BuildingSketch: Freehand Mid-Air Sketching for Building Modeling

Zhihao Liu[1,2*], Fanxing Zhang[1*], Zhanglin Cheng[1,2†]

[1] Shenzhen VisuCA Key Lab, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[2] Unversity of Chinese Academy of Sciences

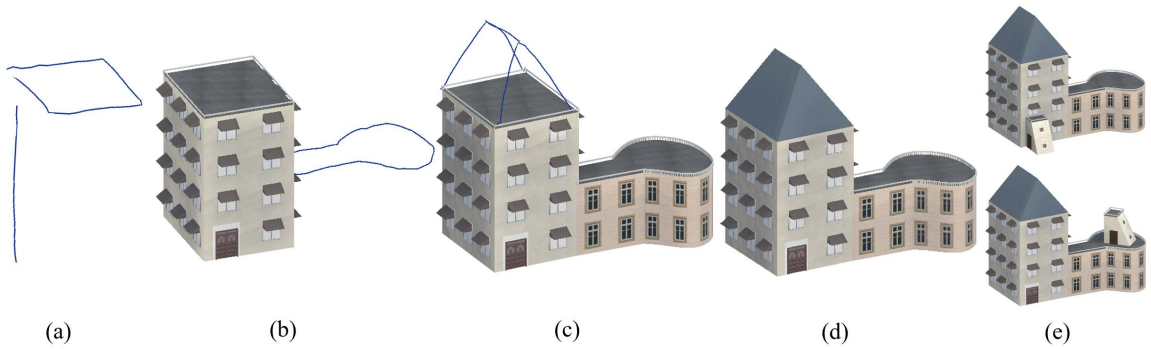


Figure 1: Structured building modeling via BuildingSketch. (a) The user draws key strokes to outline the desired building block. (b-d) BuildingSketch generates a detailed mesh model using procedural modeling based on the automatic interpretation of drawn strokes, and then the user draws new strokes to create more building blocks. (e) Created building blocks are freely combined together by the user after moving, and rotating to form a complex building model.

ABSTRACT

Advancements in virtual reality (VR) technology enable us to rethink the way of interactive 3D modeling - intuitively creating 3D content directly in 3D space. However, conventional VR-based modeling is laborious and tedious to generate a detailed 3D model in full manual mode since users need to carefully draw almost the entire surface. In this paper, we present a freehand mid-air sketching system with the aid of deep learning techniques for modeling structured buildings, where the user freely draws a few key strokes in mid-air using his/her fingers to represent the desired shapes and our system automatically interprets the strokes using a deep neural network and generates a detailed building model based on a procedural modeling method. After creating several building blocks one by one, the user can freely move, rotate, and combine the blocks to form a complex building model. We demonstrate the ease of use for novice users, effectiveness, and efficiency of our sketching system, BuildingSketch, by presenting a variety of building models.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Computing methodologies—Computer graphics—Shape modeling

1 INTRODUCTION

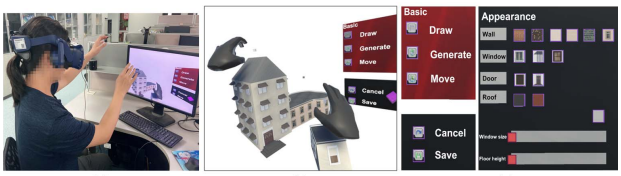
Buildings are the most common man-made objects in the world, and constructing 3D building models is important in a variety of applications such as movies, games, virtual reality, and urban planning, etc. As a result, a number of building modeling methods have been developed in the last decades. These methods can be roughly differentiated into three types: procedural building modeling [42, 47, 57, 67], data-driven building reconstruction [36, 43, 63, 68], and interactive building modeling [11, 44, 46]. Data-driven methods reconstruct 3D building models from aerial images or laser scanning point clouds that are very expensive to capture. In contrast, procedural modeling creates homogeneous models from well-defined grammars or rules that are difficult to design even for expert users. Interactive building modeling provides an intuitive way to generate 3D building models, which is more suitable for the conceptual design of a building. However, existing professional interactive modeling tools, such as Maya, 3ds Max, and blender, etc., are designed for expert users with a steep learning curve and suffer from complex and tedious operations during the modeling process. And most interactive methods for building modeling are based on inputs (sketching or drawing) on 2D displays like tablets or desktop screens, which require careful sketching and are time-consuming to generate a detailed building model.

Advancements in VR have made it possible to create and view 3D content directly in 3D space, which enables us to rethink the approaches of 3D modeling. Recent work and commercial applications like Surfacebrush [54], Tilt Brush [21], Quill [17] explores 3D drawing or sketching to create general objects using VR controller and headsets. However, all these methods are laborious and tedious to generate a detailed 3D model in full manual mode since users need to carefully draw almost the entire surface of the model. In addition, 3D drawing in mid-air is difficult to be accurate, especially by users with low spatial ability [4, 5], which often hampers the creation of delicate models like highly structured buildings. Rapid 3D building modeling for novice users with minimum and intuitive input is demanded in ideation and conceptual design [16].

In this paper, we focus on the problem of progressively creating user-desired models by novice users under VR platforms and present an efficient and easy-to-use mid-air 3D sketching system for modeling buildings. While Some VR sketching systems [2, 15] adopt tablets as physical proxies to improve the drawing accuracy, a user study carried out by Drey et al. [15] showed that users preferred to operate with 3D mid-air sketching. So freehand finger sketching in mid-air is adopted as the interactive mode in our system, which

* Z. Liu and F. Zhang are joint first authors and contribute equally.
† Z. Cheng is the corresponding author.
Email: liuzh96@outlook.com, {fx.zhang, zl.cheng}@siat.ac.cn

978-1-6654-0158-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ISMAR52148.2021.00049



(a) (b) (c)

Figure 2: Hardware and system configuration. (a) Hardware setup. The user wears an HTC Vive VR headset attached with a Leap Motion controller. (b) The workspace from user's perspective. (c) The control panel for switching operations and adjusting the building appearance.

ensures that users can draw freely in a very natural way and maximize their creative liberty. Since a building can be decomposed as a combination of several simple primitive shapes including building masses, roofs, as well as stairs, the user only needs to create some primitives one by one and then freely move and rotate the primitives to form a complex building model. In order to avoid tedious drawing of the whole shape during the primitive creation, the user only needs to roughly sketch a few key strokes in mid-air to outline the primitive shape instead of drawing complete strokes as done in existing works. Our system automatically translates the incomplete 3d strokes to primitive type (i.e., building mass, roof, stair) using a deep neural network, and then a procedural modeling method is adopted to generate a detailed mesh model of the primitive in real-time.

In summary, the main contributions of this work are:

- We propose a novel freehand mid-air sketching interface for novice users to design 3D building models intuitively. To the best of our knowledge, this is the first exploration of VR-based interface for rapid building modeling.
- We combine rough 3D sketching and deep learning to minimize the burden and accuracy requirements for mid-air sketching, which frees users from low-level detail input to focus on the overall building shape design.
- We decompose the 3D building creation task into progressive building block design and combination problems.

2 Related Works

Modeling of buildings. Procedural modeling has become a powerful framework for building generation since the pioneering work of Parish et al. [47], and several follow-up studies [37, 42, 57, 67]. They utilize a set of procedural grammars or rules to produce a wide variety of building models. However, those methods are based on text rules, and are hard to use for novice users without grammar expertise and architectural background. Even for professional modelers, designing the proper grammar for creating specific shapes remains a challenge. To circumvent this, Lipp et al. [65] introduce a visual editing tool to visually create a simple rule base for shape grammars. The editing process, however, is still tedious and not intuitive because it is implemented in the form of a tree table. The users need to drag a number of components into the tree table to form the complete buildings. On the other hand, data-driven synthesis is a recent trend in building modeling, which constructs 3D building models from real-world data such as photographs [28, 73] and 3D point clouds [8, 55]. However, such methods are devised for the reconstruction of real buildings. Modeling a virtual building from scratch is beyond their scope.

Sketch based modeling. Sketches have been widely exploited for 3D modeling due to flexibility. The famous Teddy system [31] was one of the first attempts to construct 3D models from 2D sketches. However, the method can only produce circular and symmetric shapes that are just inflated from 2D silhouettes. Similar inflation methods were adopted by MagicToon [18] and TreeSketch [40] to create cartoon models or trees with flat shapes from 2D sketches. In addition, many approaches have been proposed to infer more complex 3D shapes from sketches, based on probabilistic optimization [12], deep learning [24] and retrieval [38, 69]. Please refer to [6] for a great survey on sketch-based modeling.

There also exist several works dedicated to architecture modeling from 2D sketches. The systems in [58, 62] allow the users to draw outlines overlaid on a collection of 2D photographs, and then generate the corresponding 3D building by combining the 2D sketches with multi-view stereo algorithms. Chen et al. [11] generate 2.5D architectural geometries from freehand sketches based on topological analysis. More recently, Nishida et al. [44] and Chen el al. [10] leverage deep learning methods to reconstruct buildings from 2D strokes. However, all these methods are based on 2D sketch input and suffer from the need for predicting plausible 3D depths. Direct 3D control for geometries is beyond their capability.

3D modeling in VR. Advancement in VR technology enables us to dive in and sketch directly in mid-air, and therefore researchers can rethink the modeling tasks in a fresh way, for example using direct 3D interactions. Commercial systems like Oculus Medium [45] and ShapeLab [60] offer a variety of powerful sculpting tools for creating shapes from simple forms to complex designs. Other systems [23, 33, 54, 56, 66, 71] enable the users to draw a range of ribbons, wires, or sweeping surfaces by moving a hand-held controller in 3D space. Recent works [2,30,32] also try to improve the expressive qualities of 3D sketching by combining other approaches like generative design and 2D interactions. However, all these methods are laborious since the modelers have to carefully sketch the full details of the target. More importantly, accurately drawing the intended strokes in VR remains a challenge due to the inaccuracy in depth [1] and the lack of user spatial ability [4, 5]. Recently, a number of methods aim to help lower the difficulty of modeling by using feature lines [48], gesture recognition [7, 13, 29], plane constraint [3] and shape retrieval [20]. However, it is still difficult for these methods to create objects like buildings that have intricate details and artificial structures.

Several systems have been proposed to simplify the interactions for modeling specific objects such as 3D trees [39, 70, 72]. They enhance the 3D sketching with different automatic synthesis algorithms to produce high quality 3D tree models by only sketching the overall tree shapes. Inspired by those approaches, our system for 3D building modeling uses freehand 3D sketching to roughly draw the shape of the building, and then obtains a detailed 3D mesh model based on the user's sketches through deep learning and procedural generation methods.

Deep learning. Deep learning has made great success in many domains, including object recognition [35, 61], object detection [25, 52, 53], natural language processing [14, 64], etc. In addition, recent works devised a number of neural networks for processing 3D data. For example, PointNet [50] and its variations [49, 51] have successfully solved the classification and segmentation problems for point sets.

3 Overview

Fig. 2 (a,b) shows the hardware environment and user workspace of BuildingSketch. Wearing a VR headset, the user freely draws 3D strokes in mid-air using his/her fingers, while the finger movements are captured directly by a Leap Motion controller attached to the front of the VR headset. Fig. 2 (c) shows the control panel with a couple of buttons that are frequently used while designing buildings. The three buttons on the *Basic* panel are used for switching between different operation modes: starting/finishing drawing, generating geometry model, and moving building objects. In drawing mode, the user draws strokes using a pinch gesture. The system distinguishes whether the user is drawing a stroke or just moving hands by the

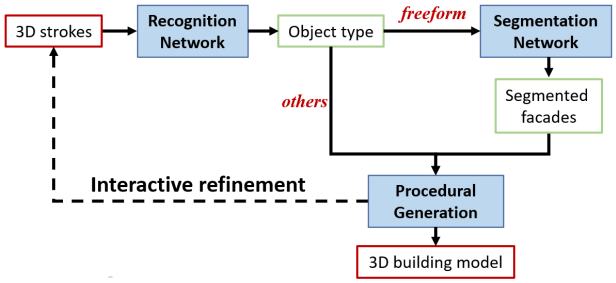


Figure 3: Pipeline of our method. A recognition network is used to identify the object type from user-drawn strokes. The detailed geometry is then synthesized by procedural generation according to the predicted type as well as the 3D shape of strokes. Note that if the strokes are recognized as 'freeform', they will be divided into straight lines and curves using a segmentation network.

distance between the thumb and index finger. If the distance is less than a tiny threshold, the system knows the user is drawing a stroke. If a recently drawn stroke is wrong or bad, the user can delete it by pressing the *Cancel* button. When the *Generate* button is pressed, the strokes are interpreted by the neural network automatically and then the building geometry is created in real time. When the *Move* button is pressed, the user can interact with the gray widgets attached to each building object to move and rotate them. Furthermore, the *Appearance* panel provides options to adjust the building appearance and procedural parameters, such as wall texture, window type, and floor height, etc.

BuildingSketch is designed to generate building models from very few 3D strokes, and Fig. 3 illustrates the pipeline of our 3D-sketch-based building modeling approach. The user first sketches a few 3D strokes to outline the shapes of the desired building objects. To interpret the shapes they draw, we use a deep neural network to recognize object types (i.e., building mass, roof, stair) from the strokes. The detailed building models are then synthesized using a procedural modeling method according to the predicted types and stroke shapes. A number of object types are supported, including different building masses, roofs, as well as stairs (see Fig. 4). The users can arbitrarily place and assemble those objects to produce buildings with more complex structures. In addition, BuildingSketch also allows the user to model freeform building masses (Fig. 4 (c)), by sketching irregular strokes to specify the shape of top floor. To achieve this, if current strokes are recognized as a freeform building mass, a PointNet is utilized to segment the stroke to get a set of planar and curved facades that together form the whole freeform building mass.

4 Building Creation from 3D Strokes

4.1 Object Recognition from Strokes

A stroke is a sequence of successive 3D points with variable length, which can be considered as spacetime data. A lot of work has been done to solve the recognition problem through CNNs [35,52,53,61], but these methods mainly deal with 2D images and the input size is fixed once the network is trained. Therefore, they are not suitable for 3D strokes.

Recently, recurrent neural networks (RNN) have become one of the most common choices to process and classify the time series data, and are suitable to tasks such as handwriting and speech recognition [22,41]. Therefore, we designed an RNNs-based recognition network to infer the intended object types from 3D strokes.

The network takes a varying number of strokes as input, and outputs the best matching object type. In our implementation, we

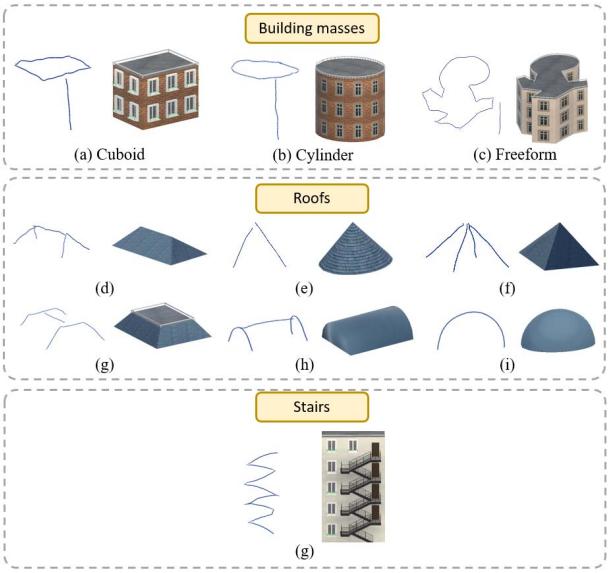


Figure 4: Predefined types of building blocks and corresponding key strokes. For each type, we show the user-drawn strokes and the corresponding 3D models.

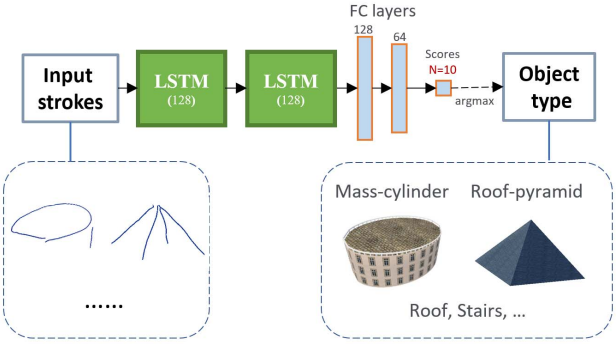


Figure 5: Structure of the recognition network for recognizing the object type from user-drawn strokes.

first concatenate current strokes into a single sequence of points (x_i, y_i, z_i, s_i), where (x_i, y_i, z_i) is the coordinate of ith point, and $s_i = 0, 1, 2, ...$ indicates on which stroke this point is originally located. As a preprocessing, we normalize the points to be zero mean and within a unit sphere before feeding them to the recognition network. The output of network is a N-dimensional vector (N=10), which represents the classification scores of the N categories. By default, the object type with the highest score is selected to generate corresponding geometry.

Fig. 5 illustrates in detail the structure of our recognition network. The network consists of common neural blocks, including LSTMs [26] for processing variable-length input, and fully connected Layers (FCs) for regression. Here, the LSTM is a specific RNN structure that can solve long time lag problems better than conventional RNNs [27]. Each layer is followed by leaky ReLU as the activation function except the output layer utilizes the sigmoid activation to get the final results. We adopt the categorical cross entropy as the loss function which is widely used in multi-class classification problem, that is,

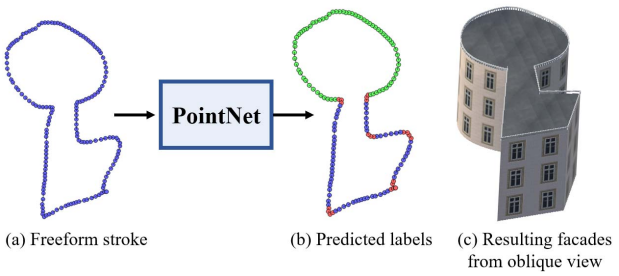


Figure 6: Segmentation of freeform strokes using PointNet. (a) A freeform stroke that depicts the top floor of a building. (b) The segmentation results using PointNet (green: curve-points, blue: line-points, red: corner-points). (c) The generated building mass.

$$L_{cce} = -\sum_{i=1}^{N} p_i \cdot \log \hat{p}_i \quad (1)$$

where p_i and $\hat{p}_i$ are the ground-truth and the predicted score for each class i. During the training process, we use the Adam optimizer [34] with an initial learning rate of 0.001 and a batch size of 16 to optimize our network.

4.2 Freeform Stroke Segmentation

While creating a building mass, the user is able to sketch a stroke to specify the shape of its top floor, and optionally draw several downward strokes to depict the facades (details are introduced in Sec. 4.3). To improve expressiveness, BuildingSketch allows users to design freeform building masses by sketching the top strokes with irregular shapes (see Fig. 4 (c)). However, unlike other object types that have regular geometric structures and can be easily fitted, the top strokes of freeform building masses are usually arbitrarily shaped. It is difficult to form the corresponding facades directly from such strokes. To solve the problem, we assume a freeform stroke can be decomposed into a set of relatively regular segments, i.e, straight lines and curves. Therefore, the top strokes are first automatically divided into smaller segments, from each of which the facades are subsequently generated and together form the final building mass.

PointNet [50] is a highly efficient and effective end-to-end network for point set segmentation, and thus it is well-suited for our task by considering a stroke as a set of 3D points. We classify the points into 3 categories according to their local features, i.e, the corner-points, curve-points and line-points. We basically adopt the original PointNet architecture. The network is set to handle up to 500 points with zero-padding to ensure that strokes have the same number of points. Our network will output 500×3 scores for each of the 500 points and each of the 3 categories. Fig. 6 (b) shows an example segmented result by PointNet. The corner-points, curve-points and line-points are marked in red, green and blue, respectively.

After segmentation, we can subsequently form the top surface of a freeform building mass. Specifically, the successive line-points segment are directly fitted into a straight line, while the curve-points segment are smoothed and resampled using a cubic hermite spline. As a result, the corresponding building mass is shown in Fig. 6 (c).

4.3 Building Construction

Once the object type of current strokes is identified by the recognition network, our method automatically synthesizes the detailed geometry that matches the shape of current strokes. The geometrical parameters, such as the height and tilt angle of facades and roofs, are obtained directly from the 3D strokes. Afterward, windows and other ornaments are automatically generated based on a collection of simple procedural grammars [42].

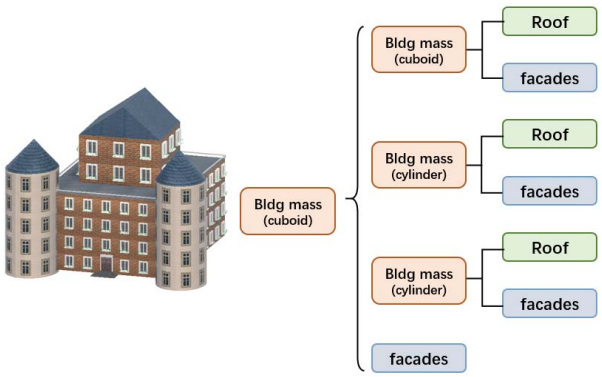


Figure 7: The hierarchical tree of an example building.

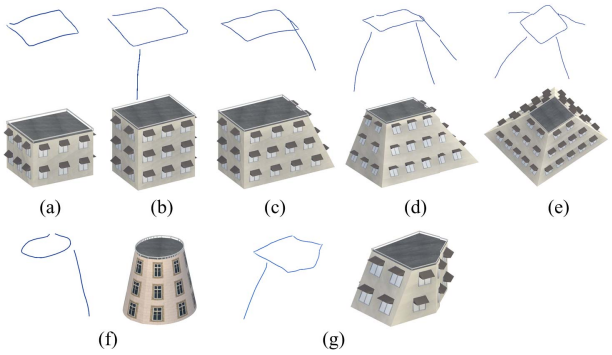


Figure 8: Downward strokes can be used to specify the height and tilt angle of facades

.

To better manage and assemble multiple building parts, we describe the buildings using the hierarchical trees (see Fig. 7), where the root is usually the first sketched object. Once the user sketches a new object or moves an existing one, the tree structure is updated immediately according to the current adjacency. Two building parts will become a group when the minimum distance between their surfaces is less than a tiny threshold. A snapping support is also provided to automatically snap together their neighboring surfaces. Moreover, it's easy for BuildingSketch to model a building complex using multiple hierarchical trees. A new tree is created if no adjacent objects exist around a new stroke.

In the following, we illustrate in detail the geometry generation for each object type.

Building mass. Our current implementation includes three types of building masses, i.e., the cuboid, cylinder, and freeform objects (the first row in Fig. 4). The users are allowed to sketch a stroke to depict the shape of the top floor, and optionally draw several downward strokes to specify the height and tilt angle for each facade. The drawing order of strokes can be arbitrary, since the top stroke can be directly located according to their Y-positions.

The top floor of the building mass is constructed from the user-drawn top stroke through planar shape fitting. The top stroke points are first projected onto the horizontal plane, and then fitted by a rectangle, an ellipse, or straight lines and cubic hermite splines for cuboid, cylinder, or freeform-shaped building mass, respectively, based on the stroke type. For the cuboid-shaped building mass, we perform the rotating calipers algorithm [59] for its top stroke

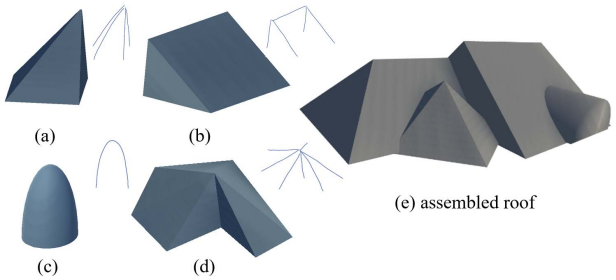


Figure 9: Variations of the building roof generated from different strokes and roof combinations. (a-d) Four roofs generated from different 3D strokes. (e) A complex roof composed of a set of primitive roofs

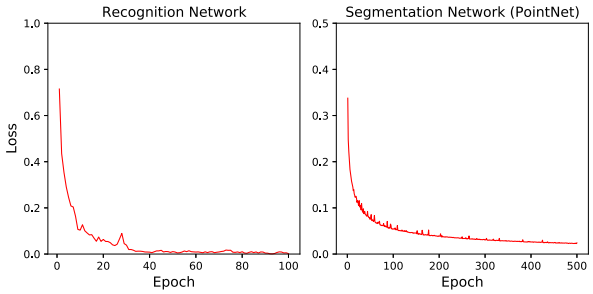


Figure 10: Loss curves of our recognition and segmentation networks.

to obtain the minimum area rectangle as the top surface. For the cylinder-shaped building mass, its top surface is generated directly through ellipse fitting based on the least squared algorithm [19]. For freeform-shaped building mass, the top stroke is fitted by lines and splines for each segment as discussed in Sec. 4.2.

The building facades are constructed from downward strokes. Fig. 8 shows the effect of using different number of downward strokes. In the simplest case, if the users only sketch the top stroke, the resulting facades are vertical, and the building height is set to reach the ground or the top floor of a lower building (see. Fig. 8 (a)). Moreover, as shown in Fig. 8 (b-e), the potential downward strokes can be used to create more facade variations, for example the tilted facades. The facade is created by fitting a plane to the downward stroke and it's nearest top stroke segment, and the building height is computed as the average height of all downward strokes.

Roof. As shown in Fig. 4 (d-i), BuildingSketch supports multiple roof types. For each type, we perform specific drawing style to depict its overall shape feature, which can be effectively identified by our recognition network. For example, to create a conical roof (Fig. 4 (e)), we use two downward strokes to draw its lateral surface, and the corresponding roof geometry is then generated according to 3D stroke structures. The users don't need to draw strictly regular or symmetric roof shapes. Instead, free drawing of strokes can help to produce more roof variations, such as the four roofs in Fig. 9 (a) to (d), which can be easily obtained by mid-air drawing from our basic roof types. Moreover, as shown in Fig. 9 (e), the users can also flexibly assemble simple roofs together to form a more complex shape.

The real roofs are generally associated with the top floor of the building mass. Therefore, the bottom of roof will be adjusted automatically to fit the shape of the building below, if they are closed enough and meanwhile their two object types are compatible, such as the cuboid building with the hip roof, or the cylinder building with the dome roof. In other cases where the two object types are not consistent with each other, we simply attract the bottom points in roof to the neighboring building edges if their distance is within a small threshold.

Stairs. The staircases are created in a simple way. Fig. 4 (g) shows an example of the user drawn stroke and its resulting stairs. We use wave strokes to represent the stairs, and only generate them closely attached to the facades. Note that the wavy shape of the stroke is only used to indicate its type, and the actual number of turns in the stair is the same as the floor number of the facade to which the stair is attached.

5 Results

BuildingSketch consists of two parts: a modeling client for user interaction, and a deep learning server to process the prediction requests from client. The client is implemented in C# with Unity, while the server as well as two networks are written in Python with Tensorflow. All the modeling results were obtained on a PC with 3.4 GHz i7-6800K CPU.

5.1 Network Training

Training. We first discuss the training details and performance. Offline training runs on a NVIDIA GeForce 1080 Ti GPU. To improve the learning ability, we augment the data by jittering and random rotation around up-axis during each training step. The loss curves of two networks are shown in Fig. 10.

To train the recognition network, we constructed 4.5k paired samples in total (500 for each object type). Especially for building mass data, 30% are sketched with only top face, while the rest of them have one or more vertical strokes to depict their facades. We also made 500 samples as the test set. As a result, the net has 99.8% prediction accuracy for training set and 97.6% for test set.

For PointNet, we labeled 3.4k freeform strokes as the training set, and 300 as test set. The final segmentation accuracy for training and test set are 96.9% and 95.2%, respectively. Fig. 12 shows several examples of segmentation and resulting models. Most of points are accurately segmented, which demonstrates the robustness of our network. In addition, to reduce the effect of wrong prediction, we perform the noise removal by directly discarding the small line and curve segments with less than 3 points.

It takes about 1h and 6h for training the recognition network and the PointNet, respectively. However, even though they may require many hours for training, once the networks are trained, the recognition and segmentation for strokes can be done within a second (average 150ms). Therefore, they are very suitable for real-time interaction.

5.2 Modeling Results

In this section, we conducted several experiments to demonstrate the modeling abilities of BuildingSketch.

As shown in Fig. 11, we create a variety of complex shaped buildings by making full use of our provided functions. All results are assembled from multiple building parts which are progressively sketched by mid-air sketching. Among them, we randomly create freeform building masses, and choose objects to make tilted facades. Note that the green building in (c) and the cross-shaped one in (e) are impossible for existing 2D sketch based systems [10, 44] to create, since their shapes heavily rely on direct 3D interaction and freeform sketching. Moreover, the arbitrary combination of objects in (d) and (g) also demonstrates the flexibility of our 3D interaction.

In Fig. 13, we also tried to reproduce a set of buildings by using real photographs. In terms of overall shapes, the resulting models bear good resemblance with target photos. Even for the building with radical shape in (e), our method can effectively recover its tilted facades and complex topology by 3D interactions.

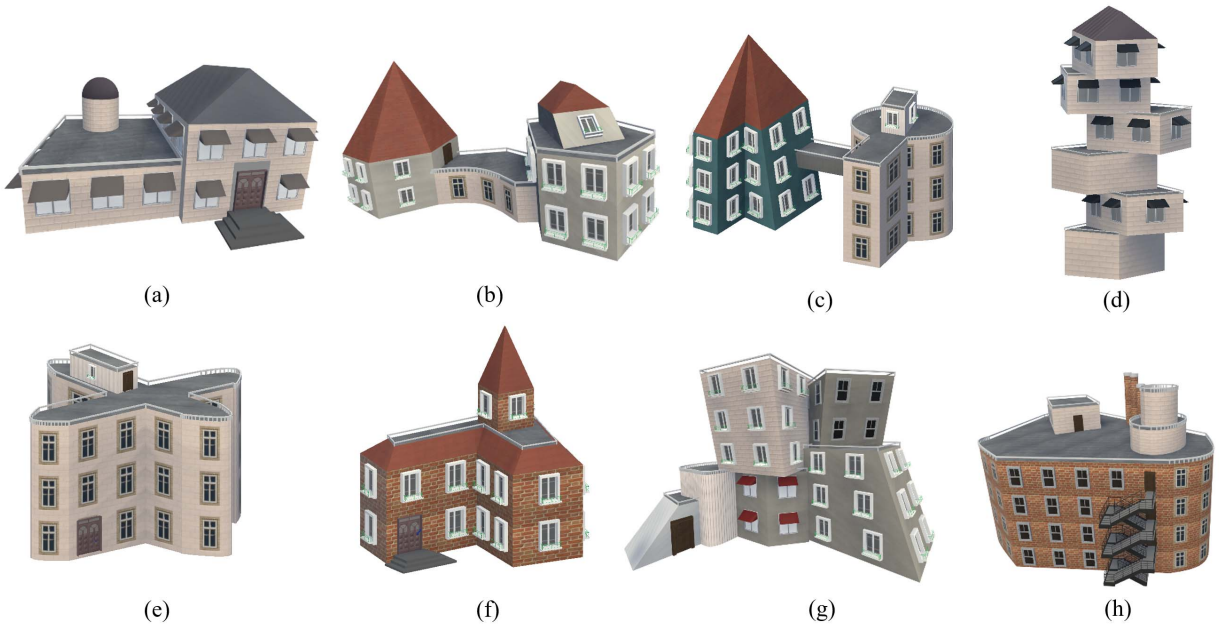


Figure 11: Example buildings created using BuildingSketch.

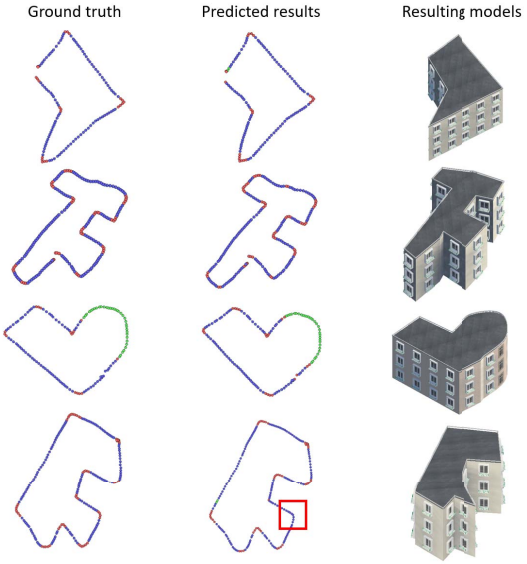


Figure 12: Stroke segmentation results and corresponding 3D buildings. The last row shows a case where a corner is incorrectly recognized as a line-point.

In addition, due to the benefit of 3D interaction, our method enables the users to adopt different combinations to obtain more variations. Fig. 14 shows several buildings that are combined from the same object parts through arbitrarily moving and rotating the object parts.

Overall, each building takes about 1-6 minutes to create, which depends on the complexity of its structure. In Tab. 1, we recorded the statistics of some selected buildings. The total generation time equals the sum of time for network prediction and geometry construction.

We notice that the most time-consuming step is to design shapes and sketch strokes, especially for the buildings that consist of a large number of building blocks. However, it's real-time to form the geometry for each object part without taking into account the user's thinking and sketching time.

Table 1: Statistics of several buildings. Note that the total time counts the whole process including interactions, while the total computation time is only the sum of geometry generation time for each object.

	Number of strokes	Total time [min]	Total computation time [sec]
Fig. 11 (a)	9	2.6	0.74
Fig. 11 (b)	16	3.2	0.93
Fig. 11 (c)	15	4.6	0.86
Fig. 11 (e)	3	0.73	0.45
Fig. 11 (g)	13	4.1	1.12

5.3 Comparison

Traditional procedural methods [42,57] automatically generate buildings mainly based on rules and grammars. However, they require the users to equip with expertise knowledge of architecture. For novice users, it's difficult to write proper rules to obtain desired buildings. Moreover, the generation process is fully automatic and thus hard for users to control and interfere. Our method, however, provides easy-to-use 3D interactions for users to design buildings without too much effort.

We then compare with a recent 2D sketch-based modeling approach for buildings [44] (see Fig. 15). This method enables the users to sketch 2D strokes, and then finds a grammar that can yield similar shape from a pre-defined grammar set. However, their grammar set only supports simple basic shapes, like cube, 'L'-shape and 'H'-shape. As shown in Fig. 15 (a), their method fails when the sketched shapes do not exist in their pre-defined grammar set. Our approach, however, can successfully generate correct 3D freeform shapes from irregular strokes (see Fig. 15 (b)). Second, their method asks the users to draw from fixed viewing orientation and position,

Figure 13: Modeling results inspired by photographs.

Figure 14: Various modeling results generated by different combinations of the same set of building blocks.

and simply generates vertical facades. Our method, by contrast, supports direct 3D sketching and viewing, which makes it possible to generate tilted facades such as the one inside the dashed box of Fig. 15. Moreover, their method requires the users to sketch almost full details of their intended shape, including the bottom edges and every visible vertical line. And therefore they cannot be applied to incomplete sketches, for example the ones in Fig. 8. Their method also trained 30 CNNs in total to estimate parameters of each grammar, which can be extremely costly in terms of computational resources. However, BuildingSketch bears limitation compared to Nishida's system as well. Our current implementation uses too many buttons to adjust the appearance so that limited choices of the building styles are provided. A more expressive user interface is still necessary for appearance control in the future.

5.4 User Evaluation

To gather feedback and evaluate BuildingSketch, we recruited 10 participants (aged 20 to 32) from local university. They major in computer science and didn't have any experience of 3D modeling before. Besides briefly introducing the study goals and procedure, we also provided the tutorial regarding the usage and all functions of BuildingSketch.

First, the participants were asked to arbitrarily design buildings they desired to create (see Fig. 16 for typical example results). We conducted a short interview with them after use, and asked them to rate BuildingSketch on ten criteria using the System Usability Scale (SUS) questionnaire [9] (1=strongly disagree, 10=strongly agree). The summary results are shown in Tab. 2. As a result, we are happy to see that the participants were generally positive about our modeling system. The subjective scores indicates that BuildingSketch is easy to learn (Q7, mean=8.0, sd=0.6) and easy to use (Q3, mean=8.4,

Table 2: Subjective scores for the ten questions of the System Usability Scale (SUS) questionnaire (mean and standard deviation), with 1 being "strongly disagree" and 10 being "strongly agree"

Question	Score
(1) I would like to use this system frequently.	7.8 (0.7)
(2) The system is unnecessarily complex.	1.2 (0.5)
(3) The system is easy to use.	8.4 (0.5)
(4) I would need the support of a technical person to master the system.	2.7 (0.4)
(5) The various functions in this system are well integrated.	8.9 (0.5)
(6) There was too much inconsistency in this system.	1.7 (0.3)
(7) The system is easy to learn for most people.	8.0 (0.6)
(8) I found the system very cumbersome to use.	1.4 (0.2)
(9) I felt very confident using the system.	9.0 (0.6)
(10) I needed to learn lots of things before I could get going with this system.	1.5 (0.4)

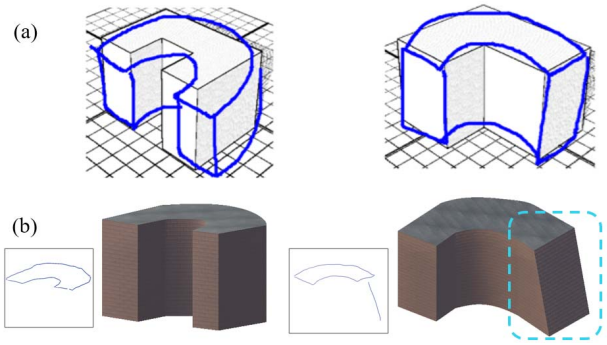


Figure 15: Comparison with Nishida's 2D-sketch-based modeling [44]. (a) Nishida's method fails when the user draw shapes with smooth curves. And it can only generate vertical facades. (b) Our approach can effectively generate freeform shapes from fewer strokes. Also, the tilted facade inside the dash box can be easily created by a downward 3D stroke.

Figure 16: Example buildings freely created by participants for user study

.

sd=0.5). All participants were satisfied with their modeling results, and stated that the 3D sketching does facilitate the easy creation of buildings. BuildingSketch provided various functions for users, and they were well integrated together (Q5, mean=8.9, sd=0.5) so that the users didn't need to pay too much for getting going with the system (Q10, mean=1.5, sd=0.4). Seven participants specially expressed excitement about the auto-recognition of the interaction intention, as well as the way to create freeform building shapes. For instance, one participant (P3) remarked, "*The intention recognition from 3D strokes impressed me the most, it effectively simplified my design process. Meanwhile I was attracted by the instantaneous display and the direct visual manipulation of the generated building models, which could ensure my interaction consistency with the system.*" Furthermore, the progressive creation of buildings with intuitive interactions seemed to make participants enjoy the modeling procedure. For example, one participant (P5) commented, "*To create buildings in such an expressive manner is like playing a game. I can create fancy buildings while having fun.*"

In addition, we also devised a simple experiment to ask participants to reconstruct a building according to a target image. Fig. 17 shows some of the resulting models. We marked each building block with different colors so as to clearly see how users create the building by combining different building blocks. In terms of overall shapes, all participants successfully created buildings that bear

Figure 17: User study excerpts. 3D models created by 4 participants according to a target building. The building blocks are marked with different colors to distinguish how different participants decompose the building.

good resemblance with the target in a short time. The results also indicated people generally tended to observe and comprehend the building shapes in different levels, and that they could freely explore diverse design ways using BuildingSketch when creating buildings. However, we also identified some usability issues of BuildingSketch. For example, one participant (P8) noted, "*It's easy for me to obtain the desired overall shapes with current system, but it provided few choices of window and wall styles, which made it difficult to fully recover all details when I tried to reconstruct an existing building from a photograph, especially for this kind of building that adopted special window patterns.*" One nearsighted participant (P1) also pointed out another problem that wearing glasses with a VR headset after a long time would make him kind of uncomfortable.

6 Conclusion and Future Work

Building modeling is motivated by a wide range of applications in computer graphics, computer vision, and virtual reality. Procedural modeling and 2D-Sketch-based modeling dominate 3D building design in current practice, where designers have to divert a lot of energy from ideation and concept design to draw details or design grammars. Recent developed VR-based modeling tools are not designed for building modeling and require tedious drawing and editing to create user-desired buildings. In this paper, we propose a novel freehand sketching interface, BuildingSketch, to create 3D buildings intuitively. BuildingSketch can generate 3D building models as the user desired progressively from very few 3D strokes with the aid of deep learning and procedural modeling methods. The user can also freely move, rotate, and combine the created models to form a new complex building. A variety of created building models and the user study show the ease of use for novice users, effectiveness, and efficiency of BuildingSketch.

Currently, BuildingSketch creates buildings from sketches and a combination of small building blocks. In the future, we will add virtual sculpting to the system to facilitate the production of more elaborate building geometries such as recessed parts as well as curved facades.

Acknowledgments

This work was supported in part by NSFC (61972388), Shenzhen Basic Research Program (JCYJ20180507182222355), the Leading Talents of Guangdong Program (00201509), and the CAS grant (GJHZ1862).

References

[1] R. Arora, R. H. Kazi, F. Anderson, T. Grossman, K. Singh, and G. W. Fitzmaurice. Experimental evaluation of sketching on surfaces in

vr. In *CHI Conference on Human Factors in Computing Systems*, pp. 5643–5654, 2017.

[2] R. Arora, R. H. Kazi, T. Grossman, G. Fitzmaurice, and K. Singh. Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2018.

[3] M. D. Barrera Machuca, P. Asente, W. Stuerzlinger, J. Lu, and B. Kim. Multiplanes: Assisted freehand vr sketching. In *The Symposium on Spatial User Interaction*, pp. 36–47, 2018.

[4] M. D. Barrera Machuca, W. Stuerzlinger, and P. Asente. The effect of spatial ability on immersive 3d drawing. In *The 2019 on Creativity and Cognition*, pp. 173–186. 2019.

[5] M. D. Barrera Machuca, W. Stuerzlinger, and P. Asente. Smart3dguides: Making unconstrained immersive 3d drawing more accurate. In *25th ACM Symposium on Virtual Reality Software and Technology*, 2019.

[6] S. Bhattacharjee and P. Chaudhuri. A survey on sketch based content creation: from the desktop to virtual and augmented reality. *Computer Graphics Forum*, 39(2):757–780, 2020.

[7] U. Bohari and T.-J. Chen. To draw or not to draw: recognizing stroke-hover intent in non-instrumented gesture-free mid-air sketching. In *23rd International Conference on Intelligent User Interfaces*, pp. 177–188, 2018.

[8] C. Brenner. Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation*, 6(3-4):187–198, 2005.

[9] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[10] C.-Y. Chen and I.-C. Lin. Rapid 3d building modeling by sketching. In *ACM SIGGRAPH 2019 Posters*, pp. 1–2. 2019.

[11] X. Chen, S. B. Kang, Y.-Q. Xu, J. Dorsey, and H.-Y. Shum. Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph*, 27(2), 2008.

[12] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based tree modeling using markov random field. *ACM Trans. Graph.*, 27(5), 2008.

[13] B. R. De Araùjo, G. Casiez, and J. A. Jorge. Mockup builder: direct 3d modeling on and above the surface in a continuous interaction space. In *Graphics Interface*, pp. 173–180, 2012.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] T. Drey, J. Gugenheimer, J. Karlbauer, M. Milo, and E. Rukzio. Vrsketchin: Exploring the design space of pen and tablet interaction for 3d sketching in virtual reality. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2020.

[16] K. Eissen and R. Steur. *Sketching: Drawing techniques for product designers (11 ed.)*. BIS Publishers, 2009.

[17] FACEBOOK. Quill. `https://quill.fb.com/`, 2019.

[18] L. Feng, X. Yang, and S. Xiao. Magictoon: A 2d-to-3d creative cartoon modeling system with mobile ar. In *IEEE VR*, pp. 195–204, 2017.

[19] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):476–480, 1999.

[20] D. Giunchi, S. James, and A. Steed. 3d sketching for interactive model retrieval in virtual reality. In *The Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pp. 1–12, 2018.

[21] GOOGLE. Tilt brush. `https://www.tiltbrush.com/`, 2017.

[22] A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pp. 273–278, 2013.

[23] GravitySketch. Gravitysketch. `https://www.gravitysketch.com/`, 2018.

[24] X. Han, C. Gao, and Y. Yu. Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling. *ACM Trans. Graph*, 36(4), 2017.

[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, pp. 2961–2969, 2017.

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] S. Hochreiter and J. Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pp. 473–479, 1997.

[28] H. Huang, M. Michelini, M. Schmitz, L. Roth, and H. Mayer. Lod3 building reconstruction from multi-source images. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43:427–434, 2020.

[29] J. Huang and R. Rai. Conceptual three-dimensional modeling using intuitive gesture-based midair three-dimensional sketching technique. *Journal of Computing and Information Science in Engineering*, 18(4):041014, 2018.

[30] K. Huo and K. Ramani. Window-shaping: 3d design ideation by creating on, borrowing from, and looking at the physical world. In *The Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*, pp. 37–45, 2017.

[31] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *SIGGRAPH*, pp. 409–416, 1999.

[32] R. H. Kazi, T. Grossman, H. Cheong, A. Hashemi, and G. W. Fitzmaurice. Dreamsketch: Early stage 3d design explorations with sketching and generative design. In *ACM Symposium on User Interface Software and Technology*, pp. 401–414, 2017.

[33] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola Jr. Cavepainting: a fully immersive 3d artistic medium and interactive experience. In *The symposium on Interactive 3D graphics*, pp. 85–93, 2001.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[36] F. Lafarge and C. Mallet. Building large urban environments from unstructured point data. In *ICCV*, pp. 1068–1075, 2011.

[37] T. Lechner, P. Ren, B. Watson, C. Brozefski, and U. Wilenski. Procedural modeling of urban land use. In *ACM SIGGRAPH Research posters*, pp. 135–es. 2006.

[38] B. Li, Y. Lu, A. Ghumman, B. Strylowski, M. Gutierrez, S. Sadiq, S. Forster, N. Feola, and T. Bugerin. 3d sketch-based 3d model retrieval. In *the 5th ACM on International Conference on Multimedia Retrieval*, pp. 555–558, 2015.

[39] Z. Liu, C. Shen, Z. Li, T. Weng, O. Deussen, Z. Cheng, and D. Wang. Interactive modeling of trees using vr devices. In *International Conference on Virtual Reality and Visualization*, pp. 69–75, 2019.

[40] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. Treesketch: Interactive procedural modeling of trees on a tablet. In *International Symposium on Sketch-Based Interfaces and Modeling*, pp. 107–120, 2012.

[41] R. Messina and J. Louradour. Segmentation-free handwritten chinese text recognition with lstm-rnn. In *13th International Conference on Document Analysis and Recognition*, pp. 171–175, 2015.

[42] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623, 2006.

[43] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. *ACM Trans. Graph*, 29(4), 2010.

[44] G. Nishida, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and A. Bousseau. Interactive sketching of urban procedural models. *ACM Trans. Graph*, 35(4), 2016.

[45] OculusMedium. Oculusmedium. `https://oculus.com/medium/`, 2016.

[46] D. J. Olsen, N. D. Pitman, S. Basak, and B. C. Wünsche. Sketch-based building modelling. In *GRAPP*, pp. 119–124, 2011.

[47] Y. I. Parish and P. Müller. Procedural modeling of cities. In *The 28th annual conference on Computer graphics and interactive techniques*, pp. 301–308, 2001.

[48] H. Perkunder, J. H. Israel, and M. Alexa. Shape modeling with sketched feature lines in immersive 3d environments. In *the Seventh Sketch-Based Interfaces and Modeling Symposium*, pp. 127–134, 2010.

[49] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, pp. 918–927, 2018.

[50] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pp.

652–660, 2017.

[51] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *International Conference on Neural Information Processing Systems*, pp. 5105–5114, 2017.

[52] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pp. 779–788, 2016.

[53] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[54] E. Rosales, J. Rodriguez, and A. Sheffer. Surfacebrush: From virtual reality drawings to manifold surfaces. *ACM Trans. Graph.*, 38(4), 2019.

[55] A. Sampath and J. Shan. Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds. *IEEE Transactions on geoscience and remote sensing*, pp. 1554–1567, 2009.

[56] S. Schkolne, M. Pruett, and P. Schröder. Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *SIGCHI conference on Human factors in computing systems*, pp. 261–268, 2001.

[57] M. Schwarz and P. Müller. Advanced procedural modeling of architecture. *ACM Trans. Graph*, 34(4), 2015.

[58] M. Schwärzler, L.-M. Kellner, S. Maierhofer, and M. Wimmer. Sketch-based guided modeling of 3d buildings from oriented photos. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 1–8, 2017.

[59] M. I. Shamos. *Computational geometry.* Yale University, 1978.

[60] ShapeLab. Shapelab. `https://store.steampowered.com/app/571890/ShapeLab/`, 2018.

[61] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[62] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. *ACM Trans. Graph*, 27(5):1–10, 2008.

[63] N. Smith, N. Moehrle, M. Goesele, and W. Heidrich. Aerial path planning for urban scene reconstruction: A continuous optimization method and benchmark. *ACM Trans. Graph.*, 37(6), 2018.

[64] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[65] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):1–14, 2011.

[66] TiltBrush. Google tiltbrush. `https://tiltbrush.com/`, 2018.

[67] P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. Instant architecture. *ACM Trans. Graph*, 22(3):669–677, 2003.

[68] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM Trans. Graph.*, 28(5), 2009.

[69] C. Yang, D. Sharon, and M. van de Panne. Sketch-based modeling of parameterized objects. In *ACM SIGGRAPH Sketches*, pp. 89–es, 2005.

[70] Q. Yuan and Y. Huai. Immersive sketch-based tree modeling in virtual reality. *Computers & Graphics*, 94:132–143, 2021.

[71] Y.-T. Yue, X. Zhang, Y. Yang, G. Ren, Y.-K. Choi, and W. Wang. Wiredraw: 3d wire sculpturing guided with mixed reality. In *CHI Conference on Human Factors in Computing Systems*, pp. 3693–3704, 2017.

[72] F. Zhang, Z. Liu, Z. Cheng, O. Deussen, B. Chen, and Y. Wang. Mid-air finger sketching for tree modeling. In *IEEE VR*, pp. 826–834, 2021.

[73] L. Zhu, S. Shen, X. Gao, and Z. Hu. Large scale urban scene modeling from mvs meshes. In *ECCV*, pp. 614–629, 2018.