Earth Science Informatics (2023) 16:2559–2571
https://doi.org/10.1007/s12145-023-01053-0

RESEARCH

Deep learning aided web-based procedural modelling of LOD2 city models

Ziya Usta[1] · Alper Tunga Akın[2] · Çetin Cömert[2]

Received: 24 March 2023 / Accepted: 18 July 2023 / Published online: 29 July 2023

Abstract

In a large variety of smart city applications, the processes are settled with LOD2 (Level of Details) and the generation of the LOD2 models requires the proper generation of the roof geometries. In general, obtaining roof type information and succeeding generations of the LOD2 models requires expensive aerial surveys and time-consuming construction processes. In this study, a methodology to generate LOD2 building models using only 2D building footprints and aerial imagery is explained to overcome these challenges. The roof type information has been obtained from an aerial image that covers the entire study area using a CNN (Convolutional Neural Network) model. Then, the roof geometries have been constructed procedurally by extending and implementing a well-known Straight Skeleton (SS) algorithm for three main types of roofs: flat, gable and hipped. These constructed roof geometries have been combined with LOD1 block models generated by extruding the 2D footprints according to the height attribute. The overall accuracy of the CNN is 89.9% and the class-wise accuracies are over 84% for all classes. The least recall value is observed for the gable roof class, the enhancement options are discussed in the relevant section. The proposed methodology has been developed as a web-based solution utilizing RESTful web services with modern web technologies. In summary, the main novelty of the study is based on two contributions: using DL for gathering roof-type information without any end-user interference and the extension of the SS algorithm for the construction of roof geometries. The final product of this study is a web-based architecture for the rapid generation of the LOD2 building models.

Keywords 3D modelling · City models · Aerial imagery · Procedural modelling · Deep learning

Introduction

With the rapid development in data acquisition methods and computer graphics, 3D City Models (3DCM) are widely used, mainly in the city planning industry. A literature review about 3DCMs and their applications can be found in (Biljecki et al. 2015). With the spread of open data policies, which are based on institutions making their data publicly available (Welle Donker et al. 2016), many institutions in Europe and the United States share 3DCMs as open data via the web using web technologies HTML5 and WebGL. Rotterdam, Amsterdam, Delft, Berlin, and New York are notable ones.

Most of the 3DCMs are found as LOD1 block models in practice since LOD2 models are much more difficult to obtain because of the need for time-consuming processes and expensive data acquisition techniques (Biljecki et al. 2017a). The minimum required level of detail in 3DCMs varies according to use cases (Biljecki et al. 2017b). While LOD1 models are sufficient for some use cases such as shadow analysis, other use cases require general roof geometries, such as solar analysis (Biljecki and Dehbi et al. 2021; Weiler et al. 2019).

In this work, the possibility of constructing LOD2 models using 2D data such as building footprints and aerial images has been investigated. It is exceedingly difficult to model building roofs from 2D building footprints without roof type information. Deep learning techniques (DL) have been used to derive roof types from aerial imagery to overcome this issue. DL techniques are currently in use to obtain required

Communicated by: H. Babaie

✉ Ziya Usta
ziyausta@artvin.edu.tr

Alper Tunga Akın
alpertunga@ktu.edu.tr

Çetin Cömert
ccomert@ktu.edu.tr

[1] Department of Geomatics Engineering, Artvin Coruh University, Artvin 08100, Turkey

[2] Department of Geomatics Engineering, Karadeniz Technical University, Trabzon 61080, Turkey

roof type information or directly reconstruct 3D roof geometries (Buyukdemircioglu et al. 2022).

DL is a machine learning technique that a computer model learns to perform classification tasks directly from images, text, or sound. DL models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers (Bengio et al. 2017). For this purpose, a DL model called CNN (Convolutional Neural Network) has been used to classify three main roof types, which are Hipped, Gable and Flat, with an aerial image in this work. After the classification, extracted roof type information has been used to construct roof geometries.

The generation of 3DCMs as LOD2 is done using procedural modelling techniques (PM) in general (Tsiliakou et al. 2014; Martinovic 2015). PM is an umbrella term which includes many modelling techniques such as L-Systems, fractals, and shape grammar. All these techniques aim to create multiple instance models by utilizing a set of predefined rules and algorithms. The batch modelling approach of PM minimizes user interaction and labour. Many studies used the CGA (Computer Generated Architecture) shape grammar to generate 3D urban objects procedurally with roof geometries or not (Alomía et al. 2021; Zhang et al. 2022). Even if the CGA eases the batch modelling, additional information to generate LOD2 models such as roof type is provided manually by the user. This situation is not practical manner, especially in city-scale applications. Besides, the user should know the grammar and the syntax of the CGA language, which can be overwhelming to a new user, to execute model generation operations (Kelly 2021; Zhang et al. 2022). In this study, the PM technique has been used with roof type information that is derived via DL to be able to construct roof geometries without a need to learn a new modelling language or input any information manually.

Existing software components for PM are limited in both their numbers and capabilities. ESRI CityEngine (Kelly 2021) which utilizes CGA shape grammar is a commercial and stand-alone desktop application. Hence, it can not be integrated into any software component. Random3DCity (Biljecki et al. 2016) constructs 3DCMs with random properties for synthetic data generation. Besides its being unsuitable for real-world use cases, it also needs a Python interpreter. BuildingReconstruction (B-Rec) is another desktop application (URL-1) which supports 27 different roof types for constructing roofs. However, it needs architectural plans as an additional data source to be able to decide the roof type to model. Those architectural plans are generally hard-to-reach data sources and this hinders the usability of the software. The platform dependency of these applications limits the accessibility and usability for the end-users who could have different platforms of devices. This drawback can be overcome by transforming the modelling algorithms to web services, hence users can model their 2D data from their browsers without any software or plugin installation.

The aim of this study is to develop and implement a methodology in order to fastly generate 3DCMs with generalized roof geometries using only 2D easy-to-reach data in a web-based environment without any user input or interference. For this purpose, there are three sequential steps which are LOD1 block model generation, obtaining roof type information, LOD2 model generation using the roof type information, and web-based visualization of the model. LOD1 block models have been generated via footprint extrusion (Ledoux and Meijers 2011; Arroyo Ohori et al. 2015) in this study. After the obtaining roof types from aerial images, roof geometries have been constructed using the Straight Skeleton Algorithm (Aichholzer et al 1996) and Sweep Line Algorithm (Souvaine 2005). Then, roof geometries are added to the top of block models to obtain the LOD2 models. RESTful web services have been developed with web technologies to be able to implement the proposed methodology as a web-based solution. The entire computational cost of the model generation has been loaded to the server. Finally, constructed roof geometries and generated 3DCM have been visualized in the client's browser via HTML5 and WebGL.

There are two main novel contributions presented in this work. The first one is using DL for gathering roof-type information from aerial images automatically without any user interference. The second one is that a well-known Straight Skeleton algorithm has been extended using the Sweep Line algorithm to construct 3D roof geometries. Finally, a web-based architecture for the rapid generation of the LOD2 building models has been proven and implemented. In addition to these two main contributions, if the user provides the condominium unit plan[1] of the building beside the 2D footprint, our pipeline is able to model the building's condominium units by extrusion of the plan.

The rest of the paper is organized as follows. In "Related work" section, the related studies to our study are criticized. In "Methodology" section, the general architecture and the development essentials are described. In "Findings and discussion" section, experimental results and possible drawbacks are discussed. Finally, "Conclusion" section states the conclusion and future work.

Related work

One of the most common methods to generate 3DCMs is to extrude 2D building footprints (Tekavec et al. 2020; Zhang et al. 2022; Nurkarim and Wijayanto 2023). Nevertheless,

[1] as used by Land registry Offices in Turkey.

3DCMs generated by footprint extrusion generally neglect roof geometries. And this hinders the widespread use of 3DCMs. While some 3D spatial analyses can be performed using LOD1 data, such as shadow analysis, others, such as solar potential, require minimum LOD2 data (Biljecki and Dehbi 2019; Weiler et al. 2019). (Senyurdusev et al. 2020; Senyurdusev and Dogru 2020) and (Alomia et al. 2021) use OSM data and 'Roof Type' or other advanced attributes that describe roof geometries to construct roofs procedurally. However, such attribution may be missing in OSM data as experienced for Turkey during this work. (Yang and Delparte 2022) uses 2D datasets and generates 3DCM of the Old Town Pocatello in Idaho procedurally. Although that work includes 3D roof geometries, there is no information about how to model or obtain 3D roofs. In (Demir and Koramaz 2018), roof geometries are obtained with user input using CGA modelling rules. Consequently, in all of the above mentioned studies the gathering of the roof type information requires either additional data sources or user interference.

(Tomljenovic et al. 2015; Bauchet and Lafarge 2019, Dehbi et al. 2021) uses LiDAR point cloud data to generate LOD2 3DCMs. Typically, LiDAR is more suitable for conducting precise surveys on a smaller scale (Khayyal et al. 2022), this option could be more resource-consuming on large-scale applications, such as huge 3DCMs. Roof shape and building shell is the most valuable geometric information that can be extracted from point cloud data. This process requires additional data such as 2D building footprints and computationally intensive data process workflows to be able to classify and construct city objects. If the use case requires photorealistic and precise models, similar to the ones in (Huang 2019), the time and resource consumption is worth making an effort. However, for a large number of smart city applications such as solar energy analyses, LOD2 models are sufficient (Wendel et al. 2017). Thus, DL models have been used to obtain general roof type information from aerial 2D images in this study.

In recent studies, DL models are commonly in use to obtain roof type or to reconstruct 3D information (Buyukdemircioglu et al. 2022). (Castagno and Atkins 2018) used CNNs to classify roof types for multi-copter emergency landing site selection. First, they used the polygon of the building roof outline for cropping data from a LIDAR-based DSM image. Then, they used the polygon for cropping from the RGB image in the same way. After the data preparation, they fuse RGB and LIDAR images as input to CNN. This data fusion could drive the decision process in a precise way, but such fusion may be more than enough for various smart city applications like shadow analysis or solar analysis which could be driven using roughly-obtained LOD2 models. (Partovi et al. 2019) chose a DL-based approach. They used LIDAR-based DSM and pan-sharpened VHR (Very High Resolution) satellite images, but their approach has drawbacks with non-rectilinear roof shapes. (Axelsson et al. 2018) used CNNs with photogrammetric point clouds obtained from aerial images for classifying roof types. But they consider the roofs as only two classes: ridge and flat roofs. (Biljecki and Dehbi 2019) have obtained roof types from LOD1 models with machine learning without using point clouds, but they have not yet constructed roof geometries using this information. Also, they used the LOD1 city model, not 2D building footprints.

Methodology

Obtaining roof type information

A Python script was written to generate train images from aerial images. These images served as open data and can

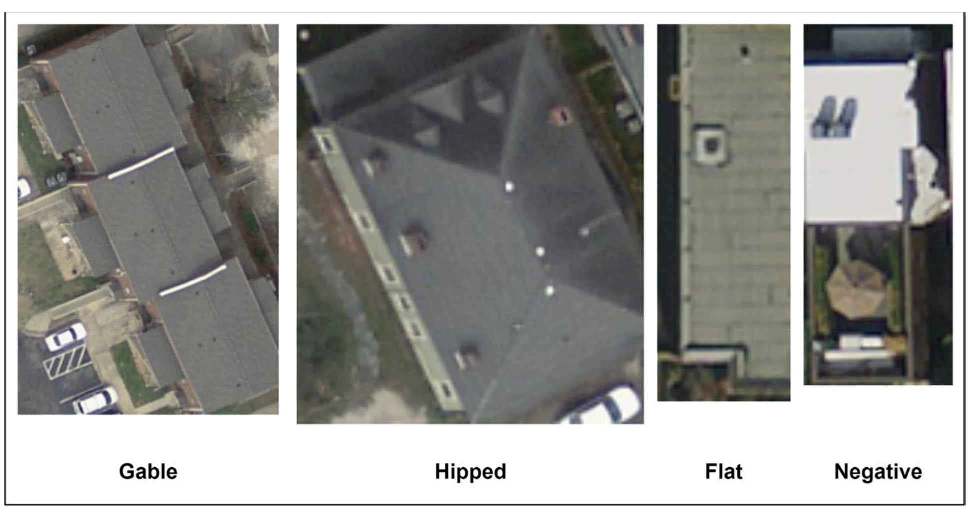

Fig. 1 Samples from the roof type dataset

be found in (Bradbury et al., 2016). This collection consists of RGB (Red–Green–Blue) orthoimages that have various resolutions between 0.15 m and 0.3 m. We have used the Atlanta_03.tif image which has 0.15 m of GSD from this collection. Geopandas library was used to extract geometry and attributes from 2D footprints which were also obtained from (Bradbury et al., 2016) as.CSV file. With this script, multiple building roofs were cropped from an aerial image and saved as separate image files using building envelopes that are extracted from 2D building footprints (PrepareData.py). Using Microsoft Custom Vision Service (Salvaris et al. 2018) that eases the training process of DL models, training images were labeled as “Hipped”, “Gable”, “Flat” and “Negative” in train data (Fig. 1). The “Negative” label is used when the image cannot be identified as hipped, gable or flat. A general workflow for the training process is shown in Fig. 2.

The batch size, optimizer and learning rate parameters, which are used for the tuning of the DL model, are not user-inferable parameters in the Microsoft Azure Custom Vision platform. The platform tunes these parameters empirically considering the increase in the performance metrics through iterations. The model is trained in the “Classification” domain in the platform and the architecture of the model is given in (URL-2). The model resizes the input RGB image in the shape of 224*224*3 and uses the ReLU function as an activation function in hidden layers and the Softmax activation function in the output, or identify layer.

After the training process, the DL model has reached 89,9% accuracy in predicting roof type. The trained model was exported from Custom Vision as a TensorFlow graph file to be used in a prediction process. A trained model can be saved as a model file and can be used in a different environment or application for the prediction process. This is often called “transfer of learning” in literature. A Python script was also coded to predict roof types. The model file was parsed using TensorFlow library (Abadi et al. 2015) and roof types were predicted via this script automatically. A shapefile which consists of 2D building footprints were enriched with roof types as an attribute. All the workflow for prediction of roof type can be found on a publicly available Github repository that includes train data, test data and scripts.

Extrusion and block model generation

A 3D model of the city can be obtained via extrusion of the building footprints. For this purpose, normal vectors of the building footprints are calculated. The normal vector is a unit vector perpendicular to the surface (Fig. 3). Extracting P1 from P2 and P2 from P3 vectors V1 and V2 are obtained. Using the cross product of these vectors, surface normal is calculated.

After the calculation of surface normals, using the “height” attribute of the building footprints, building footprints are

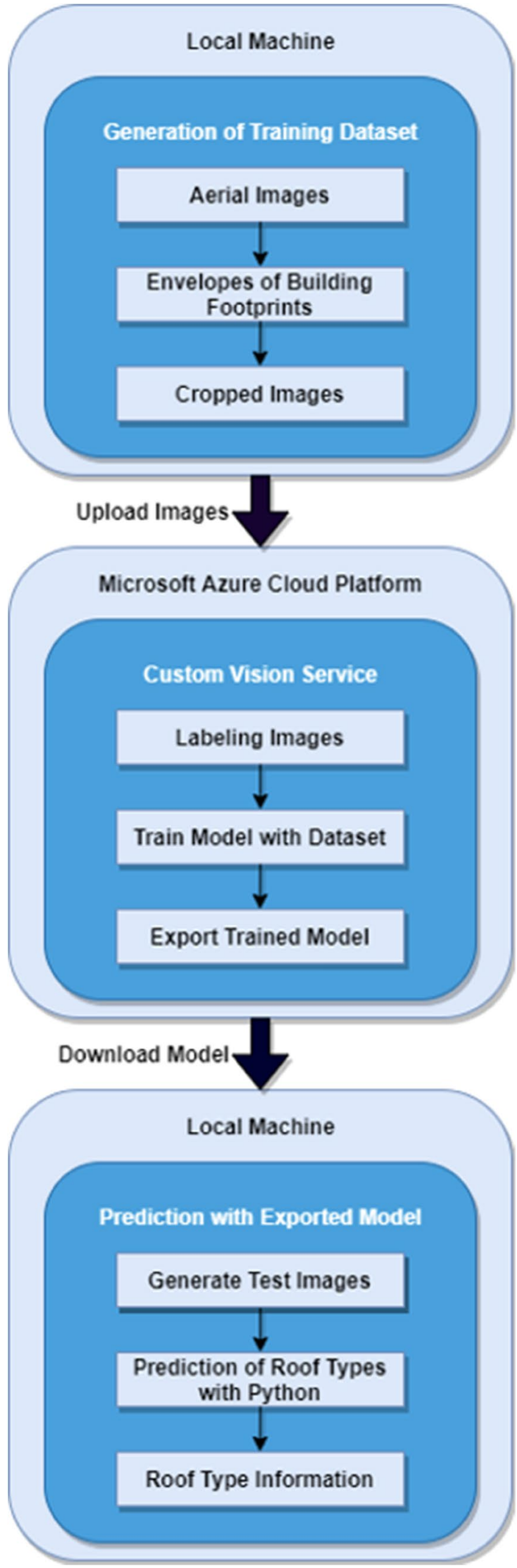

Fig. 2 Training and prediction processes

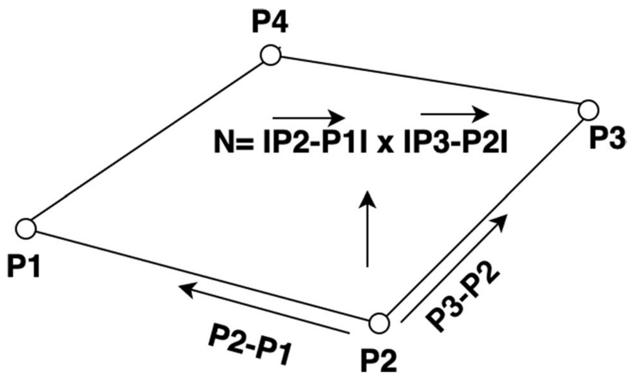

Fig. 3 Calculation of the surface normal

translated along surface normals and top surfaces are constructed. By indexing the top and footprint points, vertical surfaces are constructed and LOD1 block model is derived. (Fig. 4).

If condominium unit plans (CUPs) are provided (Fig. 5), our modeler can model condominium units as well at city scale. For this purpose, each condominium is translated along the surface normal according to the height of the condominium in CUP. Then, translated polygons are extruded and condominiums are modelled procedurally (Fig. 6).

Geometries of the condominium units are stored in a CityJSON file. Since condominium units are not defined in the CityGML data model, an ADE is developed for storing condominium units.

Construction of roof geometries

A straight skeleton algorithm is used to model roof geometries. The definition of the straight skeleton (SS) is also its construction. SS is a geometric structure that consists of only straight edges, because of the stepwise shrinking process of a polygon (Fig. 7).

In this shrinking process, each edge of the polygon moves inwards of the polygon in a self-parallel manner. Two events change the topology of the input polygon. Edge Event occurs when an edge shrinks to zero and creates a node in skeleton Fig. 8 and Split Event occurs when a reflex vertex touches to a non-consecutive edge and creates a node in skeleton Fig. 9.

There are two approaches to stimulate this shrinking process. The 2D construction method is based on angular bisectors (Felkel and Obdrzalek 1998) and the 3D construction method is based on sweep planes first introduced by (Eppstein and Erickson 1999) and used by (Kelly 2014).

In the bisector-based approach, to create roof geometries in 3D, the skeleton must be traversed and Z values of nodes that are created by the skeleton must be manipulated according to proper roof height. Since sweep plane approach does not require this additional step, sweep plane approach has been used in this work.

In sweep plane approach, all nodes of the skeleton are detected by intersection of planes. Edge events have been detected by collision of three consecutive direction planes with the sweep plane and split events have been detected by collision of two consecutive and one non-consecutive direction planes with sweep plane.

In the proposed algorithm which takes polygons as input, vertices of polygons are stored in a circular doubly linked list, every vertex therefore, has pointers to consecutive vertices and also consecutive direction planes. Sweep plane, which is a plane parallel to the input polygon, moves in the direction of the Z axis and the polygon shrinks. Edge events and split events are stored in a

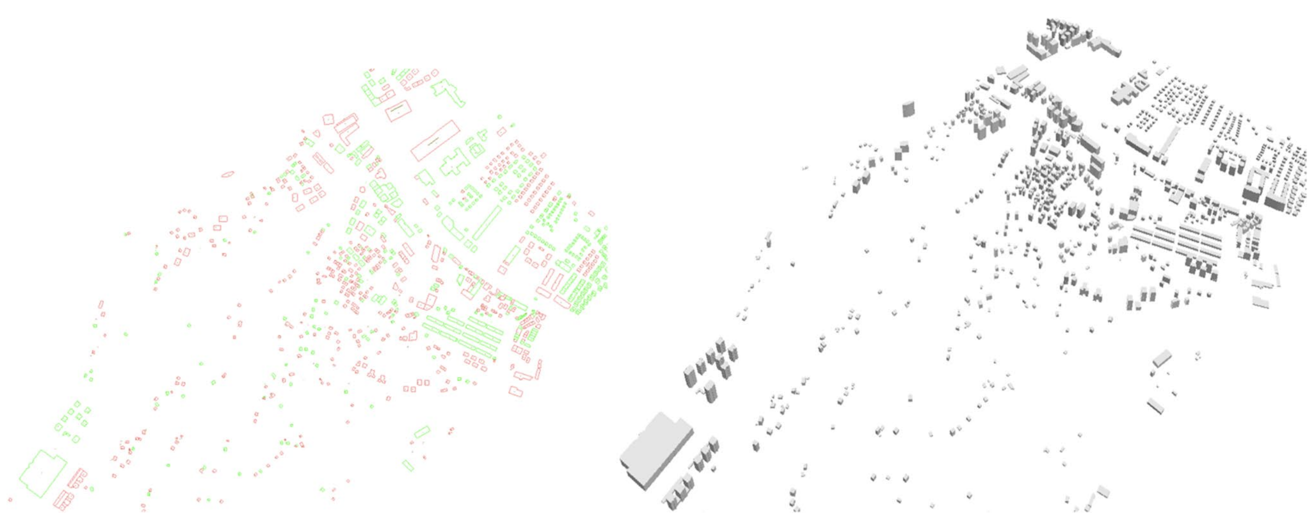

Fig. 4 2D building footprints (left) 3D block model (right)

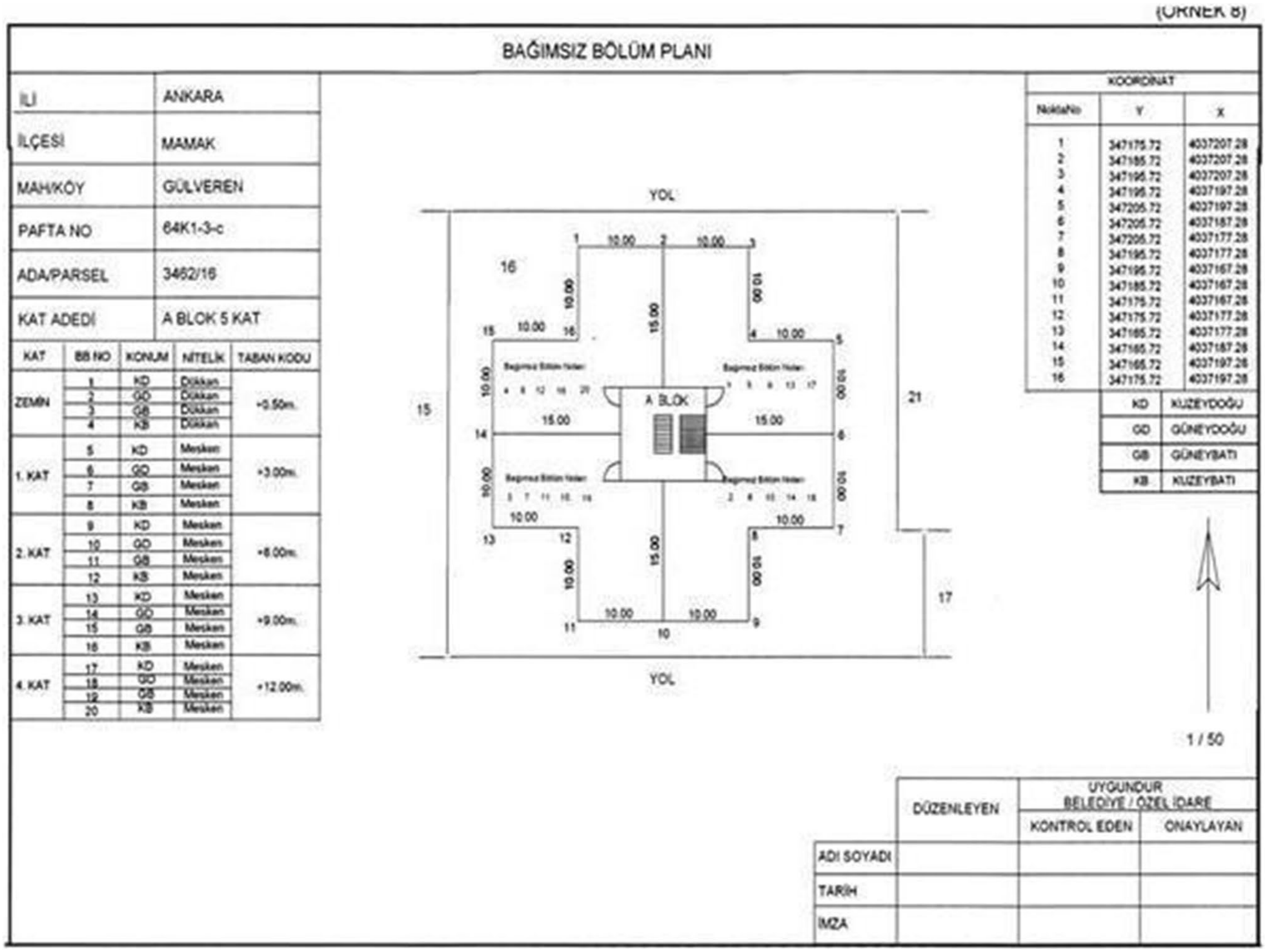

Fig. 5 Condominium unit plan (Çağdaş 2012)

priority queue according to the height of the sweep plane which gives the order of nodes in SS. When every edge of the input polygon shrinks to zero algorithm finishes and SS completed.

The result of SS is a directed graph. Roof surfaces must be generated from this graph to generate roofs. A 3D sweep line algorithm has been implemented to accomplish this. In this algorithm, first, nodes of SS are grouped according to the original edges of the input polygon. Then, for every group, a sweep line that is perpendicular to the related edge, moves from start to the end of the edge. Intersection order of nodes with this sweep line determines the order of the points for the roof polygon (Fig. 10). The horizontal roof surfaces are obtained using these orders. A predefined slope value is used to obtain the peak of the roof. Every roof surface is inclined from the edge to the skeleton node using the slope and roof geometry is obtained. In this work, the slope value is initially considered as %45

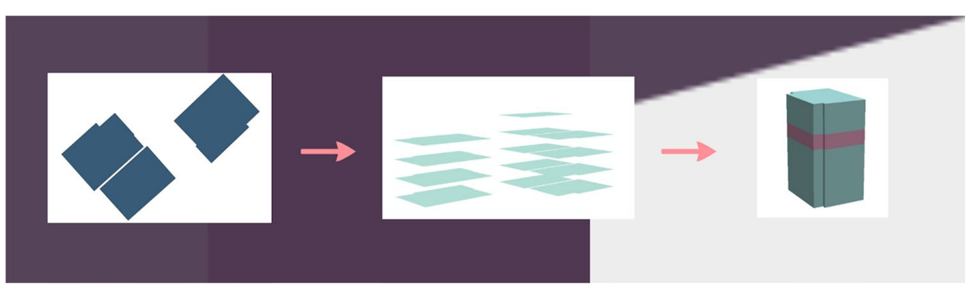

Fig. 6 Polygons (left) multiplied and translated (middle) condominium unit in 3D

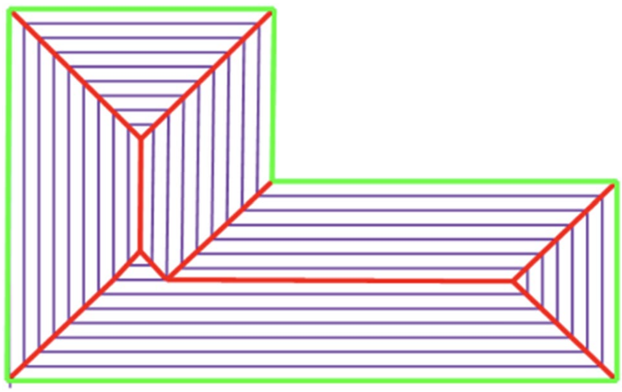

Fig. 7 Input polygon (green), wave fronts (purple lines) and SS (red lines)

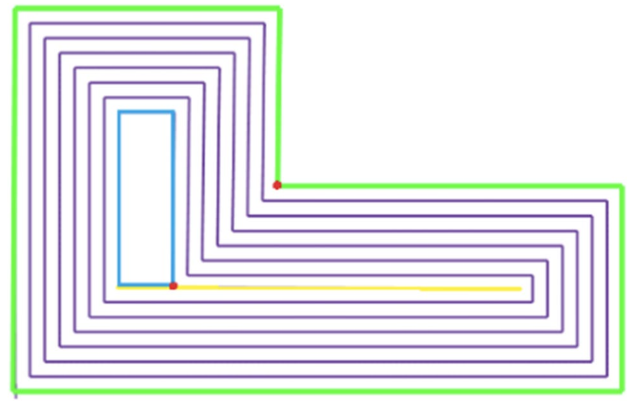

Fig. 9 Split Event (yellow edge meets red reflex vertex and this creates blue polygon)

because it is Turkey's maximum allowed roof slope value (Fig. 11) (URL-3). However, this information could differ in various zoning regulations. This parameter could be set by the user or obtained from the attribution of 2D footprint data.

This extension of the SS algorithm directly gives the Hipped type roof geometry. It needs a modification if the subject is a Gable type of roof. The closest skeleton nodes to the start and end edges of the building footprint must be extended onto these edges. The slope value becomes 90% on these surfaces. If the subject is a Flat type of roof, there is no need to use the SS algorithm and the roof geometry is directly the same as the building footprint.

SS and sweep line have been implemented via web services as a web-based solution using RESTful architecture. The web application consists of 3 sub-modules: storage module, process module and web server module. The general system architecture is given below (Fig. 12). The application has been deployed on an Amazon S3 Bucket cloud storage environment. 2D building footprints can be uploaded to a MongoDB instance on the bucket via a rest service that has been developed using the Java Jersey web framework.

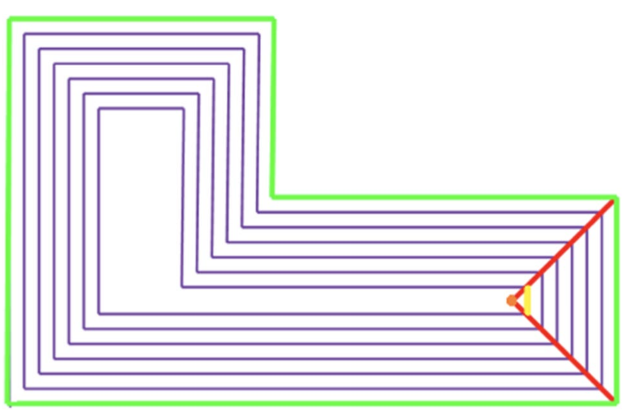

Fig. 8 Edge Event (Yellow edge shrinks to zero at the orange point)

Visualization of the 3DCM via browser

3DCMs are generally huge in size. Thus, clustering techniques should be used while passing them to clients from a server (Chen et al. 2016). Tiling approaches are used with web technologies such as HTML5 and WebGL to be able to visualize them via browsers. 3D roof geometries are added to 3D models of the tiles and with this, generated roofs are integrated into the tiling system. Thus, only needed tiles based on the user's current view in the scene are fetched from the server.

Since WebGL supports only triangles as primitives for the representation of surfaces, 3D polygons must be triangulated. An “Ear Clipping” algorithm is used for this purpose and 3D polygon surfaces that belong to roof geometries have been triangulated.

After the generation of the tileset, to visualize the tileset in the browser, the open-source JavaScript library Cesium.js is used in this study. Since Cesium.js is built on WebGL, 3DCM has been rendered using the client's GPU and without any additional plug-in (Fig. 13). The generation of the 3DCM which contains 1282 building objects is taken approximately 10 min. Including the tiling and visualization process which takes 2 min, the entire process takes approximately 12 min.

Findings and discussion

As a result of the described workflow, an application has been developed that enables the generation of LOD2 3DCMs using 2D building footprint data and aerial images, in a rapid and without-user-interference manner through a web-based interface. The test results for this application are presented in the following section of the text.

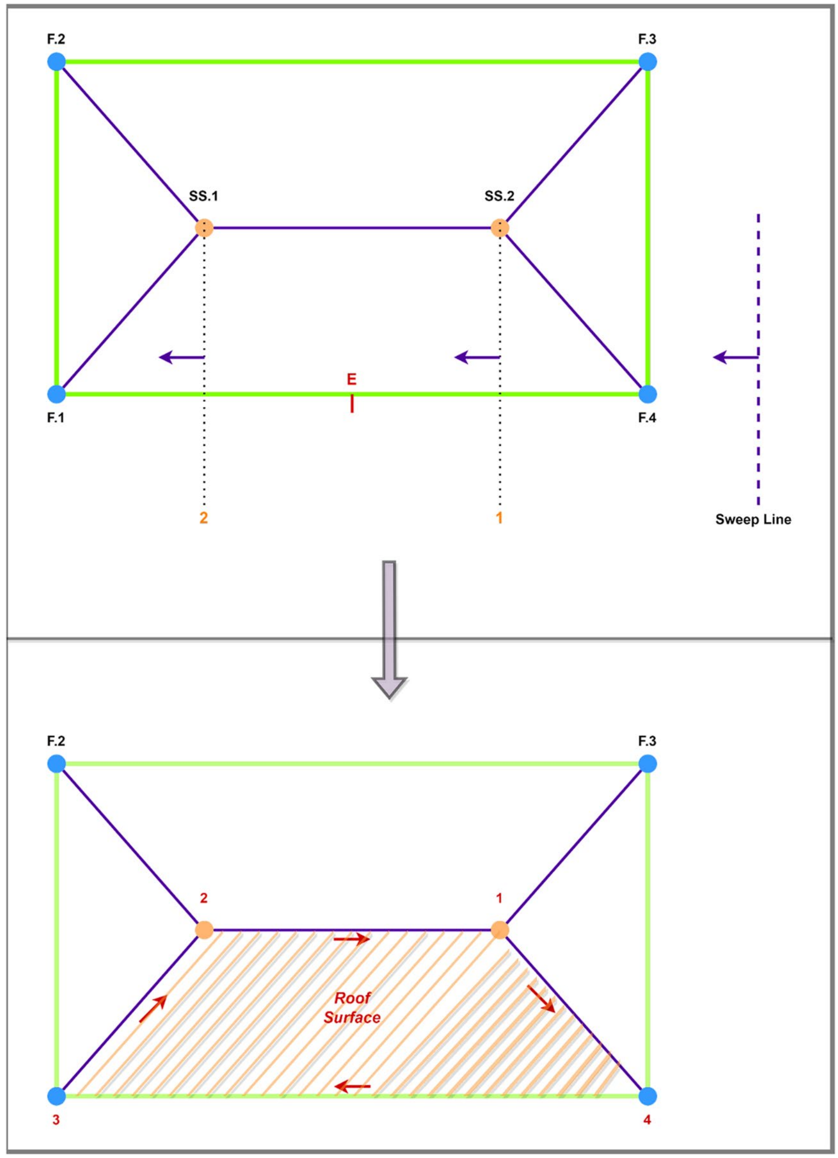

Fig. 10 Sweep Line Process. The orange nodes in the first row are belong to the SS. The purple dashed sweep line moves along the edge "E". They are ordered according to the time of intersection with the sweep line. The roof surface is demonstrated with orange diagonal lines in the second row

Performance metrics of the roof type prediction

3423 roof images that belong to hip (835), gable (1762) and flat (751) roofs were used in 0,3/0,7 of test/train ratio. The accuracy in predicting roof types is %89,9. Performance metrics are represented as AP (Average Precision), Recall and Precision (Fig. 14). The overall AP value of 0.89 proves the training is enough for prediction.

The proposed method has performed better than similar works in the related literature. The main reason for this result is that in this work, only 3 roof types have been classified while in (Biljecki and Dehbi 2019), as the most similar work, 6 different roof types are classified. They have used the geometries of LOD1 3DCM to obtain roof types. The use of cognitive processes with aerial imagery increases overall performance compared to (Biljecki and Dehbi 2019). The accuracy of prediction for individual roof types is given as a confusion matrix (Table 1). The train images, test images and written scripts for training and test phases could be found in (URL-2).

The comparison of predicted results with ground truth values could be found in Table 2. The precisions of the classes are not directly proportional with counts of images that are used to train the model. Because, as the quantity

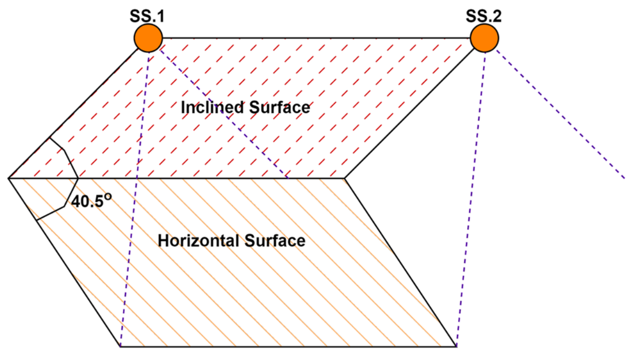

Fig. 11 The incline process according to the slope value (45% = 40.5 degrees). The orange points demonstrate the SS nodes

of the images affect the results, so does the quality of them.

The reason for having the lowest accuracy of the Gable class can be seen in the confusion matrix in Table 1. Considering the confusion matrix, it could be easily seen that the highest count of false negatives is obtained for the Gable class (29 false negatives). This situation can be also seen in the accuracy matrix, Table 2. The recall value of the Gable class is obtained as 0.77 and the precision value is obtained as 0.84. The mostly opened gap between these metrics is obtained for the Gable class and this gap is the reason for the lowest accuracy of the Gable class. The false negatives of the Gable class are mostly shared between the Hipped and Negative classes. To reduce the false negatives in the Gable class, or in other words, to increase the recall value of the class, the model should be trained with more images that contain confusable between the Gable, Hipped and Negative classes. As another option, data augmentation processes could be implemented to increase the training images and discrimination capability of the model.

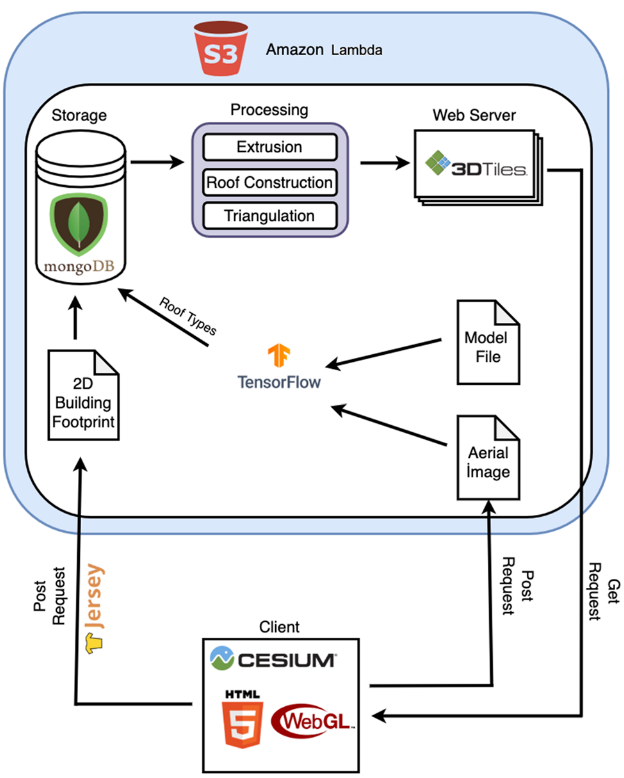

Fig. 12 General system architecture

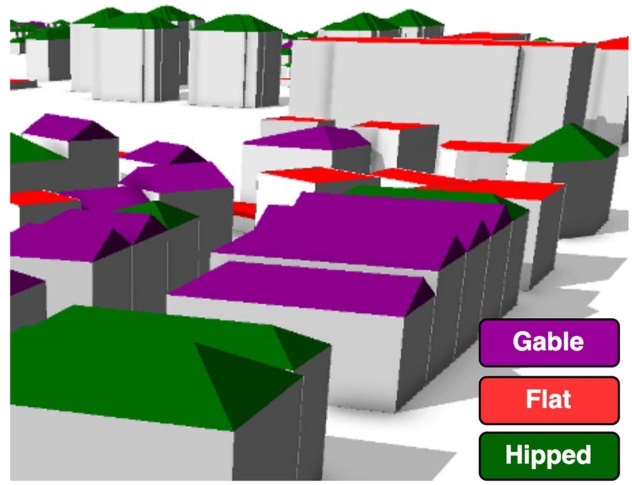

Fig. 13 Red roofs: Flats, Green roofs: Hippeds, Purple roofs: Gables

Special cases and floating-point issues

Implementing SS which requires high coordinate precision in a floating-point environment caused some degenerations that the construction algorithm must handle. While calculating intersections between planes and determining new positions of vertices, some vertices that must have exact same coordinates can have different coordinates with minimal difference due to the floating-point arithmetic. A tolerance value must be predefined to handle this situation, and vertices with small coordinate differences with this tolerance value must be unionized into one vertex. If these situations are not handled, these

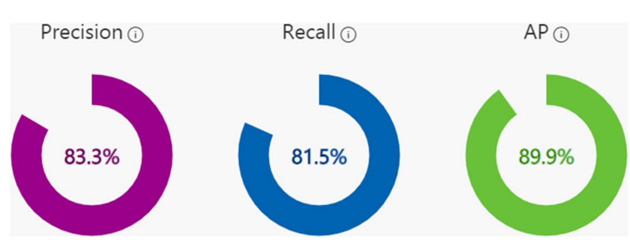

Fig. 14 Performance metrics of training process

Table 1 Confusion matrix of prediction process

	Flat	Gable	Hipped	Negative
Flat	53	5	4	0
Gable	8	96	10	0
Hipped	1	14	143	0
Negative	1	10	2	14
Total	63	125	159	14

slight differences can lead to tremendous changes in roof geometries (Fig. 15).

Limitation

Even if the web application can recognize the 3 most common roof types, it could be impossible to construct the roof properly when another roof type is the subject such as hexagonal, M-shaped, shed, or a combination of mentioned ones. The DL model classifies this different type of roof as Negative or one of the 3 main types which are the most similar to the roof and the roof construction process is done incorrectly. This construction of the roof geometries could cause incorrect results in analyses such as shadow, solar potential or visibility etc. Concerning this, the DL model's recognizing ability could be improved by retraining the DL model with new roof types and image samples.

Another limitation that can be mentioned about the developed application is the assumption of a fixed slope value of 45% for the 3D modeling of the roof. While general roof shapes can be generated using a 2D aerial image, slope of the roof surfaces can not be extracted from 2D data and requires additional resources to derive 3D geometry. The fixed slope value, which is used to be able to model roof surfaces in 3D, has been determined according to regulations in Turkey. It should be noted that this may vary for different regions when the application is used for different areas, for this purpose users can input their own fixed values.

Table 2 Performance metrics of prediction process

Class	Ground Truth	Prediction	Accuracy	Precision	Recall
Flat	63	62	94,74%	0,85	0,84
Gable	125	114	86,98%	0,84	0,77
Hipped	159	158	91,41%	0,91	0,9
Negative	14	27	96,40%	0,52	1

Conclusion

In this work, the generation of LOD2 3DCMs from 2D building footprints without any user interference is achieved by the developed architecture. This work introduces two novel contributions. Firstly, it utilizes deep learning (DL) techniques to automatically extract roof-type information from aerial images. This automated approach streamlines the process of gathering roof-type data. The second contribution involves the extension of a widely recognized Straight Skeleton algorithm by incorporating the Sweep Line algorithm. This enhanced algorithm enables the construction of 3D roof geometries,

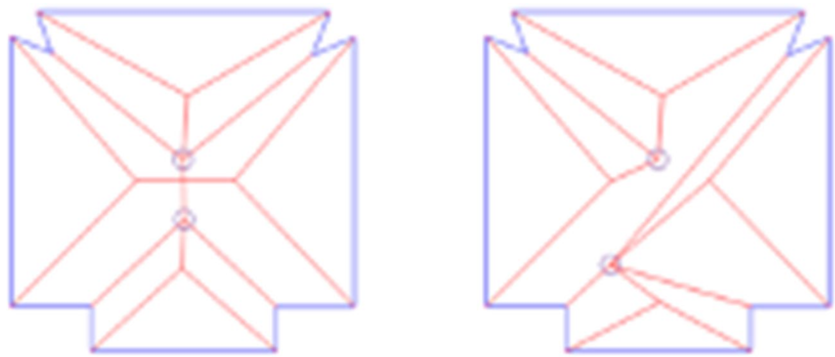

Fig. 15 Degeneration. Two Edge Events must meet on the same point (left) but they do not properly meet (right)

allowing for more detailed and accurate representations of roofs. Finally, this study establishes and implements a web-based architecture that facilitates the efficient generation of Level of Detail 2 (LOD2) building models. This architecture offers a rapid and practical solution for creating building models using the acquired roof-type information and improved 3D roof geometries. By virtue of the condominium unit plans, 3D models of condominiums could also be generated using our architecture, if the user provides unit plans.

Roof geometries were constructed using the DL-generated roof type information and the novel Straight Skeleton extension. To obtain the required roof type information for modelling roof geometries, a CNN model was trained and tested using the Microsoft Custom Vision platform. 3423 roof images that belong to hip (835), gable (1762) and flat (751) roofs were used in 0,3/0,7 of test/train ratio. The average precision was obtained as 89.9% and the class-wise average precisions were obtained over %84. The Gable class has a recall value of 0.77 and a precision value of 0.84. The largest disparity between these metrics occurs in the Gable class, resulting in its lowest accuracy. The false negatives in the Gable class are primarily misclassified as either Hipped or Negative classes. To improve the recall value and decrease false negatives in the Gable class, the model should be trained using additional images that exhibit similarities or confusion between the Gable, Hipped, and Negative classes. To enhance the recall value and minimize false negatives in the Gable class, there are two potential approaches. Firstly, the model should be trained using a larger set of images that exhibit similarities or confusion between the Gable, Hipped, and Negative classes. This expanded dataset would help the model better understand the distinguishing features of each class, improving its ability to accurately classify Gable instances. Additionally, implementing data augmentation techniques can be beneficial. Data augmentation involves generating synthetic training data by applying various transformations to existing images, such as rotation, scaling, or flipping. By augmenting the training dataset, the model gains exposure to a wider range of variations within each class, enabling it to discriminate more effectively and potentially reducing false negatives in the Gable class. Combining these two strategies, increasing the diversity of training images and employing data augmentation techniques, can help the model achieve higher recall for the Gable class and mitigate the occurrence of false negatives.

The implementation of the Straight Skeleton (SS) algorithm in a floating-point environment poses challenges due to the requirement for high coordinate precision. This can lead to degenerations in the algorithm's construction process, necessitating appropriate handling. Using a predefined tolerance value, our algorithm identifies vertices with small coordinate differences and then merges them into a single vertex. By unionizing vertices that fall within the tolerance range, the algorithm can effectively manage and mitigate the degenerations caused by limited floating-point precision.

The entire pipeline was taken approximately 12 min with our test data including 1282 buildings. Besides, the entire pipeline of the model generation is executed in the server and the results are observable in the client device even if it is a low-cost device. Furthermore, all software development stages and tests were realized using open-source development tools. Approximately 10 min of the total 12-min run time is spent on SS calculation and roof modeling. To reduce this time and accelerate the application, techniques such as multithreading, multiprocessing, or GPU-based processing acceleration can be implemented on the server side, where the algorithms run on, this optimization is a planned future work. Besides, the proposed pipeline for automatic modelling of roof geometries is currently limited to 3 classes: hip, gable and flat. Hence, there is also ongoing work on this topic, and this work will be extended to include modelling other roof types such as the hexagonal, shed etc. As another future plan, we will aim to detect and model chimneys on the roofs as well. The detection and modelling of chimneys could be beneficial for the optimization of solar panel implementation in detailed solar analyses.

Acknowledgements We would like to express our sincere gratitude to the Scientific and Technological Research Council of Türkiye (TUBITAK) for their support, their assistance has been instrumental in enabling the successful completion of this study.

Author Contribution Ziya Usta and Alper Tunga Akın performed algorithmic development stages and prepared figures. Ziya Usta, Alper Tunga Akın and Çetin Cömert wrote the main article text.

Funding This work was supported by The Scientific And Technological Research Council Of Türkiye (TUBITAK) with the grant ID of 118Y452.

Data availability The data and code used in this study are available at https://github.com/alpertungakin/Roof-Type-Classification.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

Consent for publication If the article is accepted by the Editor-in-chief after the review process, all authors consent to the manuscript being published in Earth Science Informatics. The work did not include human participants in order to obtain their consent.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Man´e D, Monga, R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Vi´egas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X. (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. https://www.tensorflow.org/

Aichholzer O, Aurenhammer F, Alberts D, Gartner B (1996) A novel type of skeleton for polygons. In: J. UCS The Journal of Universal Computer Science, pp. 752–761

Alomía G, Loaiza D, Zúñiga C, Luo X, Asorey-Cacheda R (2021) Procedural modeling applied to the 3D city model of bogota: a case study. Virt Real Intell Hardw 3(5):423–433

Arroyo Ohori K, Ledoux H, Stoter J (2015) A dimension-independent extrusion algorithm using generalised maps. Int J Geogr Inf Sci 29(7):1166–1186

Axelsson M, Soderman U, Berg A, Lithen T (2018) Roof type classification using deep convolutional neural networks on low resolution photogrammetric point clouds from aerial imagery. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1293–1297

Bauchet J-P, Lafarge F (2019) City reconstruction from airborne lidar: a computational geometry approach. In: 3D GeoInfo 2019–14th Conference 3D GeoInfo

Bengio Y, Goodfellow I, Courville A (2017) Deep Learning, vol 1. MIT press Cambridge, MA, USA

Biljecki F, Dehbi Y (2019) Raise the roof: Towards generating LOD2 models without aerial surveys using machine learning. ISPRS Annals Photogramm Remote Sens Spat Inf Sci 4:27–34

Biljecki F, Ledoux H, Stoter J (2017a) Does a finer level of detail of a 3d city model bring an improvement for estimating shadows? In: Advances in 3D Geoinformation, pp. 31–47

Biljecki F, Ledoux H, Stoter J (2017b) Generating 3d city models without elevation data. Comput Environ Urban Syst 64:1–18

Biljecki F, Ledoux H, Stoter J (2016) Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. ISPRS Ann. Photogramm Remote Sens Spatial Inf Sci 3:51–59

Biljecki F, Stoter J, Ledoux H, Zlatanova S, Çöltekin A (2015) Applications of 3d city models: State of the art review. ISPRS Int J Geo Inf 4(4):2842–2889

Bradbury K, Brigman B, Collins L, Johnson T, Lin S, Newell R, Park S, Suresh S, Wiesner H, Xi Y (2016) Aerial imagery object identification dataset for building and road detection, and building height estimation. figshare

Buyukdemircioglu M, Kocaman S, Kada M (2022) Deep learning for 3D building reconstruction: A review. Int Arch Photogramm Remote Sens Spat Inf Sci, 359–366

Çağdaş V (2012)Taşınmaz Değerlemesi Ders Planı, https://slideplayer.biz.tr/amp/3151467/

Castagno J, Atkins EM (2018) Automatic classification of roof shapes for multicopter emergency landing site selection. In: 2018 Aviation Technology, Integration, and Operations Conference, p. 3977

Chen J, Li J, Li M (2016) Progressive visualization of complex 3D models over the internet. Trans GIS 20(6):887–902

Dehbi Y, Henn A, Gröger G, Stroh V, Plümer L (2021) Robust and fast reconstruction of complex roofs with active sampling from 3D point clouds. Trans GIS 25(1):112–133

Demir C, Koramaz TK (2018) GIS-based procedural modeling in contemporary urban planning practice. In 2018 22nd International Conference Information Visualisation (IV) (pp. 553–560). IEEE

Eppstein D, Erickson J (1999) Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. Discret Comput Geom 22(4):569–592

Felkel P, Obdrzalek S (1998) Straight skeleton implementation. In: Proceedings of Spring Conference on Computer Graphics. Citeseer

Huang, J. (2019) A 3D GIS-based valuation system for assessing the scenic view in residential property valuations

Kelly T (2014) Unwritten procedural modeling with the straight skeleton. PhD thesis, University of Glasgow

Kelly T (2021) CityEngine: an introduction to rule-based modeling. Urban informatics, 637–662

Khayyal HK, Zeidan ZM, Beshr AA (2022) Creation and spatial analysis of 3D city modeling based on GIS data. Civ Eng J 8(1):105

Ledoux H, Meijers M (2011) Topologically consistent 3d city models obtained by extrusion. Int J Geogr Inf Sci 25(4):557–574

Martinovic A (2015) Inverse procedural modeling of buildings

Nurkarim W, Wijayanto AW (2023) Building footprint extraction and counting on very high-resolution satellite imagery using object detection deep learning framework. Earth Sci Inf 16(1):515–532

Partovi T, Fraundorfer F, Bahmanyar R, Huang H, Reinartz P (2019) Automatic 3-d building model reconstruction from very high resolution stereo satellite imagery. Remote Sens 11(14):1660

Salvaris M, Dean D, Tok WH (2018) Cognitive services and custom vision. In: Deep Learning with Azure, pp. 99–128

Senyurdusev G, Dogru AO (2020) An Alternative Approach For Creating 3d City Model Using Procedural Modeling, In Proceedings of the 8th International Conference on Cartography and GIS, Nessebar, Bulgaria

Senyurdusev G, Dogru AO (2020) Ulugtekin, N. N. Exploring the opportunities of open source data use in creation 3d procedural city models. In Proceedings of the 8th International Conference on Cartography and GIS, Nessebar, Bulgaria (pp. 15–20)

Souvaine D (2005) Line segment intersection using a sweep line algorithm. Tufts University

Tekavec J, Lisec A, Rodrigues E (2020) Simulating Large-Scale 3D Cadastral Dataset Using Procedural Modelling. ISPRS Int J Geo Inf 9(10):598

Tomljenovic I, Höfle B, Tiede D, Blaschke T (2015) Building extraction from airborne laser scanning data: An analysis of the state of the art. Remote Sens 7(4):3826–3862

Tsiliakou E, Labropoulos T, Dimopoulou E (2014) Procedural modeling in 3d gis environment. Int J 3-D Inf Model (IJ3DIM) 3(3):17–34

URL-1: https://vc.systems/en/products/building-reconstruction/

URL-2: https://github.com/alpertungakin/Roof-Type-Classification

URL-3: https://www.resmigazete.gov.tr/eskiler/2018/05/20180520-4.htm

Weiler V, Stave J, Eicker U (2019) Renewable energy generation scenarios using 3d urban modeling tools—methodology for heat pump and co-generation systems with case study application. Energies 12(3):403

Welle Donker F, Van Loenen B, Bregt AK (2016) Open data and beyond. ISPRS Int J Geo Inf 5(4):48

Wendel J, Simons A, Nichersu A, Murshed SM (2017) Rapid development of semantic 3D city models for urban energy analysis based on free and open data sources and software. In Proceedings of the 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics (pp. 1–7)

Yang X, Delparte D (2022) A Procedural Modeling Approach for Ecosystem Services and Geodesign Visualization in Old Town Pocatello. Idaho Land 11(8):1228

Zhang M, Wu J, Liu Y, Zhang J, Li G (2022) GIS Based Procedural Modeling in 3D Urban Design. ISPRS Int J Geo Inf 11(10):531