Big Earth Data

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tbed20

An Interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network

Hongchao Fan, Gefei Kong & Chaoquan Zhang

To cite this article: Hongchao Fan, Gefei Kong & Chaoquan Zhang (2021) An Interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network, Big Earth Data, 5:1, 49-65, DOI: 10.1080/20964471.2021.1886391

To link to this article: https://doi.org/10.1080/20964471.2021.1886391

Published online: 12 Apr 2021.

BIG EARTH DATA
2021, VOL. 5, NO. 1, 49–65
https://doi.org/10.1080/20964471.2021.1886391

An Interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network

Hongchao Fan[a], Gefei Kong[b*] and Chaoquan Zhang[a*]

[a]Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway; [b]School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China

ABSTRACT

The applications of 3D building models are limited as producing them requires massive labor and time costs as well as expensive devices. In this paper, we aim to propose a novel and web-based interactive platform, *VGI3D*, to overcome these challenges. The platform is designed to reconstruct 3D building models by using free images from internet users or volunteered geographic information (VGI) platform, even though not all these images are of high quality. Our interactive platform can effectively obtain each 3D building model from images in 30 seconds, with the help of user interaction module and convolutional neural network (CNN). The user interaction module provides the boundary of building facades for 3D building modeling. And this CNN can detect facade elements even though multiple architectural styles and complex scenes are within the images. Moreover, user interaction module is designed as simple as possible to make it easier to use for both of expert and non-expert users. Meanwhile, we conducted a usability testing and collected feedback from participants to better optimize platform and user experience. In general, the usage of VGI data reduces labor and device costs, and CNN simplifies the process of elements extraction in 3D building modeling. Hence, our proposed platform offers a promising solution to the 3D modeling community.

ARTICLE HISTORY
Received 29 October 2020
Accepted 28 January 2021

KEYWORDS
3D building modeling; VGI; convolutional neural network; user interaction; low cost

1. Introduction

3D building models play a significant role in designing urban 3D and virtual cities. They support a variety of applications in urban planning, environment analysis, virtual tourism, and augmented reality (Haala & Kada, 2010; Verma, Kumar, & Hsu, 2006). 3D building models are usually divided into five level of details (LoDs) (Biljecki et al., 2016) according to the CityGML 2.0 standard (Gröger, Kolbe, Nagel, & Häfele, 2012). LoD0 model means the footprint of a building; LoD1 model is represented as a cuboid by extruding the LoD0 model. On the basis of LoD1, LoD2 model contains the roof shape of a building, and the semantic information of building elements. Compared with LoD2, LoD3 model is more complex and can be called as "architecturally detailed model", which contains facade elements of a building, such as windows and doors. LoD4 model is more complete and

CONTACT Hongchao Fan hongchao.fan@ntnu.no

*These authors contributed equally to this work.

has both internal and external building elements. 3D building modeling in LoD1 and LoD2 has had many mature approaches (Kim & Han, 2018; Li et al., 2019; Wu et al., 2017), and is a key focus for some platforms (Preka & Doulamis, 2016), but these fields actually prefer 3D building models in LoD3 or higher.

Classic methods that can create highly detailed 3D building models usually adopt grammar and topology rules (Becker, 2009; Dehbi & Plümer, 2011). While their productions are stable and complete, only limited architectural styles can be covered. In recent years, with the development of algorithm, software and hardware in computer science, more studies focused on automatic 3D building modeling and achieved a great success (Agarwal et al., 2011), which can reconstruct 3D models in most scenarios. For instance, automatic methods based on point cloud data from laser or LiDAR (Yu, Helmholz, & Belton, 2017) not only guarantee very high modeling precision, but also eliminate the defects of limited architectural styles. However, these automatic methods need the assistance of professional and expensive devices to ensure the size and quality of data to obtain impressive results. All of these requirements lead to higher costs on labor and time. Moreover, they are sensitive to noise and thus easily lead to unstable and incomplete results. In other words, that means automatic methods are difficult to reconstruct 3D models effectively when lacking data or facing data noise, without the help from users.

To address these issues, some researchers tried to reconstruct 3D building models in an interactive way. Wolberg and Zokai (2018) proposed a photocentric 3D modeling platform, and added user interaction capabilities to overcome the instability of automatic methods. Users firstly sketched the building's walls. Then structure from motion (SfM) algorithm (Ma, Soatto, Kosecka, & Sastry, 2012) was used to generate point clouds in order to more accurately reconstruct building models. However, SfM needs many images that belong to the same building (more images, the better the result) and need to know image's GPS location and camera internal parameters for camera calibration. In their experiment, it took them around 20 minutes to reconstruct two building models. That indicated its computational inefficiency and strong reliance on the number of images. Nishida, Garcia-Dorado, Aliaga, Benes, and Bousseau (2016) achieved highly detailed 3D building modeling with only single image by combining sketch-based and procedural methods. However, users were asked to draw windows and doors and select their corresponding styles. It is not friendly to non-expert users, as they are not familiar with it. In addition, they defined some patterns based on the buildings they previously used, and hence it might not be helpful to buildings with different styles. Although other interactive methods are able to reconstruct 3D building models from one image as well (Chen, Zhu, Shamir, Hu, & Cohen-Or, 2013; Zheng et al., 2012), they are dependent on accurate edge detection. They could be seriously affected by image's luminance.

In short, many existing methods rely on the size of datasets and are time consuming. Methods with higher efficiency and less image data often require user sketching, which is not convenient and friendly for non-expert users. Methods based on multiple images are often influenced by data quality, such as illumination change, camera position change, and perspective distortion. The collection and processing of datasets consume a significant amount of time and labor as well. Methods based on point clouds are sensitive to noise points. Thus additional data processing is typically necessary to solve this issue. Finally, existing automated methods for building modeling are time consuming

because they usually have to estimate camera's position and generate point clouds from a sequence of images.

Considering all the above shortcomings, a low-cost 3D building modeling method is necessary, and it can accept lower-quality images from various devices without additional information, such as GPS or camera parameters, which will reduce the cost of data collection. In addition, this method can reconstruct 3D building models with less image data. Hence, we designed an interactive platform (called *VGI3D*) based on volunteered geographic information (VGI) images for 3D building modeling. In our platform, we take VGI image data as input and all the facades prepared for modeling should be fully captured on an image. Users only need to simply outline the facade of a building, and then our platform will automatically detect and adjust the bounding boxes of facade elements. Finally, 2D locations of all the elements will be transformed into 3D coordinate system, and 3D model of this building will be shown to users in real time.

VGI employs tools to create, assemble, and disseminate geographic data provided voluntarily by individuals (Sangiambut & Sieber, 2016). With the development of WebGL and hardware, VGI data have played a more active role in geoinformation collection, especially in the collection of urban images. Although the quality of VGI images is difficult to guarantee and measure, the low-cost and high-richness of VGI image data can narrow existing research gaps to some extent. Furthermore, non-expert users modeling 3D buildings will be conducive to the development and spread of urban 3D modeling.

Compared to existing methods, the key contributions of our work are as follows:

(1) Low-cost data requirements: the input to our platform is VGI image data. These images are captured by non-expert users or volunteers who upload images to the VGI image platform. Compared to traditional methods of data collection, VGI image data have lower costs, and can cover larger areas. The use of VGI image data will significantly reduce the time and labor costs during the data collection process. Moreover, considering that the geolocations of VGI images are usually scattered and that these images are at a street level, only one or two views of a building can be obtained in many cases. Hence, our platform is designed to model simple 3D buildings using less than two images. This implies that the time required and cost incurred for 3D modeling are lower.
(2) Fast 3D building modeling: the algorithm for facade elements extraction embedded in our platform can automatically extract windows, doors and balconies within one second. Additionally, the entire workflow is completed in 30 seconds, which is faster than most existing automatic building modeling methods.
(3) Relatively High-quality modeling results: in a low-cost environment, our platform can obtain modeling results that are of relatively high quality. The modeling results will be complete and stable even on low-quality images. Meanwhile, semantic information, such as facades, windows, balconies, and roofs, will be obtained and shown to users.

The remainder of this paper is organized as follows. Section 2 describes the modeling workflow and detailed algorithms of our platform. Experimental results and a platform usability testing are going to present in Section 3. Section 4 provides the conclusions as well as future work.

2. Methodology

2.1. Workflow

Considering the convenience and availability of the proposed method, we developed a web-based platform embedded with all the algorithms (as shown in Figure 1). First, our platform accepts building images uploaded by users. Then, users provide three pieces of information to our web platform in the way of interaction, including facades boundaries, roof type, and facades views of the building in images. Next, images from users are taken into the module of facade elements extraction to automatically obtain the locations of the windows, doors and balconies. In this module, the object detection network in facade elements extraction module automatically detects facade elements and gives them semantic labels. Finally, these locations in 2D are transformed into the 3D coordinate system to construct the 3D building model and show it on the Web in real time. The generated 3D model can also be downloaded for further research.

2.2. User interaction

Since most input VGI images are at a street level and only one or two facades of a building can be available, multi-view facades of a single building are difficult to obtain by non-expert volunteers. Actually, 3D building models for urban planning or virtual cities do not

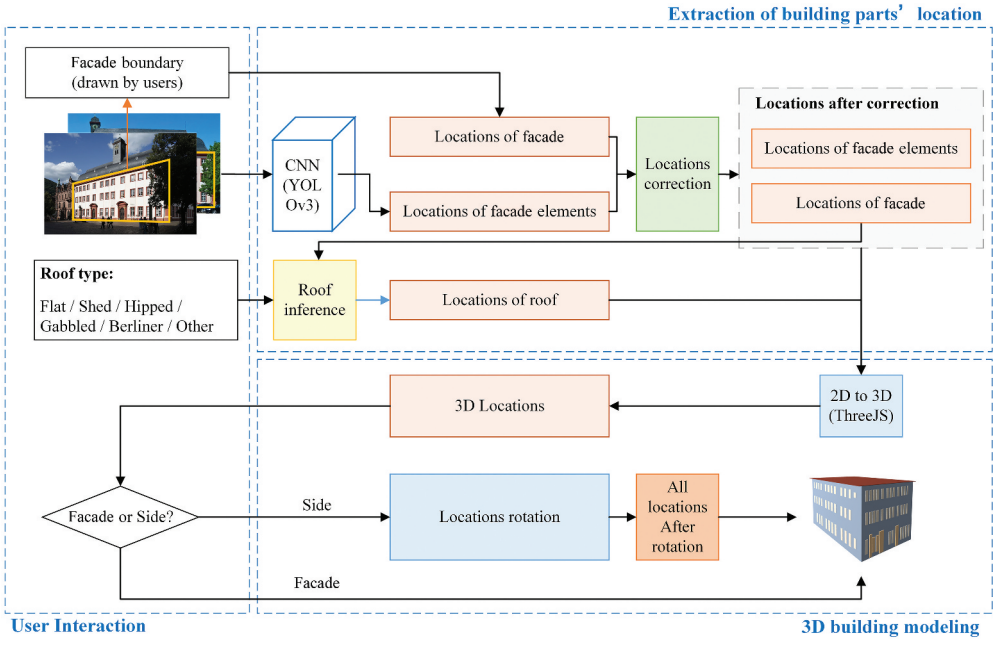


Figure 1. Framework of our platform. In the user interaction module, users provide the facades boundaries, roof type, and facade views of the building in images to our web platform. Then, images are fed into facade elements extraction module to automatically obtain the locations of the windows, doors and balconies. In the 3D building modeling step, the locations are transformed from 2D to 3D space. Then locations are rotated according to the information of facades views. Finally, the 3D building model is built and visualized to users.

require over-detailed models. Hence, in the user interaction module, users only need to upload no more than two images of a building. Facade and side images for the same building can be uploaded together to obtain a more comprehensive model. The number of images was recorded as N_I. In terms of the image requirements, there are no specific requirements regarding the cameras used or the overlap percentages. The only requirement is that users should ensure each image containing a complete facade of the building. Then, users draw the corners of the facade (X_F, Y_F) in an image to create a corresponding facade boundary, where $X_F = \{x_{F1}, x_{F2}, \ldots, x_{Fn}\}$ and $Y_F = \{y_{F1}, y_{F2}, \ldots, y_{Fn}\}$, $(X_F, Y_F) \in \{(X_F, Y_F)\} = \{(X_F, Y_F)_{ni} \mid ni = 1, \ldots, N_I\}$. The corners are used to build 3D facade and correct the locations of the facade elements. Meanwhile, the roof type of the building should be selected to help reconstruct the roof. These images and facade boundaries will be sent to our facade extraction method for the preparation of 3D building modeling.

2.3. Extracting locations of facade elements

2.3.1. Detecting facade elements

In user interaction module, we have obtained images and locations of the building facades. Images from our user interaction module are usually at a street level with various complex scenes, such as multi-view scenes, complex illumination and background. Traditional methods cannot handle these scenes well. Existing methods of interactive modeling require users to draw facade elements, such as windows and doors, which is time-consuming and labor-intensive. In addition, the accuracy and completeness of facade elements cannot be ensured in user drawings. Hence, we want to use a convolutional neural network (CNN) for object detection or semantic segmentation, to extract facade elements. However, many windows and balconies are too small in street-level images, and might be difficult to be completely segmented by semantic segmentation network. CNNs for semantic segmentation are not suitable for extracting them. Hence, we chose an object detection CNN, YOLO v3 (Redmon & Farhadi, 2018), for detecting facade elements (Kong & Fan, 2020). YOLO v3 is a one-stage, fast, and highly accurate object detection neural network. It can maintain a significant balance between detection accuracy and speed. As Kong and Fan (2020) have done, the Darknet53 was used as the backbone of YOLO v3. And the detection network was pretrained on our FacadeWHU dataset to detect windows, doors and balconies. Our dataset contains 900 street-level images (850 from Paris, France, and 50 from Trondheim, Norway) and corresponding annotations for semantic segmentation and object detection. These annotations contain six classes – window, door, balcony, roof, shop, and wall, which can meet the requirement of our facade elements detection. Uploaded images were detected directly by our trained model from Kong and Fan's work (2020). Then, every location (which is also called "bounding box") of windows, balconies, and doors was obtained and organized as $(C_{cnn1}, C_{cnn2}, classcode) = ((x_{cnn1}, y_{cnn1}), (x_{cnn2}, y_{cnn2}), classcode)$ for correction. $(C_{cnn1}, C_{cnn2}, classcode) \in \{(C_{cnn1}, C_{cnn2}, classcode)\} = \{(C_{cnn1}, C_{cnn2}, \text{classcode})_{ni}^{k,i} \mid k = 1, \ldots, N_c; i = 1, \ldots, n_k; ni = 1, \ldots, N_I\}$, where N_c is the number of classes, n_k is the number of facade elements in every class of a facade, and N_I is the number of input images.

2.3.2. Correction and inference of locations of facade elements

The locations of the building facade and its elements are based on images. However, original VGI images commonly have perspective distortion, which distorts the shape and

layout of facade elements. Therefore, the locations of the facade and corresponding elements (windows, doors and balconies) cannot be directly applied to 3D building modeling. To solve this problem, we added a submodule for location correction. We considered the following measures to correct perspective distortion for the facade in every image:

(1) First, we calculated the aspect ratio $r_F = h_f/w_F$ of the bounding box of every corresponding facade. h_F and w_F are the height and width of the facade bounding box from the user interaction module, respectively. They were calculated using Equation (1). At the same time, the facade height after perspective correction is defined as the image height h_p, and the facade width after perspective correction is defined as $w_p = h_p/r_F$. Then, we obtained the bounding box of the facade after perspective correction, as ((*0, 0*), (w_p, h_p)), and the location of facade is denoted as (X_{pF}, Y_{pF}) = ((*0, 0,* w_p, w_p), (*0,* h_p, h_p, *0*)).

$$h_F = \max(Y_F) - \min(Y_F), w_F = \max(X_F) - \min(X_F) \tag{1}$$

(2) Second, we calculated a homography matrix, *M*, by the bounding box of facade before and after perspective correction as Equations (2)–(4). The homography matrix is regarded as a perspective transformation matrix and was used for perspective correction.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = M \begin{bmatrix} x_F \\ y_F \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x_F \\ y_F \\ 1 \end{bmatrix} \tag{2}$$

$$x_p = \frac{x'}{w'} = \frac{m_{11}x_F + m_{12}y_F + m_{13}}{m_{31}x_F + m_{32}y_F + m_{33}} \tag{3}$$

$$y_p = \frac{y'}{w'} = \frac{m_{21}x_F + m_{22}y_F + m_{23}}{m_{31}x_F + m_{32}y_F + m_{33}} \tag{4}$$

where (x_F, y_F) is a facade location from user interaction module input (X_F, Y_F). (x_p, y_p) is the facade location after perspective correction.

(3) Then, the location of the facade and its elements were corrected using the perspective transformation matrix *M*. After this step, locations from user interaction and CNN, including (X_F, Y_F) for the facade and {(C_{cnn1}, C_{cnn2}, *classcode*)$_{ni}$} for facade elements, were corrected. And locations after perspective correction were obtained and denoted as (X_{pF}, Y_{pF}) for the facade and {(C_{p1}, C_{p2}, *classcode*)$_{ni}$} for facade elements, where every facade element's location of this facade is organized as (C_{p1}, C_{p2}, *classcode*)$_{ni}$ = ((x_{p1}, y_{p1}), (x_{p2}, y_{p2}), *classcode*)$_{ni}$; (C_{p1}, C_{p2}, *classcode*)$_{ni}$ ∈ {(C_{p1}, C_{p2}, *classcode*)$_{ni}$}.

The perspective distortion of the locations has been corrected. However, the facade-element layout of the perspective-corrected locations were misaligned, which will still affect the 3D building modeling process. Hence, in this step, we proposed a layout correction method for facade elements in every image:

(a) For every facade, first, we reorganized locations of facade elements after perspective correction from *xy-xy* to *xy-wh* as {((x_{pc}, y_{pc}), (we_p, he_p), *classcode*)$_{ni}$}. (x_{pc}, y_{pc}) is the center of every facade element, and (we_p, he_p) is the width and height of every facade element, respectively. Then, the locations were separated by *classcode* and output as {((x^k_{pc},y^k_{pc}), (we^k_p,he^k_p), *classcode*^{k})$_{ni}$}, where k = 1, 2, …, N_c, and N_c is the number of classes. In

the following steps, we corrected the layout of the facade elements of a facade in every class.

(b) Second, in every class, the new *xy-wh* locations were sorted by y_{pc}. We calculated the height differences between two neighboring locations and obtained the result of every class as ($h_{diff\,p}^{k}$, $classcode^{k}$).

$$h_{diff\,p}^{k,i} = y_{pc}^{k,i+1} - y_{pc}^{k,i} \quad k = 1,2,\ldots,N_c; i = 1,2,\ldots,(n_k - 1) \tag{5}$$

where N_c is the number of classes, and n_k is the number of facade elements in every class of a facade.

(c) Third, a cluster algorithm, *k-means++* (Arthur & Vassilvitskii, 2007), was used to divide the height differences into two groups. One group G_s had small height differences, which refers to the height difference of the facade elements on a single floor. The other group G_l had large height differences, which refers to the height difference of the facade elements between neighboring floors. The number of floors, N_f, was obtained using G_l as $N_f = N_{G_l} + 1$. N_{G_l} is the number in group G_l.

(d) Fourth, the facade elements in every class were divided into N_f groups based on the value of G_l and y_{pc}^{k}. Each group referred to one floor of the building. In every floor, we calculated the average *y-center* coordinate y_{ac}^{kj} and average height he_a^{kj} of the facade elements, j = 1, 2, …, N_f. The *y-center* coordinates of every facade element in every floor y_{pc}^{kj} is corrected to y_{ac}^{kj}, and the height of every facade element in every floor he_p^{kj} is corrected to he_a^{kj}.

(e) Fifth, the *x-center* coordinates of the facade elements also needed correction. Hence, we repeated steps (b) to (d). In these steps, the height differences, *y-center* coordinates and average height were replaced with width differences, *x-center* coordinates and average width. Finally, the layout of the facade elements was corrected.

The locations of the facade elements after layout correction are denoted as {(x_{ac}, y_{ac}), (we_a, he_a), $classcode$)$_{ni}$}. This *xy-wh* format cannot be used for 3D building modeling. Hence, we changed the *xy-wh* locations to *xy-xy* locations and obtained the location of every facade element as (C_{l1}, C_{l2}, $classcode$)$_{ni}$ = ((x_{l1}, y_{l1}), (x_{l2}, y_{l2}), $classcode$)$_{ni}$.

Steps (1) to (3) and (a) to (e) in section 2.3.2 were repeated to obtain the locations of all facades {(X_{pF}, Y_{pF})}and locations of their facade elements {(C_{l1}, C_{l2}, $classcode$)}.

Then the location of roof was calculated based on the location of facade from the image in the front direction. In general, basic roof shapes are divided into five classes: flat, shed, gabbled, hipped, and berliner (Kada & McKinley, 2009). Hence, we set the basic roof types as these five types. After the users choose the corresponding roof type of the building from the five basic roof types in the user interaction module, we can reconstruct the roof based on the corrected facade corners. The group with corners on the top edge of the facade is selected and is regarded as the bottom edge of the roof. Then, the top edge of the roof is automatically calculated according to the roof type. The top and bottom edges of the roof were concatenated as the final roof location (X_R, Y_R).

We grouped the locations of facades, a roof, and facade elements into N_I groups according to the number of images, and then sent them to the 3D building modeling module for 3D modeling.

2.4. 3D building modeling

After the location detection module, we obtained the locations of facades as $\{(X_{pF}, Y_{pF})\}$, the locations of windows, balconies, doors as $\{(C_{l1}, C_{l2}, classcode)\}$, and the location of roof as (X_R, Y_R). The range of these locations except roof is from 0 to h_p. h_p is based on every image height, which varies depending on the image. Thus, the size of the building models cannot remain stable. Hence, it is necessary to normalize these locations to the same range based on the height of every facade.

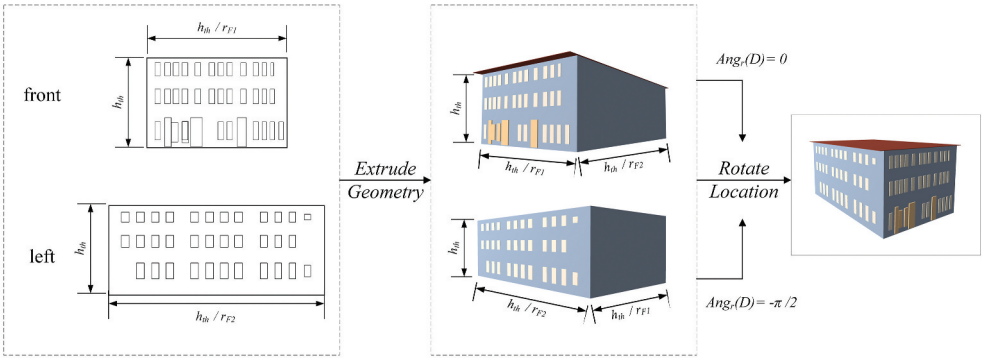
For locations from every image *ni*, we normalized these locations from range (0, h_p) to range (0, h_{th}) in height, and from range (0, w_p) to (0, h_{th}/r_F) in width as shown in Equation (6). h_{th} is a constant and we defined it as 10 to ensure the same height of all images. Moreover, to obtain a better visualization result, the center of all locations will be moved to (0, 0) by location translation.

$$x_{nr}^{k,i} = x_l^{k,i}/w_p \times h_{th}/r_F - \frac{1}{2}(h_{th}/r_F), y_{nr}^{k,i} = y_l^{k,i}/h_p \times h_{th} - \frac{1}{2}h_{th}$$
$$k = 1,2,\ldots,N_c, \qquad i = 1,2,\ldots,n_k \tag{6}$$

The location of the facade is recorded as (X_{nrF}, Y_{nrF}), and the location of every facade element after normalization is recorded as $(C_{nr1}, C_{nr2}, classcode)_{ni} = ((x_{nr1}, y_{nr1}), (x_{nr2}, y_{nr2}), classcode)$, $(C_{nr1}, C_{nr2}, classcode)_{ni} \in \{(C_{nr1}, C_{nr2}, classcode)_{ni}\}$.

The normalized locations of facade and other elements are still in the 2D coordinate system. For 3D building models, these locations need to be transformed into a 3D coordinate system. Hence, we utilized an easy and lightweight JavaScript 3D library, Three.js (Dirksen, 2015), to address this problem. Three.js provides a modeling function, *ExtrudeGeometry*, to build 3D shapes from 2D polygons. This function is applied to our platform to transform the coordinate system. The 3D locations after transformation are recorded as $((X_{3D}, Y_{3D}, Z_{3D}), classcode)_{ni}$. $X_{3D} = \{X_{3D}^0, X_{3D}^1, \ldots, X_{3D}^{N_{bp}}\}$, and Y_{3D}, Z_{3D} are the same as X_{3D}; $X_{3D}^i = (x_{3D}^{i0}, x_{3D}^{i1}, \ldots, x_{3D}^{i(co-1)})$. N_{bp} is the number of all the elements, which can be obtained by $N_{bp} = 1 + \sum_{k=1}^{N_c} n_k$. "1" in this equation means a facade number. *co* is the number of corners for the facade and every facade element. For the image in the front direction, the location of roof needs to be considered, so $N_{bp} = 2 + \sum_{k=1}^{N_c} n_k$, where "2" in this equation means the number of a roof and a facade, and *co* is the number of corners for facade, roof and every facade element when transforming locations from the image in the front direction.

We repeat the steps from section 2.4 for every input image and obtain the N_I 3D location groups of one building as $\{((X_{3D}, Y_{3D}, Z_{3D}), classcode)\} = \{((X_{3D}, Y_{3D}, Z_{3D}), classcode)_{ni} \mid ni = 1, \ldots, N_I\}$. If $N_I > 1$, these 3D location groups represent different facades of one building, which have different directions in reality. However, because the final 3D location groups are obtained from 2D images without *z-axis* information before 2D-to-3D transformation, they will have the same direction after transforming the coordinate system. Hence, these location groups need to be rotated in order to maintain their relative direction for 3D modeling. The relative directions, *D*, of these images have been provided by users in user interaction step. And the angle of rotation can be obtained using Equation (7).



(a) directions and 2D locations of every facade (X_{nrF}, Y_{nrF}) & $(C_{nr1}, C_{nr2}, classcode)_{ni}$ (b) 3D locations of every facade ni $((X_{3D}, Y_{3D}, Z_{3D}), classcode)_{ni}$ (c) 3D building model

Figure 2. The process of 3D transformation. First, the 2D locations of every facade and its facade elements are extruded using *ExtrudeGeometry* method from Three.js, to obtain the corresponding 3D locations group of the facade in (b); Then, Every 3D locations group is rotated using Equations (7)~(8) based on the facade direction from user interaction step; Finally, the 3D building model is obtained.

$$Ang_r(D) = \begin{cases} 0 & D = front \\ -\pi/2 & D = left \\ +\pi/2 & D = right \\ \pi & D = back \end{cases} \tag{7}$$

In Three.js, it uses the right-handed coordinate system, and the *y-axis* is face-up. Hence, the rotation matrix M_r can be recorded as follows.

$$M_r = \begin{bmatrix} \cos Ang_r(D) & 0 & \sin Ang_r(D) \\ 0 & 1 & 0 \\ -\sin Ang_r(D) & 0 & \cos Ang_r(D) \end{bmatrix} \tag{8}$$

Then, the 3D location groups are rotated using Equation (9) and denoted as $\{((X_{3Dr}, Y_{3Dr}, Z_{3Dr}), classcode)\}$.

$$\begin{bmatrix} X_{3Dr}{}^g & Y_{3Dr}{}^g & Z_{3Dr}{}^g \end{bmatrix}^T = M_r \begin{bmatrix} X_{3D}{}^g & Y_{3D}{}^g & Z_{3D}{}^g \end{bmatrix}^T \\ g = 1, \ldots, N_l \tag{9}$$

The process of 3D transformation is visualized in Figure 2.

The final 3D model of this building was built based on 3D locations after rotation. The semantic information of models was based on the class code of every element and was visualized using different colors.

3. Experiments

3.1. Experiment environment

We developed a web-based interactive platform. For the front-end, we used Hyper Text Markup Language (HTML), JavaScript (JS), Cascading Style Sheets (CSS), and bootstrap templates. The 3D models were rendered in the front-end using the 3D JavaScript library – Three.js. We used Python and a lightweight web application framework, Flask, to build our

algorithms and back-end. The convolutional neural network YOLOv3 was implemented with the open-source machine learning framework Pytorch (Paszke, Gross, Chintala, & Chanan, 2017). What's more, our platform was deployed on a server equipped with an Intel Core i7-8700 K CPU, 16 GB memory, and an NVIDIA GTX 1080Ti GPU.

3.2. Experiment results

In this subsection, we discuss the results of two key points in our interactive platform: the locations of all the elements and the 3D building modeling. The extraction results of facade elements and the results of 3D building modeling are shown in Section 3.2.1 and 3.2.2, respectively.

3.2.1. Extraction results of facade elements

In this paper, we used a pretrained model of YOLO v3 to extract windows, doors and balconies, which was trained on the Facade WHU dataset in Paris from Kong and Fan (2020). There are three subnetworks in Kong and Fan's work, and we only chose the window/door/balcony detection network. This model used Stochastic gradient descent (SGD), with momentum = 0.97 and weight decay = 0.00045 as the optimizer. The weighted multipart loss is also used as the loss function. This model was trained on a computer equipped with two NVIDIA 1080Ti GPUs. In the Facade WHU dataset, we only used the annotations of window, door and balcony to train the facade elements detection network, and it achieved state-of-the-art results: mAP = 71.6% and F1 score = 67.3%, where mAP is the mean average precision of objects from the three classes. We followed the standard definition of F1 score, where F1 score = $2 \times \frac{Precision \times Recall}{Precision + Recall}$. In the F1 score, precision = $\frac{TP}{TP+FP}$, and recall = $\frac{TP}{TP+FN}$. Here, TP, FP, TN, and FN mean true positive, false positive, true negative, and false negative, respectively. The visualization results are shown in Figure 3.

3.2.2. Results of 3D building modeling

The pretrained model for facade elements extraction used 850 images in Paris, France from the Facade WHU dataset as training and test data. Hence, to ensure that our workflow can work in different architectural styles, we chose five buildings from Heidelberg, Germany, and five buildings from Paris, France as test data. These images were from open-source VGI platforms Mapillary (Mapillary, 2020) and Flickr (Flickr, 2020). We used three metrics to evaluate the results of 3D building modeling. These three metrics are component precision (P_c), component recall (R_c), and component integrality (In_c), where P_c and R_c follow their standard definitions in computer vision, and In_c is defined by ourselves. TP_c, FP_c, TN_c, and FN_c are defined as true positive, false positive, true negative, and false negative of facade elements in every class.

$$P_c^k = TP_c^k / \left(TP_c^k + FP_c^k\right) k = 1,2,\ldots,N_c \tag{10}$$

$$R_c^k = TP_c^k / \left(TP_c^k + FN_c^k\right) k = 1,2,\ldots,N_c \tag{11}$$

where / is the division operator. The left side of the operator represents the number of corresponding objects, and the right side represents the total number of corresponding

Figure 3. Visualization results for extracting facade elements locations (Containing different views, lighting, and architectural styles).

objects. N_c is the number of classes. The number of facade elements in a building is usually less than 100, which means that a few false or missing detections will seriously affect the percentage of these three metrics. Hence, compared with the percent symbol, %, the division operator,/, can better represent the performance of 3D building modeling.

Unlike P_c and R_c for per-class evaluation, In_c is designed for comprehensive evaluation. In 3D building modeling, we are more concerned with the total number of correct objects that were actually retrieved than the correct objects among the whole retrieved objects. Hence, we define In_c as the ratio of the total number of the correct facade elements that were actually retrieved to the actual number of facade elements, which is similar to the definition of Recall. In the actual number of facade elements, the classes were not considered. In_c is presented as follows.

$$In_c = \sum_{k=1}^{N_c} TP_c^k / N_a \qquad k = 1,2,\ldots,N_c \tag{12}$$

where N_c is the number of classes and N_a is the actual number of facade elements.

The facade elements that are obscured by trees, people, and other foreground were not calculated in P_c, R_c, and In_c because the ground truth of these facade elements cannot be obtained. The statistical modeling results of the 10 buildings are shown in Table 1. The buildings from Heidelberg, Germany are numbered 1 to 5, and the buildings from Paris, France are numbered 6 to 10. From the table, we can see that the overall modeling integrality of our workflow In_c is higher than 75% in various complex scenes. When facing two complex tasks of street-level 3D building modeling, (1) buildings with complex illumination and serious perspective distortion such as No. 2, 4, 7, 8, and 10, and (2) buildings with many facade

Table 1. Statistic modeling results of the 10 buildings.

Building No.	1		2		3		4		5	
	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c
window	19/21	19/21	18/19	18/19	18/19	18/18	10/12	10/10	66/66	66/68
balcony	3/3	3/4	0/0	0/1	-	-	-	-	-	-
door	-	-	1/1	1/1	1/1	1/1	1/1	1/2	2/5	2/2
In_c	22/25		19/21		19/19		11/12		68/70	
Building No.	6		7		8		9		10	
	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c
window	4/6	4/4	10/12	10/10	2/2	2/3	22/22	22/23	13/16	13/13
balcony	3/3	3/3	10/10	10/10	-	-	19/19	19/20	2/2	2/4
door	0/0	0/1	3/3	3/4	1/1	1/1	1/1	1/1	0/0	0/1
In_c	7/8		23/24		3/4		42/44		15/18	

elements such as No. 1, 2, 3, 5, and 9, our workflow can achieve stable 3D building modeling results with the In_c higher than 85%. At the same time, our workflow achieves impressive performance in the modeling of the key class. For the key class *window*, both of component precision P_c and recall R_c, are higher than 67%. In the case of buildings with more than five windows, the P_c and R_c of window are even higher than 80%. Moreover, for the modeling results of buildings in Paris and Heidelberg, there are the similar In_cs of building models in

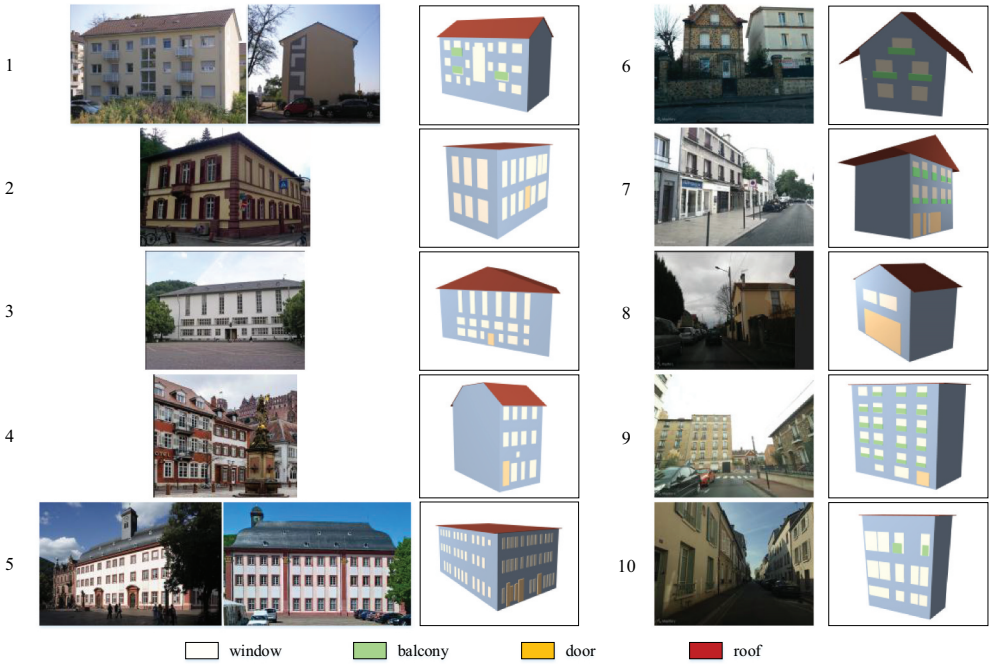


Figure 4. Qualitative result of 3D building modeling. The first and third columns are the original images of every building. The second and fourth columns are the 3D modeling results of every building through our workflow.

both cities. Hence, our workflow can be used for buildings with different architectural styles without extra work. The qualitative results of 3D building models are shown in Figure 4.

3.2.3. Performance of the platform

In addition to evaluating the performance of building extraction and modeling in our interactive platform, we tested the performance of our entire workflow. The modeling time of our workflow, *T*, was used for evaluation. Modeling time refers to the running time from when the user saves their drawing to the 3D building model shown on our website. Except for the time of facade elements extraction and 3D building modeling, the I/O time between the front-end and back-end was planned to be included to offer a more comprehensive evaluation. Given the different background of users, there usually is a big difference among them finishing user interaction, such as uploading images and drawing corresponding building facades. Hence, this time is not included in the modeling time *T*. Our platform was tested on a local area network (LAN). In our preliminary test, we found that the I/O time is less than 1 ms and can be ignored. The convolutional neural network takes most of the modeling time. Hence, in further tests, the I/O time was not taken into account, and the time taken by the convolutional neural network was tested in two different modes: GPU mode and CPU mode. In the GPU mode, we used an NVIDIA 1080Ti GPU to test the modeling time. The average modeling time in GPU mode aT_G = 2.03 s, and the maximum modeling time in GPU mode mT_G = 6.62 s. In the CPU mode, we used an Intel 8700 K CPU to test the modeling time. The average modeling time aT_C = 15.35 s, and the maximum modeling time mT_C = 27.08 s. The modeling time of our platform in the GPU mode is much faster than that in the CPU mode. Even in the CPU mode, our interactive platform can still achieve the 3D modeling of simple buildings in 30 s, which is faster than most 3D building modeling methods.

Figure 5 illustrates our *VGI3D* platform. Users can upload images and draw facades of buildings using our website. The website for our platform is available at https://18.210.26.42:5002/facade/.

3.3. Usability testing

User interaction plays an important role in the interactive 3D modeling workflow. However, this is difficult to evaluate. Hence, we invited four students to experience and test our platform. Then user interaction module will be evaluated according to their feedback. Four students included two men and two women. One among them is an expert user of 3D city modeling, and the others are users focusing on other fields, such as spatial analysis. The usability testing didn't follow any additional guidance from our development team. They were only told the purpose of the platform and the functions of every button and every workspace. Fortunately, we received positive responses from all four participants. In terms of user interaction logic, all the participants, both men and women, considered the interface to be clear and concise. As for 3D modeling, our platform can model buildings at a rapid pace. The speed and completeness of the modeling were beyond their expectations. At the same time, the three participants without background in 3D city modeling were also able to construct 3D building models only with basic guidance as we mentioned before.

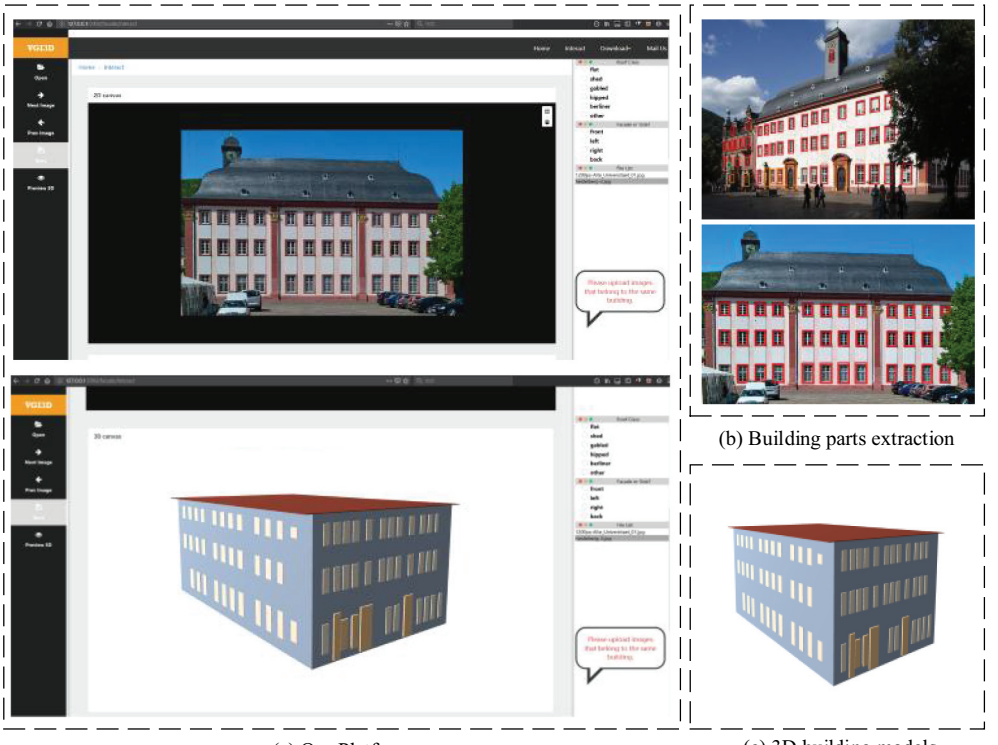


Figure 5. Results of our platform. (a) Interface of our platform. (b) Result of facade elements extraction, which is a middle part of our workflow. (c) Final result of 3D building modeling shown on our website.

4. Conclusions

In this study, we propose a new interactive platform, *VGI3D*, for 3D building modeling. This platform is mainly composed of user interaction, automatic facade elements extraction, automatic roof inference, and real-time modeling of 3D buildings. Our platform only uses one or two images and simple user sketching as input. The extraction of facade elements and building modeling are automatically achieved by an object detection network and inference in 30 s, leading the platform to realize fast and complete urban 3D building modeling with lower labor and time costs. The 3D modeling performance of our platform is evaluated on images captured by various mobile phones and basic digital cameras, in which buildings have different architectural styles. Our experimental results demonstrate the capability of our platform for lightweight 3D building modeling. We also conducted a usability testing by inviting four students to try our platform and then give feedback. Their feedback indicates that our platform is easy to use and the interface is user friendly.

For further progress on 3D building modeling, we have released our platform on https://18.210.26.42:5002/facade/. In the future, we will further improve our platform to model complex buildings with more facades, such as shopping complexes and museums. Moreover, we will try to support the geographical matching of building models in the platform to make it easier to apply to urban 3D and other fields.

Disclosure statement

There are no conflicts of interest to declare.

ORCID

Hongchao Fan http://orcid.org/0000-0002-0051-7451

Data Availability

The FacadeWHU dataset that supports this study is openly available at: https://drive.google.com/drive/folders/1BeKXMuNAaRcwP6lSeiqFOSfscbP9ZKm7?usp=sharing.

Funding

This work is supported by the National Natural Science Foundation of China (NSFC) under project [no. 41771484].

Notes on contributors

Hongchao Fan is professor for 3D Geoinformatics at the Department of Civil and Environmental Engineering at the Norwegian University of Science and Technology (NTNU). He received his master's degree in Geodesy and Geoinformatics at the University of Stuttgart and obtained his Ph.D. at the Technical University of Munich in Germany. After that he worked as Group Leader for 3D Data Infrastructure at the Heidelberg University for six years. In 2018, he started his work as professor at NTNU in Trondheim, Norway. His research interests include 3D city modeling, spatial data mining from VGI data and laser scanning.

Gefei Kong received the bachelor's degree in geographical condition monitoring from Wuhan University, Wuhan, China, in 2018, and is currently pursuing the master's degree with Wuhan University, Wuhan, China. Her research interests include geographical information engineering, computer vision, and 3D modelling, especially urban 3D modeling with their applications.

Chaoquan Zhang received the M.S. degree in software engineering from Nanjing University of Finance and Economics, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway. His research interests include digital city 3D visualization, deep learning-based point cloud object detection and segmentation, and 3D digital modeling.

References

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., & Szeliski, R. (2011). Building rome in a day. *Communications of the ACM*, *54*(10), 105–112.

Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (pp. 1027–1035). New Orleans, LA: Society for Industrial and Applied Mathematics.

Becker, S. (2009). Generation and application of rules for quality dependent facade reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, *64*(6), 640–653.

Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, *59*, 25–37.

Chen, T., Zhu, Z., Shamir, A., Hu, S.-M., & Cohen-Or, D. (2013). 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, *32*(6), 195.

Dehbi, Y., & Plümer, L. (2011). Learning grammar rules of building parts from precise models and noisy observations. *ISPRS Journal of Photogrammetry and Remote Sensing*, *66*(2), 166–176.

Dirksen, J. (2015). *Learning Three. js–the JavaScript 3D Library for WebGL*. Birmingham, UK: Packt Publishing Ltd.

Flickr. (2020). Retrieved from https://www.flickr.com/

Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K. H. (2012). OGC city geography markup language (CityGML) encoding standard.

Haala, N., & Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, *65*(6), 570–580.

Kada, M., & McKinley, L. (2009). 3D building reconstruction from LiDAR based on a cell decomposition approach. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *38*(Part 3), W4.

Kim, H., & Han, S. (2018). Interactive 3D building modeling method using panoramic image sequences and digital map. *Multimedia Tools and Applications*, *77*(20), 27387–27404.

Kong, G., & Fan, H. (2020). Enhanced facade parsing for street-level images using convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 1–13. doi:10.1109/TGRS.2020.3035878

Li, Y., Ding, Z., Miao, W., Zhang, M., Li, W., & Ye, W. (2019, October). Low-cost 3D building modeling via image processing. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 331–335). Beijing, China: IEEE.

Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. S. (2012). *An invitation to 3-d vision: From images to geometric models* (Vol. 26). New York, NY: Springer Science & Business Media.

Mapillary. (2020). Retrieved from https://www.mapillary.com/

Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B., & Bousseau, A. (2016). Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)*, *35*(4), 130.

Paszke, A., Gross, S., Chintala, S., & Chanan, G., 2017. Pytorch: Tensors and dynamic neural networks in python with strong GPU acceleration, 6.

Preka, D., & Doulamis, A. (2016). 3D building modeling in LoD2 using the CityGML standard. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, *42*, 11–16.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Sangiambut, S., & Sieber, R. (2016). The V in VGI: Citizens or civic data sources. *Urban Planning*, *1*(2), 141–154.

Verma, V., Kumar, R., & Hsu, S. (2006, June). 3d building detection and modeling from aerial lidar data. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (Vol. 2, pp. 2213–2220). New York, NY: IEEE.

Wolberg, G., & Zokai, S. (2018). PhotoSketch: A photocentric urban 3D modeling system. *The Visual Computer*, *34*(5), 605–616.

Wu, B., Yu, B., Wu, Q., Yao, S., Zhao, F., Mao, W., & Wu, J. (2017). A graph-based approach for 3D building model reconstruction from airborne LiDAR point clouds. *Remote Sensing*, *9*(1), 92.

Yu, Q., Helmholz, P., & Belton, D. (2017). Semantically enhanced 3D building model reconstruction from terrestrial laser-scanning data. *Journal of Surveying Engineering*, *143*(4), 04017015.

Zheng, Y., Chen, X., Cheng, M. M., Zhou, K., Hu, S. M., & Mitra, N. J. (2012). Interactive images: Cuboid proxies for smart image manipulation. *ACM Transactions on Graphics*, *31*(4), 99–100.