schwing, and M. Pollefeys, "Efficient structured parsing of facades using dynamic programming," in *IEEE Computer Vision and Pattern Recognition*, 2014, pp. 3206–3213.

[125] H. Riemenschneider, U. Krispel, W. Thaller, *et al.*, "Irregular lattices for complex shape grammar facade parsing," in *IEEE Computer Vision and Pattern Recognition*, 2012, pp. 1640–1647.

[126] C. Yang, T. Han, L. Quan, and C.-L. Tai, "Parsing façade with rank-one approximation," in *IEEE Computer Vision and Pattern Recognition*, 2012, pp. 1720–1727.

[127] A. Martinovic and L. Van Gool, "Bayesian grammar learning for inverse procedural modeling," in *IEEE Computer Vision and Pattern Recognition*, 2013, pp. 201–208.

[128] R. Gadde, R. Marlet, and N. Paragios, "Learning grammars for architecture-specific facade parsing," *International Journal of Computer Vision*, vol. 117, no. 3, 2016.

[129] H. Liu, J. Zhang, J. Zhu, and S. C. H. Hoi, "Deepfacade: A deep learning approach to facade parsing," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017.

[130] R. Fathalla and G. Vogiatzis, "A deep learning pipeline for semantic facade segmentation," in *Proceedings of the British Machine Vision Conference 2016, BMVC 2017, September, 2017*, c 2017. The copyright of this document resides with its authors. It may be distributed unchanged freely in print or electronic forms., Sep. 2017. [Online]. Available: http://publications.aston.ac.uk/id/eprint/31805/.

[131] M. Mathias, A. Martinović, and L. Van Gool, "Atlas: A three-layered approach to facade parsing," *International Journal of Computer Vision*, vol. 118, no. 1, 2016.

[132] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[133] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," *CoRR*, vol. abs/1801.07892, 2018. arXiv: 1801.07892. [Online]. Available: http://arxiv.org/abs/1801.07892.

[134] C. Zheng, T.-J. Cham, and J. Cai, "Pluralistic image completion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1438–1447.

[135] Y. Sasaki, "The truth of the f-measure," *Teach Tutor Mater*, Jan. 2007.

[136] K. R. Prabhakar, V. S. Srikar, and R. V. Babu, "Deepfuse: A deep unsupervised approach for exposure fusion with extreme exposure image pairs," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[137] D. Joo, D. Kim, and J. Kim, "Generating a fusion image: One's identity and another's shape," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[138] Y. Li, J. Zhang, Y. Cheng, K. Huang, and T. Tan, "Df2net: Discriminative feature learning and fusion network for rgb-d indoor scene classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[139] M. C. Trinidad, R. Martin-Brualla, F. Kainz, and J. Kontkanen, "Multi-view image fusion," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[140] S. Li, C. Zou, Y. Li, X. Zhao, and Y. Gao, "Attention-based multi-modal fusion network for semantic scene completion," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020.

[141] M. Zhu, P. Pan, W. Chen, and Y. Yang, "EEMEFN: low-light image enhancement via edge-enhanced multi-exposure fusion network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[142] K. Olszewski, S. Tulyakov, O. Woodford, H. Li, and L. Luo, "Transformable bottleneck networks," *The IEEE International Conference on Computer Vision (ICCV)*, Nov. 2019.

[143] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[144] T. Yu, Z. Guo, X. Jin, *et al.*, "Region normalization for image inpainting.," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[145] Y. S. Jie Yang Zhiquan Qi, "Learning to incorporate structure knowledge for image inpainting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Feb. 2020.

[146] R. Tyleček and R. Šára, "Spatial pattern templates for recognition of objects with regular structure," in *Proc. GCPR*, Saarbrucken, Germany, 2013.

[147] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios, "Shape grammar parsing via reinforcement learning," in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[148] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, Jun. 2015.

[149] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Sep. 2018.

[150] F. Liu, X. Deng, Y.-K. Lai, Y.-J. Liu, C. Ma, and H. Wang, "Sketchgan: Joint sketch completion and recognition with generative adversarial network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[151] A. Ghosh, R. Zhang, P. K. Dokania, *et al.*, "Interactive sketch & fill: Multiclass sketch-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2019.

[152] H. Lin, Y. Fu, X. Xue, and Y. Jiang, "Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[153] G. Su, Y. Qi, K. Pang, J. Yang, and Y.-Z. Song, "Sketchhealer: A graph-to-sequence network for recreating partial human sketches," in *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*, 2020.

[154] Y. J. Lee, C. L. Zitnick, and M. F. Cohen, "Shadowdraw: Real-time user guidance for freehand drawing," *ACM Trans. Graph.*, 2011.

[155] X. S. Poma, E. Riba, and A. Sappa, "Dense extreme inception network: Towards a robust cnn model for edge detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020.

[156] A. Middel, J. Lukasczyk, R. Maciejewski, M. Demuzere, and M. Roth, "Sky view factor footprints for urban climate modeling," English (US), *Urban Climate*, vol. 25, pp. 120–134, Sep. 2018, ISSN: 2212-0955. DOI: 10.1016/j.uclim.2018.05.004.

[157] M. Rothermel, N. Haala, and D. Fritsch, "A median-based depthmap fusion strategy for the generation of oriented points.," *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 3, no. 3, 2016.

[158] M. Rothermel, K. Wenzel, D. Fritsch, and N. Haala, "Sure: Photogrammetric surface reconstruction from imagery," in *Proceedings LC3D Workshop, Berlin*, vol. 8, 2012, p. 2.

[159] M. Bosch, A. Leichtman, D. Chilcott, H. Goldberg, and M. Brown, "Metric evaluation pipeline for 3d modeling of urban scenes," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 239–246, May 2017.

[160] H. Zeng, J. Wu, and Y. Furukawa, "Neural procedural reconstruction for residential buildings," in *The European Conference on Computer Vision (ECCV)*, Sep. 2018.

[161] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216, ISBN: 0-89791-896-7. DOI: 10.1145/258734.258849. [Online]. Available: https://doi.org/10.1145/258734.258849.

[162] C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet, "Evaluation of automatic road extraction," *Inter. Arch. Photogramm. Remote Sens.*, vol. 32, Oct. 1997.

[163] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19, Nice, France: ACM, 2019, ISBN: 978-1-4503-6889-6/19/10. DOI: 10.1145/3343031.3350535. [Online]. Available: https://doi.org/10.1145/3343031.3350535.

[164] ISPRS, *2d semantic labeling contest-potsdam*, 2019. [Online]. Available: http://www2.isprs.org/commissions/%20comm3/wg4/2d-sem-label-potsdam.html.

[165] H. Jiang, L. Nan, D. Yan, W. Dong, X. Zhang, and P. Wonka, "Automatic constraint detection for 2d layout regularization," *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[166] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and Locally Consistent Image Completion," *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, vol. 36, no. 4, 107:1–107:14, 2017.

[167] A. Mustafa, X. Zhang, D. G. Aliaga, *et al.*, "Procedural generation of flood-sensitive urban layouts," *Environment and Planning B: Urban Analytics and City Science*, vol. 47, no. 5, pp. 889–911, 2020. DOI: 10.1177/2399808318812458. eprint: https://doi.org/10.1177/2399808318812458. [Online]. Available: https://doi.org/10.1177/2399808318812458.

[168] J. Ching, D. Aliaga, G. Mills, *et al.*, "Pathway using wudapt's digital synthetic city tool towards generating urban canopy parameters for multi-scale urban atmospheric modeling," *Urban Climate*, vol. 28, p. 100 459, 2019, ISSN: 2212-0955. DOI: https://doi.org/10.1016/j.uclim.2019.100459. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212095519300975.

[169] J. Ching, G. Mills, D. Aliaga, *et al.*, "The wudapt approach towards supporting multi-scale fit for purpose intra-urban atmospheric modeling and analyses applications," Jan. 2020.

[170] P. Patel, S. Jamshidi, R. Nadimpalli, *et al.*, "Modeling Large-Scale Heatwave by Incorporating Enhanced Urban Representation," *Journal of Geophysical Research (Atmospheres)*, vol. 127, no. 2, e35316, e35316, Jan. 2022. DOI: 10.1029/2021JD035316.

[171] P. Patel, S. Karmakar, S. Ghosh, D. G. Aliaga, and D. Niyogi, "Impact of green roofs on heavy rainfall in tropical, coastal urban area," *Environmental Research Letters*, vol. 16, no. 7, p. 074 051, Jul. 2021. DOI: 10.1088/1748-9326/ac1011.

A. 3D CITY MODELS

(a) Chicago - Synthetic

(b) Chicago - Aerial

(c) New Orleans - Synthetic

(d) New Orleans - Aerial

(e) San Francisco - Synthetic

(f) San Francisco - Aerial

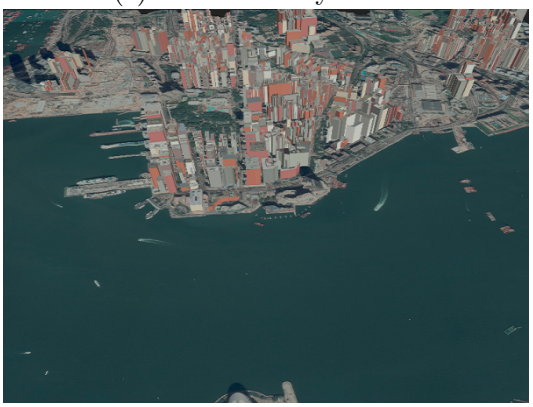
Figure A.1. *Example Models.* We show several 3D urban procedural models and similar views using Google Earth output. We used a global color remapping tool to make the color schemes similar.

(a) Dublin - Synthetic

(b) Dublin - Aerial

(c) Toulouse - Synthetic

(d) Toulouse - Aerial



(e) Hong Kong - Synthetic

(f) Hong Kong - Aerial

Figure A.2. *Additional Example Models.* Similar to previous figure, We show several 3D urban procedural models and similar views using Google Earth output.

B. 3D BUILDING MODELS

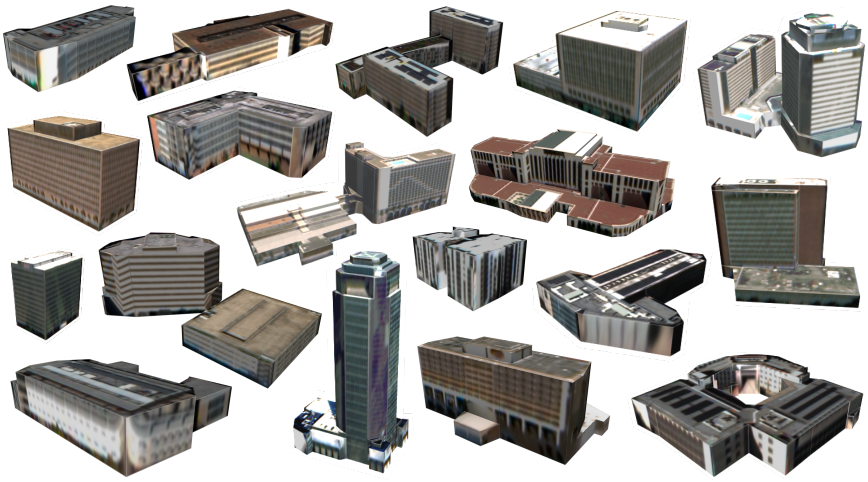


Figure B.1. *Examples.* We show close-ups of our buildings using projective texture mapping.

C. ROOF MODELS

Figure C.1. *More examples.* (a) Input urban areas from SpaceNet, CrowdAI and Urban3D respectively. (b) The inital segmentation of (a). (c) Our decomposed roof parts and predicted ridges. (d) Our procedural outputs on top of real image (a).

D. FACADE MODELS

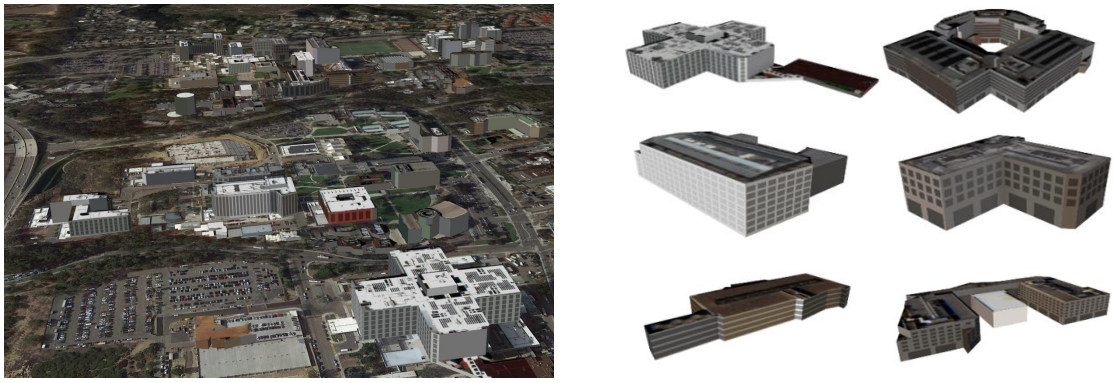


Figure D.1. *Examples.* We show a view of a reconstructed area A2 within Google Earth and close-ups of our buildings.