PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science
https://doi.org/10.1007/s41064-024-00319-6

ORIGINAL ARTICLE

Multi-branch convolutional neural network in building polygonization using remote sensing images

Yajin Xu[1] · **Philipp Schuegraf**[1] · **Ksenia Bittner**[1]

Received: 25 November 2023 / Accepted: 6 September 2024

Abstract
Building extraction and polygonization is important for urban studies, such as urbanization monitoring, urban planning. Remote sensing images, especially in RGB bands, provide sufficient semantic information which is useful for the task of building extraction and polygonization. Deep learning using Convolutional Neural Networks (CNNs) is proven to be successful in many fields, including building extraction from remote sensing images. In this paper, we propose a two-stage method to solve the task of building polygonization from remote sensing images based on deep learning. Firstly, we decompose a 2-D building footprint model into three basic geometry primitives. Leveraging stacked Multi-Branch Modules (MBMs), we separate the task of building extraction into tasks of predicting the three geometry primitives using our proposed CNN. At the second stage, we propose an efficient enhanced building polygonization and adjustment algorithm to generate the final building polygons. This algorithm is able to handle both building blocks and individual buildings. We evaluate our model on three open datasets. For building blocks, our model achieved average precision of 62.7% and average recall of 73.6% on the CrowdAI mapping challenge dataset, and 13.9% and 24.4% respectively on the Urban Building Classification (UBC) dataset which contains mainly individual buildings. On the Inria aerial image dataset, the proposed method achieved Intersection over Union (IoU) over 71%.

Keywords Building extraction · Deep learning · Computer vision · Remote sensing · Geoscience

1 Introduction

Buildings are one of the most important aspects in urban studies, such as studies of urbanization progress, urban changes. With remote sensing images, surveys of buildings have become more efficient, since remote sensing images are capable of covering much larger areas. Building extraction using deep learning is proven to be powerful and successful, by introducing computer vision knowledge into remote sensing domain.

With the development of deep learning and Convolutional Neural Networks (CNNs), the general workflow of building extraction can be described as first to extract high-level deep image features using a backbone network, and then to delineate building footprints. The spatial dimension of the extracted features can be the same of the original input image, e.g. U-Net (Ronneberger et al. 2015), or smaller than the input image, e.g. Hourglass network (Newell et al. 2016). Usually, the delineation is done by semantic segmentation with post-processing refinement (Zhao et al. 2018; Mahmud et al. 2020) or by edge/corner extraction with additional Graph Neural Network (GNN) models (Zhao et al. 2022; Alidoost et al. 2020; Zorzi et al. 2022). Additionally, when the spatial dimension of the extracted features is smaller than input, up-sampling of prediction images is necessary to generate final building footprints.

In contrast to building footprints (in raster format), building polygons in vector format are more efficient to use in practice, for only vertex coordinates are involved for such representation. Geometrically speaking, a building polygon consists of three primitives: vertices, edges, footprint. This inspired us to extract building polygons by first extracting these three geometry primitives from the image. This is

Yajin Xu
yajin.xu@outlook.com

Philipp Schuegraf
philipp.schuegraf@dlr.de

✉ Ksenia Bittner
ksenia.bittner@dlr.de

[1] German Aerospace Center (DLR), Münchener Straße 20, 82234 Weßling, Germany

Published online: 25 November 2024

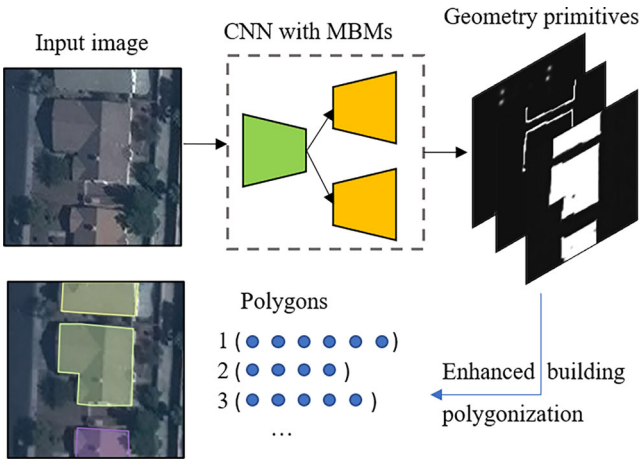

Fig. 1 The overall workflow of our proposed method. Each RGB image is fed to a CNN with MBMs to obtain the three predicted basic geometry primitives. The enhanced polygonization module outputs building polygons based on the three predictions

similar to the methods which extract building edges and/or corners.

The three aforementioned geometry primitives are highly coupled with each other in nature. Therefore, we seek to develop a model which enables information fusion of the three geometry primitives in the task of building extraction and polygonization. Multi-Branch Module (MBM) (Batra et al. 2019) is proven to be successful in enhancing road extraction from satellite imagery, by providing an explicit information exchange of predicted road network and predicted orientation of roads. Inspired by this, we extend and further develop the MBM with modifications in this paper and propose our own model for building extraction. MBMs provide learned interactions in training between each branch responsible for one basic geometry primitive type. These geometry primitives are then used for later building polygonization.

In literature, there are two possible types of building segments: building blocks and individual buildings. The difference is whether a building is considered two individual buildings when Common Borders (CBs) exist, i.e. shared common edges. For the extraction of building block segments, it is usually a semantic segmentation task, while the extraction of individual buildings with CBs is seen as an instance segmentation task. We intend to include both building segment types and develop a method that works in both scenarios.

Following building extraction, we propose an efficient building polygonization procedure aiming to generate building polygons with quality close to manual delineation, which allows our model to handle both building blocks and individual buildings. This distinction of building segment types is studied explicitly in our work by presenting our evaluation results on two different datasets, i.e. the CrowdAI dataset (Mohanty et al. 2020) which focuses on building blocks and the Urban Building Classification (UBC) dataset (Huang et al. 2022) which focuses on individual buildings. We also include the Inria aerial image dataset (Maggiori et al. 2017) to test our proposed method with higher building variations in sense of building density, building shapes, etc.

Our method consists of two stages (Fig. 1). The first stage involves basic geometry primitives prediction with same spatial resolution as input images using CNN with MBMs, and the second stage is an enhanced building polygonization process based on predicted geometry primitives. Our method has the advantage in favour of simplicity and straightforward interpretation, and is capable of generating polygons directly based on remote sensing images. The main contributions of our work are as follows:

- we extend the MBM to three branches with modifications, and demonstrate that learned interaction is helpful in learning of the three basic geometry primitives for building polygonization;
- we propose an efficient enhanced building polygonization algorithm which produces polygons with sharp corners and straight edges;
- we develop a method which is capable of handling both building blocks and individual buildings;
- we tested our method on three public datasets, i.e. CrowdAI (Mohanty et al. 2020), UBC (Huang et al. 2022) and Inria dataset (Maggiori et al. 2017) and provide our evaluation results as baselines for comparison.

The rest of this paper is organized as follows: Sect. 2 presents existing models and methods related to our work, Sect. 3 describes in detail our proposed method, Sect. 4 describes our experimental settings and implementation details, Sect. 5 presents experiment results, Sect. 6 presents our general discussions, and Sect. 7 concludes our work with final remarks regarding limitations.

2 Related work

In this section, related neural network models and methods for extraction of building polygons using remote sensing images are summarized and categorized into building representation based and geometry primitives based methods. Different neural networks and network designs related to the task of building polygonization are first presented, followed by the introduction and discussion of the two types of methods.

2.1 Building polygonization

In general, the task of building polygonization using remote sensing images is tackled by first extracting image features using backbone networks, followed by method-specified predictions. In most works, three kinds of representations are set as learning targets, i.e. as maps of building vertices, edges and footprints. These three basic geometry primitives are essential parts of a building. Some research (Girard and Tarabalka 2018; Zhao et al. 2020) regress coordinates of building vertices and output polygons directly without intermediate outputs of any of the three geometry primitives. However, the regressed coordinates are not trustworthy, since they are predicted completely by neural networks without any controls. Therefore, we prefer the methods based on geometry primitives, which are more robust in generating correct building polygons. In this work, we focus only on the methods based on geometry primitives.

We categorize the existing methods into two classes. The first class of methods aims to generate building polygons using different representations of buildings, e.g. building footprints, followed by regularization for straight building edges, sharp corners, etc. The second class uses geometry primitives and construct building polygons based on predictions. Depending on definition of building instances, two types of building segments are studied: building blocks and individual buildings. The main difference is that for individual buildings, two buildings with CBs are seen as individual instances, while for building blocks, they are treated as one building instance. We do not distinguish the types of building segments explicitly that are studied in each work unless specified.

2.2 Backbone networks

Most research requires a backbone network to extract deep image features. Different designs of networks and modules are usually adopted simultaneously. The encoder-decoder structure proposed in U-Net (Ronneberger et al. 2015) is capable of capturing both local and global features, and is adapted for building footprint extraction (Liu et al. 2019; Li et al. 2021). The residual connection (residual block) in ResNet (He et al. 2016) provides direct identity mapping, i.e. adding the module input to the module output, which improves deeper network performance, and is integrated in many networks (Mahmud et al. 2020; Alidoost et al. 2020). To obtain predictions with same size of the input images, a path of down-sample-up-sample is usually designed. In order to better retain information of object structures and shapes in down-sample and up-sample procedures, intermediate supervision is studied by Xie and Tu (2015). With intermediate supervision, not only the final outputs of networks but also the intermediate layer outputs are supervised. In this way, the spatial structures and shapes of objects are more consistent at different feature levels. Based on the reported performance of these designs in literature, we will adapt the aforementioned architectures and modules in our own design.

Other than CNNs, GNNs are also studied, especially for polygonizing the extracted building segments (Zhao et al. 2022; Zorzi et al. 2022). These methods usually start with a CNN with appended GNNs for down-stream tasks. Graph Convolutional Network (GCN) (Kipf and Welling 2016) adapts the idea of image convolution in graph domain. In GCNs, information exchange between nodes is done by "convolving" among connected nodes. GCN is already used to learn to separate foreground and background for extracted potential roof lines (Zhao et al. 2022). Similar to the idea of GCN, Graph Attention Network (GAN) enables information exchange between nodes through a so-called "self-attention" mechanism, and is also proven to be useful in building polygonization (Zorzi et al. 2022).

In all, the GNNs based methods consist of combined networks, i.e. leading CNNs to extract image features followed by GNNs for polygonization, which leads to higher computational requirements. Therefore, we focus more on CNNs based methods for a more efficient building polyonization procedure.

2.3 Representation learning based methods

With the extracted image features from backbone networks, representation learning based methods seek to predict building-related representation maps. The most straightforward building representation is its footprint, transforming the building polygonization problem into a semantic or instance segmentation problem. For semantic or instance segmentation based methods, prediction heads are attached to the backbone network and output building segments. The output is then refined to generate the final building polygons. This refinement is treated as a problem of regularization, and carried out through polygon regularization (Zhao et al. 2018), automatic regularization by introducing another CNN (Zorzi et al. 2021; Liu et al. 2019), or by height filtering based on Digital Surface Model (DSM) (Sun and Wang 2018). However, the polygon refinement process in these research either requires extra external inputs or additional computation.

Another approach is to generate a different representation of building footprints, instead of a segmentation mask, as training target. This representation also serves as input to the following polygonization part, avoiding extra external inputs. For example, frame field is the representation, in which two orthogonal directions to connected pixels are calculated at each location, and it is adapted for building poly-

gonization (Girard et al. 2021). Wang and Frahm (2017) converted each pixel into vectors describing 3D cuboids and extracted the cuboid with the highest score as predicted building footprint. Similarly, Qian et al. (2021) encoded each roof as 4D vectors with facade facings and this representation is learned by a CNN. The idea of transforming building footprints into other representative models is adapted in our design, in which we use building footprint, building edges and building vertices as a complete building model.

2.4 Geometry primitives based methods

In contrast to the representation learning based approaches, the strategies which extract directly geometry primitives from images for building polygonization seem more straightforward, and are able to output polygons directly. Instead of generating segmentation maps, Alidoost et al. (2020) extracted classified rooflines and used these lines to reconstruct roof planes. Interestingly, Wu et al. (2023) used rotated bounding boxes to represent building edges for building footprint reconstruction. Another approach is to solve the polygonization problem by connecting corner points in a series manner by first predicting building vertices (Castrejón et al. 2017; Li et al. 2019; Hu et al. 2023), but with more intensive computation for its iterative procedure. Extracting both edges and vertices has also been studied by Wang et al. (2021). These methods are applied to building blocks only, and do not study individual buildings in detail.

Specially for individual building extraction, Schuegraf et al. (2022) investigated existing models to extract CBs together with building segments, and obtained individual buildings using watershed transform. With the help of frame field representation, Girard et al. (2021) proposed active skeleton model for building polygonization based on predicted building edges to handle individual buildings.

GNNs are useful for polygonization problems, since vector data and graphs have very similar structures. For GNNs based approaches, a backbone network (mostly a CNN) is applied to extract useful image features as well as location of vertices. These vertices are then converted into a graph and fed to a GNN. For example, Zorzi et al. (2022) first extracted building vertex candidates, and the connections between vertices are learned by a GNN. Zhao et al. (2022) demonstrated that edges can also be converted into graphs and passed to a GNN for reconstructing roof structures. However, it is reported in the aforementioned methods that when the connection is wrongly predicted, polygons will collapse, resulting in completely wrong building polygons. Therefore, a more robust method with less computational complexity is still desired.

2.5 Joint learning

To obtain building polygons, multiple components are usually needed. Therefore, joint learning, or multi-task learning, is adapted (Li et al. 2021; Zhao et al. 2022). These components are mainly building vertices, edges and footprints. The motivation of adapting joint learning is to enable better interaction and information fusion of different branches for different learning targets.

Coupled losses are studied to encourage consistency between different outputs (Girard et al. 2021), which also provides a certain degree of information exchange. Recently, Xu et al. (2023) used signed distance map as prior, which contains implicitly information of building segments and corners. Another approach (Batra et al. 2019) is to provide an earlier fusion of different network outputs, adapting intermediate supervision and network stacking (Newell et al. 2016) simultaneously. This architecture consists of multiple stacked MBMs, and each MBM contains one encoder and several decoders. The outputs of MBMs are supervised, as well as the final outputs of the whole network. MBMs enable automatically learned interaction across different branches, i.e. different learning targets, and are proven to be successful in closing gaps in road network prediction (Batra et al. 2019). These techniques and architectures inspired us in designing our proposed network.

3 Methodology

In this section, the adapted MBM is first explained in detail, followed by our overall network design. The proposed enhanced building polygonization method is elaborated as the second stage of our proposed method.

Overall, our method takes as input RGB remote sensing images, and first outputs predictions of building vertex density maps, building edge maps and building footprint maps. The training of network is supervised, with learning targets being normalized vertex density maps of reference building vertices, binary maps of building edges and binary maps of building footprint, respectively. The second stage takes the three predictions as inputs, and outputs refined enclosed polygons for each building, i.e. an ordered list of building vertex coordinates.

3.1 Multi-branch module and reasoning

We adapt the idea of MBM (Batra et al. 2019) with our own modifications and design a network suitable for our task. The whole network architecture is shown in Fig. 2. Without loss of generality, we discuss here how our network processes the input image in a universal manner without restrictions of number of stacks or branches.

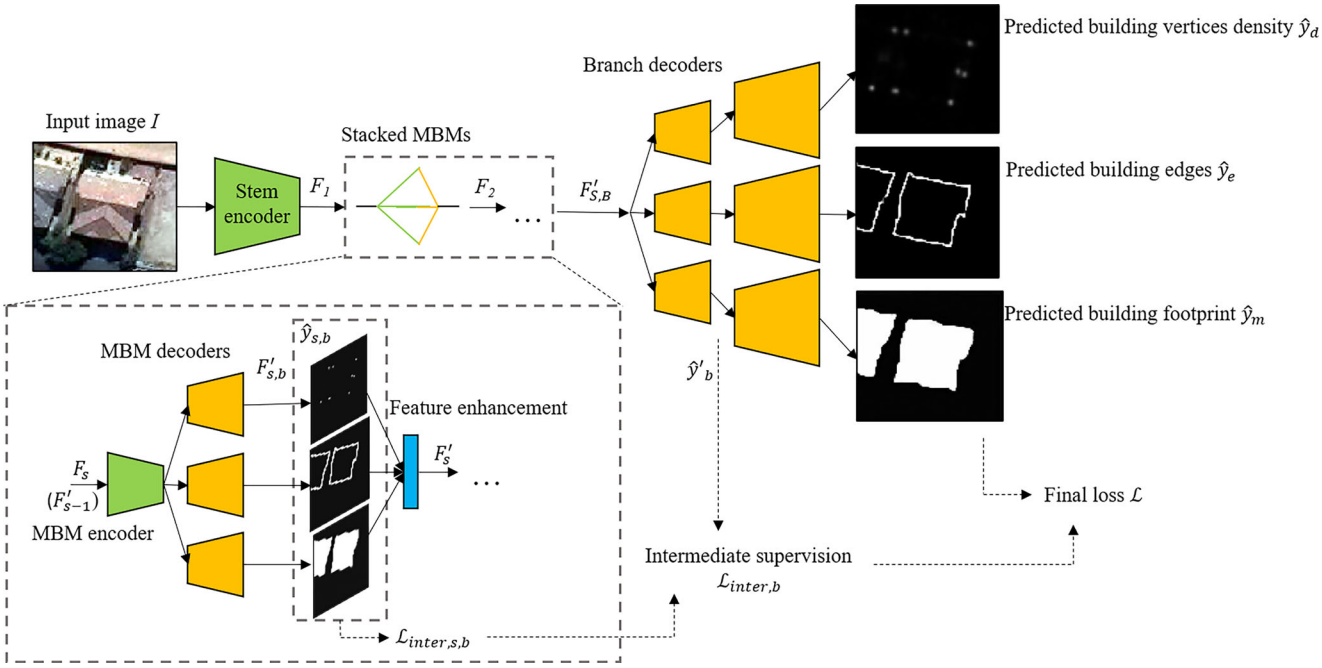

Fig. 2 Architecture of our designed network. The whole network consists of three parts: stem encoder, stacked MBMs and branch decoders. The stem encoder first maps the input image into feature space with certain dimension, and down-samples to one fourth of the original spatial dimension of the input image. The stacked MBMs refine the latent features by allowing information flow across different branches. In the end, the branch decoders up-sample the features and make predictions with spatial dimension of the original input. Each up-sample path contains skip connections from the corresponding down-sample path across the whole network, and intermediate supervision is used in the feature enhancement block

For a given image $\mathbf{I}$ of shape (C, H, W) (for simplicity, we omit the batch number in notations), where C is the number of channels, H and W are the spatial dimensions of height and width respectively, the stem encoder first down-samples the input image twice, with several residual blocks (He et al. 2016) between the first 2-D convolutional layer with stride 2 and the next max pooling layer, as "Conv2d–BatchNorm2d–ReLU–residual blocks–MaxPool–residual blocks". Rectified Linear Unit (ReLU) activation function is used unless specified explicitly. The output after down-sampling is denoted as F_1 of shape $\left(D, \frac{H}{4}, \frac{W}{4}\right)$, where D is the feature dimension. F_1 is then fed to multiple sequentially stacked MBMs.

Inside each MBM, the input is denoted as F_s, where $s = 1, 2, \ldots, S$ is the index of the MBM, and S is the total number of stacked MBMs. Each MBM contains one MBM encoder and multiple MBM decoders. The number of decoders is equal to the number of branches. MBM encoder first further down-samples the input F_s three times with residual blocks between two consecutive max-pooling layers, and MBM decoders up-sample the output from the MBM encoder to the same spatial dimension of F_s. For each branch, there is one corresponding MBM decoder. We use skip connection (Ronneberger et al. 2015) between MBM encoder and each MBM decoder.

Next, we need to fuse and enhance the extracted features from different branches. We adopt intermediate supervision for each branch to support and encourage structure consistency in high-level feature space. In the end of each MBM decoder, we attach a prediction head which predicts corresponding intermediate prediction $\hat{y}_{s,b}$ of shape $\left(1, \frac{H}{4}, \frac{W}{4}\right)$ using the features $F'_{s,b}$ extracted by the bth decoder, where $b = 1, 2, \ldots, B$ is the index of each branch, and B is the total number of branches. The intermediate predictions are supervised by reference images y''_b.

In previous work (Batra et al. 2019), the features $F'_{s,b}$ are fed to a layer consisting of 1×1 convolution to reduce the channel dimension to 1 (denoted as $\bar{y}_{s,b}$) and passed to the corresponding activation functions for intermediate predictions $\hat{y}_{s,b}$. The mapped features $\bar{y}_{s,b}$ are mapped back to the same dimensions as $F'_{s,b}$ and added up with $F'_{s,b}$ before passing to the next MBM. However, we believe that the enhanced features F'_s only, which contain coupled information from different branches, should be used, instead of reusing features from different branches. Discarding the separated features from different branches, features extracted from the joint intermediate predictions only are more representative for all branches, which is expected to improve feature fusion and information exchange among branches. Therefore, we modify the MBM in the previous work and obtain

the enhanced and coupled features from the concatenated intermediate predictions as

$$F'_s = f\left(\bar{y}_{s,1} \oplus \bar{y}_{s,1} \oplus \ldots \oplus \bar{y}_{s,1}\right), \tag{1}$$

where $f(\cdot)$ consists of a 1×1 convolutional layer, batch normalization layer and a ReLU non-linearity layer, and $\oplus$ is concatenation.

In this way, the information from different branches are coupled into the enhanced features without reusing separated branch features $F'_{s,b}$. The enhanced features F'_s are then fed to the next MBM, i.e. $F_{s+1} = F'_s$. The effectiveness of our design is also proved through experiments.

The extracted features from the final MBM $F'_{S,B}$ are of shape $\left(D, \frac{H}{4}, \frac{W}{4}\right)$. These features are then fed to different branch decoders. Each branch decoder is responsible for one branch. The branch decoders up-sample the input to original spatial dimension (H, W), with intermediate prediction $\hat{y}'_b$ of shape $\left(1, \frac{H}{2}, \frac{W}{2}\right)$ supervised by the corresponding reference images y'_b. Prediction heads are attached in the end of each branch decoder, and output corresponding predictions for each branch. The detailed implementation will be elaborated in Sect. 3.3.

The most obvious advantage of using MBM is that it allows information to flow between different branches. The three basic geometry primitives of a polygon are highly coupled with each other. Therefore, the information for these three basic geometry primitives should interact with each other not only in geo-spatial dimensions but also in latent feature space. MBM enables such interaction by fusing the deep features from three branches representing the three basic geometry primitives. Additionally, our modification realizes feature enhancement without reusing separate features from different branches, which is expected to improve interaction and information fusion among branches.

For the task of building polygonization, we use three branches in total to handle the three basic geometry primitives, namely building vertices branch, building edges branch and building footprint branch. In the rest of the paper, we use v, e and m as branch indices as $b \in \{v, e, m\}$ to refer to the three branches respectively.

3.2 Building vertex density regression

The design of the building vertices branch is slightly different than the other two branches. For building edges and building footprints, learning directly binary maps is less problematic than learning a binary building vertices heatmap, due to its extremely high imbalance of positive and negative samples, i.e. there are much more background pixels than building vertex pixels. Therefore, instead of using binary masks of building vertices as training targets, we apply a Gaussian function to the binary masks and set the training targets for the building vertices branch as building vertex density maps, inspired by Bahmanyar et al. (2019), to help mitigate the problem from highly imbalanced samples.

Building vertex density maps are calculated based on a 2D Gaussian function as

$$g\left(\mathbf{p}, \mathbf{p_0}, \sigma\right) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-i_0)^2 + (j-j_0)^2}{2\sigma^2}\right), \tag{2}$$

where $\mathbf{p} = (i, j)$ is any location in an image, $\mathbf{p}_0 = (i_0, j_0)$ is the coordinate of peak value, i.e. the coordinate of reference building vertex, and σ controls the shape of the Gaussian function.

Based on Eq. 2, building vertex density map is then calculated with given reference vertices as

$$\tilde{y_d}(i, j) = \max_{n=1,2,\ldots,N_p} \left(g\left(\mathbf{p}, \mathbf{p}_n, \sigma\right)\right), \tag{3}$$

where $\mathbf{p}_n$ is the coordinate of each reference vertex, with N_p being the total number of reference vertices.

Using Eq. 3 for each reference building vertex at all locations in the image, the building vertex density maps as reference are obtained. By taking the maximal response value instead of summing up, each peak is retained, avoiding high density values at non-peak locations accumulated by very dense vertices.

Note that the center of each pixel, where each reference vertex falls into, is not necessarily the location of each peak. The peaks always coincide with locations of reference building vertices. Theoretically, this setting allows sub-pixel accuracy by estimating 2D Gaussian distributions from the predicted vertex density maps. This is not studied in detail in this work, since the provided image coordinates of building vertices are integer numbers without sub-pixel precision.

The learning targets for the building vertices branch are chosen to be the normalized building vertex density maps, such that each peak has value 1, to be consistent with the learning targets for the other branches, as

$$y_d = \frac{\max\left(\tilde{y_d}\right) - \tilde{y_d}}{\max\left(\tilde{y_d}\right)}. \tag{4}$$

3.3 Network implementation

In this work, the three basic geometry primitives are represented in image domain as binary masks. More specifically, binary maps of building vertices, building edges and building footprints are used as reference data y''_b, y'_b, where $b \in \{v, e, m\}$. An exception is for the final output from the building vertices branch decoder, where the normalized

building vertex density maps y_d are used as supervision, while for the building edges branch and the building footprint branch, binary maps y_e and y_m are used respectively.

Prediction heads are added for intermediate and final supervisions. Each prediction head consists of a 2D convolutional layer with kernel size 1, batch normalization layer, and an activation function. For the intermediate predictions $\hat{y}_{s,b}$ and $\hat{y}'_b$, sigmoid activation function is used for all branches. For the final predictions $\hat{y}_b$, sigmoid activation function is used for the building edges branch and the building footprint branch, while ReLU is used for the building vertices branch, for the learning targets being density maps.

For network training, we used different losses for different branches and for different stages. For the intermediate predictions $\hat{y}_{s,b}$ inside MBMs and $\hat{y}'_b$ inside branch decoders, the reference images are correspondingly downsampled (using max pooling) binary maps of building vertices, edges and footprints. Dice loss $\mathcal{L}_d$ (Drozdzal et al. 2016) and focal loss $\mathcal{L}_f$ (Lin et al. 2017) are chosen for all three branches for supervision for their better performance than the Binary Cross-Entropy (BCE) loss when faced with class imbalance.

Focal loss (Lin et al. 2017) is developed upon BCE loss, calculated as

$$\mathcal{L}_f\left(\hat{y}, y\right) = \alpha\left(1 - \exp\left(\mathrm{BCE}\left(\hat{y}, y\right)\right)\right)^{\gamma} \mathrm{BCE}\left(\hat{y}, y\right),$$

where $\hat{y}$ is prediction, y is reference, α and γ are hyperparameters and were set to 0.25 and 2 respectively (which are suggested by Lin et al. (2017), to which we refer the readers for more details), and BCE$(\cdot)$ is BCE loss given as

$$\mathrm{BCE}\left(\hat{y}, y\right) = -\frac{1}{hw}\sum\sum\left(y \cdot \log\left(\hat{y}\right) + (1-y) \cdot \log\left(1-\hat{y}\right)\right),$$

where h and w are the height and the width of prediction/reference images respectively, and $\cdot$ is element-wise multiplication.

Additionally, to further tackle the problem of imbalanced training samples, we also weight each predicted sample in the BCE calculation by the inverse of the total number of reference positive/negative samples in the focal loss as

$$\text{weighted BCE}\left(\hat{y}, y\right) = \sum\sum\left(\frac{1}{N_{pos}} y \cdot \log\left(\hat{y}\right) + \frac{1}{N_{neg}}(1-y) \cdot \log\left(1-\hat{y}\right)\right),$$

where N_{pos} and N_{neg} are the numbers of positive and negative reference samples respectively. Note that $N_{pos} + N_{neg} = hw$.

Dice loss (Drozdzal et al. 2016) aims to deal with highly imbalanced datasets and is calculated as

$$\mathcal{L}_d\left(\hat{y}, y\right) = \frac{1}{hw}\sum\sum\left(1 - \frac{2 \cdot \hat{y} \cdot y + 1}{\hat{y} + y + 1}\right).$$

To summarize, the intermediate loss for $\hat{y}_{s,b}$ and $\hat{y}'_b$ is given as

$$\mathcal{L}_{inter,b} = \frac{1}{S}\sum_{s=1}^{S} \mathcal{L}_{fd}\left(\hat{y}_{s,b}, y''_b\right) + \mathcal{L}_{fd}\left(\hat{y}'_b, y'_b\right), \tag{5}$$

where $\mathcal{L}_{fd}(\cdot) = \mathcal{L}_f(\cdot) + \mathcal{L}_d(\cdot)$ is the linear combination of focal loss and dice loss.

For the final predictions $\hat{y}_b$, the same focal loss and dice loss are used for the building edges branch and the building footprint branch, but for the building vertices branch, we use smooth L_1 loss $\mathcal{L}_{L_1}$, since the reference image is a vertex density map. It is defined mathematically as

$$\mathcal{L}_{L_1}\left(\hat{y}, y\right) = \frac{1}{hw}\sum\sum sL_1\left(\hat{y}_{i,j}, y_{i,j}\right),$$

$$sL_1\left(\hat{y}_{i,j}, y_{i,j}\right) = \begin{cases} \frac{1}{2}(y_{i,j} - \hat{y}_{i,j})^2 & \text{if } |y_{i,j} - \hat{y}_{i,j}| < 1, \\ |y_{i,j} - \hat{y}_{i,j}| - \frac{1}{2} & \text{otherwise.} \end{cases}$$

The total loss is then written as

$$\mathcal{L} = \sum_{b}^{\{v,e,m\}} \mathcal{L}_{inter,b} + \mathcal{L}_{L_1}\left(\hat{y}_d, y_d\right) + \sum_{b}^{\{e,m\}} \mathcal{L}_{fd}\left(\hat{y}_b, y_b\right). \tag{6}$$

3.4 Enhanced building polygonization and adjustment

The predicted segmentation map is still not ideal for direct polygonization, mainly due to irregular building shapes. Especially, the predicted segmentation map does not have any information on individual buildings with CBs. Thus, in this paper, we propose a novel algorithm to generate individual building polygons automatically, based on the predicted geometry primitives, i.e. the predicted footprint maps $\hat{y}_m$, the predicted vertex density maps $\hat{y}_d$ and the predicted edge maps $\hat{y}_e$.

Theoretically, individual building instances are extracted by thresholding the difference map $\hat{y}_\Delta = \hat{y}_m - \hat{y}_e$. However, the extracted line segments in $\hat{y}_e$ are not always connected with each other in practice, leading to polygons of building blocks instead of individual buildings, as is shown on the left in Fig. 3. In order to tackle this problem, we propose an algorithm to split building blocks into individual buildings, described in Pseudo-code 1.

Fig. 3 Example of building blocks and individual buildings. After our enhanced building polygonization, building blocks (**a**) are split into individual buildings (**b**)

Pseudo-code 1 Split building blocks

$\hat{y}_{\Delta} = \hat{y}_{\Delta} > \theta_{\Delta}$
$\hat{y}_{\Delta,o} = \hat{y}_{\Delta} \circ \mathbb{1}$
$\hat{y}_{\Delta,c} = \hat{y}_{\Delta} \bullet \mathbb{1}$
$\mathrm{CBs} = \hat{y}_{\Delta,c} - \hat{y}_{\Delta,o}$
for $(\mathbf{i},\mathbf{j}) \subseteq \mathrm{CBs}$ **do**
  $a, b = \mathrm{LSE}(\mathbf{i},\mathbf{j})$
  $(i', j') = (\mathrm{mean}(\mathbf{i}), \mathrm{mean}(\mathbf{j}))$
  $(\mathbf{i}', \mathbf{j}') = (\{i'\}, \{j'\})$
  for direction in $\{0, 1\}$ **do**
    while (i', j') is not background or is not in CBs **do**
      $i' = i' - 1$ if direction is 0 else $i' = i' + 1$
      $j' = a \cdot i' + b$
      $(\mathbf{i}', \mathbf{j}') = (\mathbf{i}' \cup \{i'\}, \mathbf{j}' \cup \{j'\})$
    end while
  end for
  if $\max\left(\hat{y}_{d,(\mathbf{i}',\mathbf{j}')}\right) > \theta_d$ **then**
    $\hat{y}_{\Delta,(\mathbf{i}',\mathbf{j}')} = 0$
  end if
end for

We will explain the algorithm with a simulated example in Fig. 4, where individual buildings cannot be extracted directly from binarized $\hat{y}_{\Delta}$ due to broken predicted CBs. Figure 4a is a simulated difference image $\hat{y}_{\Delta}$ after thresholding, where foreground pixels are building block pixels but not building edge pixels. CBs are extracted using morphological binary opening (denoted as $\circ$) and binary closing (denoted as $\bullet$), with matrix of ones $\mathbb{1}$ as structuring element.

We apply opening to binarized $\hat{y}_{\Delta}$ to remove small artifacts due to misalignment of the predicted footprint maps and the predicted edge maps, also to possibly connect isolated predicted edge pixels to background and with each other for broken lines, i.e. to enlarge the predicted edges. After binary opening, Fig. 4b shows some cases where disconnected broken edge pixels are connected (the red rectangle on the right) and where isolated edge pixels inside the building block are connected with background (the two red rectangles on the left). Binary closing is used to remove CBs by filling CB pixels with 1, connecting adjacent buildings, as is shown in Fig. 4c, where the complete building block is obtained. Then, we subtract the opening results from the closing results to get CBs, resulting in extracted CBs being in foreground, as is shown in Fig. 4d.

Individual CB is extracted by applying Connected Component Analysis (CCA) to the difference image of binary closing and opening results. For each extracted CB, we estimate a straight line with slope a and intercept b using Least Squares Estimation (LSE), based on the image coordinates of the pixels of each CB. We use bold $(\mathbf{i}, \mathbf{j})$ to refer to collection of pixel coordinates and normal (i, j) for a single pixel.

Starting from the center of each CB (starting (i', j') in Pseudo-code 1, illustrated using red dots in Fig. 4e), the pixels associated with each CB along the corresponding estimated straight lines are visited iteratively in both directions (illustrated using red arrows in Fig. 4d), indexed as 0 and 1. For each direction, the iteration stops when either background pixel (non-building-block pixels in the image after binary closing) is reached, or when a CB pixel is reached (as is shown in the enlarged rectangle in Fig. 4e). It stops also when the pixel reaches beyond the input image extent. A special case is for horizontal and near horizontal lines where the maximal difference within $\mathbf{i}$ is or is close to 0. We then only iterate in horizontal directions and change j' while keeping i' unchanged.

When iterations in both directions are stopped, all the visited pixels are denoted as $(\mathbf{i}', \mathbf{j}')$, presented in Fig. 4e as red straight lines. If the highest predicted vertex density (simulated using white dots in Fig. 4e) of pixels $(\mathbf{i}', \mathbf{j}')$ is larger than a threshold θ_d, we then split the corresponding connected component in the binarized $\hat{y}_{\Delta}$ using the prolonged straight line into two parts, i.e. two individual buildings. If all pixels $(\mathbf{i}', \mathbf{j}')$ do not surpass the threshold θ_d, the extended line is discarded and the original building block is unchanged, as is highlighted in the red rectangle in Fig. 4f. In practice, binary opening should be used in the end to deal with close CBs with small gaps, as is the case shown in the enlarged rectangle in Fig. 4e.

Finally, after processing each CB, we trace each connected component by applying CCA to get the initial polygons P_{init} for individual buildings.

In order to obtain polygons of quality close to manual delineation, e.g. with sharp corners, straight edges, redundant vertices that are not building corners should be removed from the initial polygons $\mathbf{P}_{init}$. Theoretically, this can be done by evaluating each vertex with corresponding predicted building vertex density and discarding the vertices with predicted vertex density lower than threshold. Consequently, only the vertices that are most probably building vertices are kept and form a building polygon with straight edges and sharp corners.

However, due to false negative predictions regarding building vertices, real building vertex locations could have low predicted building vertex density, resulting in discarded

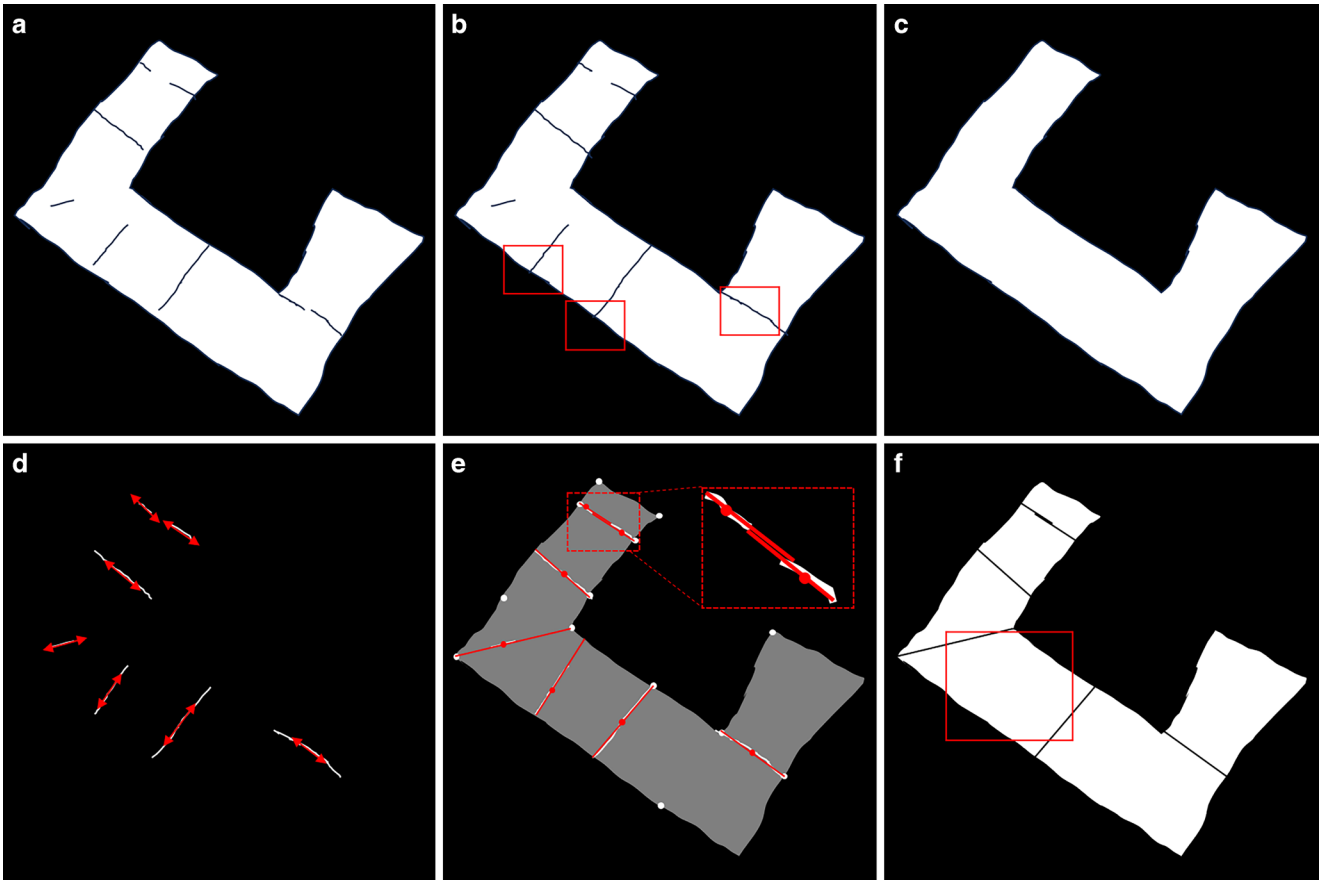

Fig. 4 Illustration of splitting building blocks. White pixels represent foreground (with value 1) and black pixels represent background (with value 0) in each image. Red rectangles in **b** show where isolated predicted edge pixels are connected with background after binary opening. In **d**, white curves represent extracted CBs after subtracting the binary opening result from the binary closing result, and red arrows show both directions of estimated straight lines for each CB. In **e**, we overlay the result after binary closing for reference in gray for better visualization; red dots show starting points for each CB (mean location of CB pixels) for iteration, red lines represent visited pixels for each CB, and white dots simulate high predicted vertex density locations. **f** shows the split buildings after subtracting the visited along line pixels, where the red rectangle shows the case where the predicted vertex density values are all smaller than threshold and thus the building is not split for this extracted CB. Note that these figures are manually drawn to illustrate the algorithm to split building blocks and do not represent real predictions. We refer to the main text for more details. **a** Difference image $\hat{y}_{\Delta}$ after thresholding, **b** Image after binary opening $\hat{y}_{\Delta,o}$, **c** Image after binary closing $\hat{y}_{\Delta,c}$, **d** Extracted CBs and estimated straight lines, **e** Extended lines consisting of visited pixels, **f** Individual buildings after splitting

real building vertices and corrupted polygons. Therefore, in practice, we use the combined predicted vertex density and predicted edge response to include more candidate vertices in thresholding.

Pseudo-code 2 elaborates the polygon adjustment algorithm. For each vertex $\mathbf{p}$ in each initial polygon $\mathbf{P}$, we inspect a surrounding circular area $\mathbf{p}'$ with a certain radius, and replace each vertex with the location of the highest combined vertex density and edge response $\mathbf{p}_{max}$. If this response is smaller than a threshold, we then discard the corresponding initial polygon vertex. Since it is likely that more than one vertices are adjusted to the same high response location, the duplicated extra vertices are removed to obtain the final predicted building polygon. This procedure is also able to remove possible shifts in the previous splitting process by adjusting the location of each vertex.

Pseudo-code 2 Polygon adjustment

$\mathbf{P}_{init} = \mathrm{CCA}\,(\hat{y}_{\Delta})$
$\hat{y}_{resp} = \hat{y}_d + \hat{y}_e$
for $\mathbf{P}$ in $\mathbf{P}_{init}$ **do**
  for $\mathbf{p}$ in $\mathbf{P}$ **do**
    $\mathbf{p}' = \mathbf{p} \pm$ search radius
    $\mathbf{p}_{max} = \arg\max\,(\hat{y}_{resp,\mathbf{p}'})$
    if $\hat{y}_{resp,\mathbf{p}_{max}} > \theta_d + \theta_{\Delta}$ **then**
      $\mathbf{p} = \mathbf{p}_{max}$
    else
      delete $\mathbf{p}$
    end if
  end for
end for

We choose to add the predicted edge response to include possibly low predicted vertex density at real building vertex locations, i.e. to reduce the impacts of false negative predictions from building vertices extraction. However, this

Table 1 Datasets studied. In this work, we used in total three datasets with varying image sizes. The CrowdAI and the UBC datasets are based on satellite images, while the Inria dataset is based on aerial images

Dataset	Image size (pixels)	Used image size (pixels)	# of training samples	# of test samples
CrowdAI (Mohanty et al. 2020)	300×300	320×320	280,741	60,317
UBC (Huang et al. 2022)	600×600	512×512	560	160
Inria (Maggiori et al. 2017)	5000×5000	512×512	14,400	14,400

has a negative impact on the final quality of the building polygons in sense of complexity, since it is more likely to include false building vertices. This is a trade-off problem and will be discussed further in the ablation study.

When deciding whether to drop or to keep the initial polygon vertices, the threshold depends on the used predicted response image $\hat{y}_{resp}$. If only predicted vertex density is considered, θ_d only, same as that in Pseudo-code 1, is used. When the predicted vertex density and the predicted building edge response are combined as $\hat{y}_{resp}$, θ_Δ is additionally added to drop edge pixels in adjustment. All hyperparameters in these two algorithms will be discussed in the following section.

4 Experiments

In this section, we first present the datasets we used in our experiments, followed by the implementation details of the proposed method and the experimental settings.

4.1 Dataset and pre-processing

An overview of the datasets that we used in this work is shown in Table 1. We conducted our experiments based on three datasets. The first dataset is the CrowdAI mapping challenge dataset (Mohanty et al. 2020), the second dataset is the UBC dataset (Huang et al. 2022), and the third dataset is the Inria aerial image dataset (Maggiori et al. 2017).

The CrowdAI dataset contains satellite images of size 300×300 pixels as RGB images, as well as annotations of building blocks. In this paper, we used the CrowdAI training dataset which contains in total 280,741 image tiles, and testing dataset[1] with 60,317 image tiles. The Ground Sampling Distance (GSD) of the CrowdAI dataset is unknown. The UBC dataset contains satellite images of size 600×600 pixels as RGB images and annotations of individual buildings with varying GSD from 0.5 m to 0.8 m. We used the UBC training dataset with 560 image tiles and testing dataset with 160 image tiles. The Inria training set contains 180 aerial images of size 5000×5000 pixels as well as building footprint masks, and the test set contains 180 RGB images, both with GSD 0.3 m. We re-tiled each large image in the Inria dataset into tiles of size 512×512 pixels, resulting in total 14,400 training samples and 14,400 test samples.

To generate the reference data as learning targets, the polygon annotations (ordered lists of building vertex coordinates for each polygon) were rasterized into binary images as building footprints, and the polygon edges were rasterized into binary building edge heatmaps, where each edge was of width 1 pixel. The polygon vertices were rasterized into binary building vertex heatmaps (for intermediate supervision) and used as input to Eq. 4 for building vertex density maps.

Since there are no reference polygon annotations provided in the Inria dataset but only binary building footprint images, we generated the binary building edge heatmaps using CCA and rasterized the contours of each connected component. Each contour was then simplified using Douglas-Peucker algorithm (Ramer 1972) with maximal distance set to 5 pixels, and the vertices after simplification were used to generate the building vertex density maps using Eq. 4. Consequently, the generated reference building vertex density map is only able to serve as pseudo-reference. Therefore, for the Inria dataset, we only consider the predicted building footprints for result analysis.

In order to make the input data better fit our network structure, e.g. avoiding padding in down-sample and up-sample paths, we resized the CrowdAI images to size 320×320 pixels and the UBC dataset to size 512×512 pixels, using bilinear interpolation. The reference annotations were also recalculated accordingly during training. We used max pooling to generate references for intermediate supervision. At inference stage, we resized the predictions to corresponding original size for evaluation.

4.2 Network implementation and training

In this work, the number of branches was set to 3, one for each geometry primitive type. As for the number of stacks of MBMs, we experimented with different stacks of 1, 2 and 3. The results are reported in the following section as ablation study.

For the whole network, we used ReLU as activation function, and did not apply dropout. In the stem encoder, we used 2D convolutional layer with stride 2 followed by max

[1] Although we use the term "testing dataset", it refers to the validation dataset in the CrowdAI dataset. Same applies to the UBC dataset.

pooling layer to reduce the spatial dimension to fourth of inputs. Inside MBM encoders, down-sampling was realized by a 2D convolutional layer with stride 2 and kernel size 3, while up-sampling in MBM decoders was implemented using bilinear interpolation. The output feature of the deepest layer was of spatial dimension 32 times smaller than the input image. Skip connection was implemented using concatenation, i.e. corresponding features from MBM encoder and MBM decoders were concatenated in feature dimension and fed to the following residual blocks. We used three residual blocks between each two layers that change spatial dimension. The number of features was set to 64 after stem encoder, and each time when spatial dimension reduced, the number of features was doubled, while it reduced to half when spatial dimension increased. PyTorch (Paszke et al. 2017) was used for network implementation.

When generating the reference vertex density image for building vertices, we chose σ in Eq. 3 as 1 to obtain a full width at half maximum window size of around 5×5 pixels, and normalized each density image such that the peak values equal to 1 using Eq. 4. In this way, the density value at location 2 pixels away from a peak center is approximately 0.5. This smoothing helps compensate possible shifts between given reference vertices and real vertices present in images, while mitigating the problem of highly imbalanced positive and negative samples.

Our model was first trained from scratch using the CrowdAI dataset, since there are abundant training samples as well as for test. Each model was trained for 165 epochs. The trained model was then fine-tuned on the UBC dataset (for 50 epochs) to study the transferability of our model from building blocks to individual buildings, considering the much smaller number of training samples available in the UBC dataset. The model variant used was with 1 stack of MBM to keep the number of parameters similar with the comparative model based on FFL (Girard et al. 2021). As for the Inria dataset, we trained our model from scratch for fair comparison with other methods.

During training, the learning rate was kept constant as 10^{-4}. We used AdamW optimizer (Loshchilov and Hutter 2018) with weight decay of 10^{-4}. For the Inria dataset, the validation set was a randomly selected hold-out subset of the training data, which was used to select the best model.

Furthermore, we trained another model based on an existing method (Girard et al. 2021) and report the corresponding evaluation results, as a comparison for the UBC dataset. The model was trained on the UBC training dataset using a pre-trained model on the CrowdAI dataset, available online[2].

4.3 Hyperparameters in enhanced building polygonization

There are several hyperparameters in the proposed enhanced building polygonization and adjustment method. In Pseudo-code 1, the threshold θ_Δ of the difference image $\hat{y}_\Delta$ controls the amount of foreground pixels in the image representing building blocks without CBs (Fig. 4a). It is safe to set this value to 0.5, interpreted as "pixels with probability larger than 50% of being building blocks but not edges are selected".

The size of the structuring element used in binary closing and binary opening depends on many aspects, e.g. the GSD of input images, the actual predictions, especially on the predicted edge maps. The size of the structuring element should be large enough to remove CBs after the binary closing operation, but not too large to remove buildings after the opening operation. We chose 5×5 empirically to achieve a balance, considering wide range of GSDs of studied datasets and the actual prediction results.

The threshold θ_d for building vertex density depends on the actual predictions in practice. A higher threshold leads to more correct splitting of building blocks, while resulting in more false negative individual buildings due to low predicted vertex density at real building vertex locations. We set this threshold to 0.2 empirically to include more split buildings.

When adjusting the initial polygon vertices in Pseudo-code 2, the search radius depends on how further away each polygon vertex is allowed to move. We found that the same window size of the structuring element in Pseudo-code 1 can be used, since the choice of this search radius is related to the possible changes of boundary pixels after binary opening and closing. With the same window size, these shifts are able to be compensated in this adjustment procedure.

The threshold for deciding whether to keep or drop the vertex in Pseudo-code 2 is kept in line with the threshold for building vertex density used in Pseudo-code 1, since they both select "high enough responses". An additional θ_Δ is added when using both predicted building vertex density and predicted building edge probability as response image, to compensate the added response from the predicted building edges.

Overall, there are in total three sets of hyperparameters: (a) threshold θ_Δ to binarize difference image $\hat{y}_\Delta$, (b) size of the structuring element used in binary closing and opening in Pseudo-code 1, which is kept the same as the search radius in Pseudo-code 2, (c) threshold θ_d for choosing high predicted vertex density values in both algorithms.

[2] https://drive.google.com/drive/folders/1poTQbpCz12ra22CsucF_hd_8dSQ1T3eT.

5 Results

In this section, the evaluation metrics we used are first introduced. Next, the quantitative evaluation results of the three datasets are presented.

5.1 Evaluation protocols

We report the evaluation of our predictions based on two sets of protocols. The first set is based on raster calculation. Under this protocol, we calculate Average Precision (AP) and Average Recall (AR) according to MS COCO (Lin et al. 2014) using final building polygons. We also calculate the Intersection over Union (IoU) using rasterized final building polygons with reference building footprint maps, which is the ratio of the overlap area and the union area of prediction w.r.t. the corresponding reference, as

$$\text{IoU} = \frac{A_{\text{predict}\cap\text{reference}}}{A_{\text{predict}\cup\text{reference}}}, \tag{7}$$

where A is the area, and predict $\cap$ reference and predict $\cup$ reference are the intersection and the union of prediction and reference respectively.

When applied to polygons, IoU is calculated using rasterized polygons in this work. Overall, AP and AR related evaluation is based on building polygons (instances) while IoU is based on binary masks.

In addition to raster-based evaluation, we evaluate our predictions also polygon-wise, since our final product is in polygonal form. For this evaluation, we calculate Ratio of vertex Numbers (RN) of the predicted polygons and the reference polygons as

$$\text{RN} = \sum_{i=1}^{N} \frac{n_{\hat{\mathbf{P}},i}}{n_{\mathbf{P},i}}, \tag{8}$$

where $\mathbf{P}$ and $\hat{\mathbf{P}}$ are the reference and the predicted polygon, n represents the number of vertices, and N is the total number of pairs of predicted polygons and reference polygons.

RN reflects reconstructed complexity of the predicted polygons. A perfectly reconstructed polygon should have RN equal to 1 w.r.t. the reference polygon.

We also include C-IoU (Zorzi et al. 2022), which is IoU with a penalty term of relative RN given as

$$\text{C-IoU}\left(\mathbf{P}_1, \mathbf{P}_2\right) = \text{IoU}\left(\mathbf{P}_1, \mathbf{P}_2\right)\left(1 - \frac{|n_{\mathbf{P}_1} - n_{\mathbf{P}_2}|}{n_{\mathbf{P}_1} + n_{\mathbf{P}_2}}\right), \tag{9}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are the input polygons for evaluation.

Additionally, to measure distances/differences between two polygons, we define a directional Polygon Difference (PD) as

$$\text{PD}\left(\mathbf{P}_1, \mathbf{P}_2\right) = \frac{1}{n_{\mathbf{P}_1}} \sum_{\mathbf{p}\in\mathbf{P}_1} \min \|\mathbf{p}, \mathbf{P}_2\|, \tag{10}$$

where $\mathbf{p}$ is the coordinate of each vertex in polygon $\mathbf{P}_1$, and $\|\mathbf{p}, \mathbf{P}_2\|$ is the Euclidean distance of vertex $\mathbf{p}$ to each vertex in polygon $\mathbf{P}_2$.

PD reflects overall polygon difference of two polygons based on vertex location. In this work, we report bidirectional PDs, which are

$$\text{PD}_{t,p} = \sum_{i=1}^{N} \text{PD}\left(\mathbf{P}, \hat{\mathbf{P}}\right)_i, \tag{11}$$

and

$$\text{PD}_{p,t} = \sum_{i=1}^{N} \text{PD}\left(\hat{\mathbf{P}}, \mathbf{P}\right)_i. \tag{12}$$

$\text{PD}_{t,p}$ measures vertex location difference of each reference vertex w.r.t. the predicted vertices, and $\text{PD}_{p,t}$ is the opposite. By inspecting PDs in both directions, we are able to evaluate prediction quality in sense of polygon difference based on vertex shifts.

In practice, for each reference polygon, we calculate IoU for each predicted polygon in the same image scene, and choose the predicted polygon with the highest IoU as the matched prediction. Each predicted polygon is only allowed to match with at most one reference polygon. RN and PD are then calculated for each reference and prediction pair. The final result is averaged over the total number of such pairs.

5.2 Quantitative evaluation on the CrowdAI dataset

We provide two sets of evaluation protocols for the CrowdAI dataset in Table 2. As discussed above, the first set is raster-based, for which we report AP, AR and IoU, and the second set is polygon-based, for which we report C-IoU (Zorzi et al. 2022), RN and bidirectional PDs. For raster-based evaluation protocols, our method achieved AP of 62.7% and AR 73.6%.

5.3 Quantitative evaluation on the UBC dataset

Table 3 reports evaluation results on the UBC dataset to address the ability of handling individual buildings of our method. The evaluation results on the UBC dataset were

Table 2 Evaluation results on the CrowdAI dataset. We present the results with the highest AP of all possible variants for each method, including our Multi-branch Model for Building Polygonization (MBM-BP). Bidirectional PDs are reported separately in ablation study. We compare our method MBM-BP to other published methods in the table, namely Mask R-CNN (He et al. 2017), PANet (Liu et al. 2018), PolyMapper (Li et al. 2019), Frame Field Learning (FFL) model (Girard et al. 2021) and PolyWorld (Zorzi et al. 2022). All units other than RN are percentage. The best and the second best values are highlighted with bold and underlined texts respectively

Method	AP ↑	AP_{50} ↑	AP_{75} ↑	AR ↑	IoU ↑	C-IoU ↑	RN ∼ 1
Mask R-CNN (He et al. 2017)	41.9	67.5	48.8	47.6	-	-	-
PANet (Liu et al. 2018)	50.7	73.9	62.6	54.4	-	-	-
PolyMapper (Li et al. 2019)	55.7	86.0	65.1	62.1	-	-	-
FFL (Girard et al. 2021)	61.3	<u>87.6</u>	**70.6**	64.9	84.1	<u>73.3</u>	<u>1.13</u>
PolyWorld (Zorzi et al. 2022)	**63.3**	**88.6**	<u>70.5</u>	**75.4**	**91.3**	**88.2**	**0.93**
MBM-BP (ours)	<u>62.7</u>	82.4	69.7	<u>73.6</u>	<u>89.4</u>	58.3	1.71

Table 3 Evaluation results on the UBC dataset. The results of our method are calculated using the adjustment based on vertex density plus edge response. All units are percentage

Method	AP ↑	AP_{50} ↑	AP_{75} ↑	AR ↑
FFL	12.1	23.9	11.0	21.5
MBM-BP	13.9	28.1	12.5	24.4

calculated based on the adjustment including edge response, since edge information plays an important role in separating individual buildings. Overall, our method achieved AP around 13.9%, with AR over 24%.

5.4 Evaluation on the Inria dataset

We report IoU for the Inria dataset which is calculated by the dataset owner[3].

The Inria dataset is more challenging than the CrowdAI dataset for its large variation of buildings and high scene complexity. For example, there are many densely built areas, which makes the prediction a difficult and challenging task, as is shown in Fig. 5. Nevertheless, our method achieved satisfying results with IoU over 71%. The complete leader-board is online available[4].

6 Discussion

In the following, we present our discussions based on the interpretations of our results and the comparisons with the comparative methods. Ablation studies regarding the number of stacks of MBMs and the enhanced building polygonization are analyzed, as well as computational complexity of the proposed method.

Fig. 5 Example of a prediction on the Inria test dataset, where densely built buildings are present

6.1 Quantitative evaluations

AP measures the overall prediction quality by considering precision and recall, which is the area under the precision-recall curve[5]. Higher AP indicates larger fraction of correct building polygons among all predictions, and that more true building polygons are retrieved from the reference polygons.

Our proposed method achieved good AP of 62.7%, with 0.6% smaller than previously reported state-of-the-art method PolyWorld (Zorzi et al. 2022), as is reported

[3] https://project.inria.fr/aerialimagelabeling/.

[4] https://project.inria.fr/aerialimagelabeling/leaderboard/.

[5] It is worth mentioning that, for original mean AP and mean AR developed for multi-class object detection (Lin et al. 2014), two criteria are considered: prediction scores of each prediction and the IoU values of each prediction and reference pair. Therefore, if we fix an IoU threshold (e.g. at 50%), we still get a precision-recall curve against different prediction score thresholds. We adapt the same metrics but set all prediction scores to 1, indicating that each building polygon is considered a positive sample regarding prediction scores. Consequently, AP and AR are evaluated based on IoU thresholds only. Same procedure was used in the cited comparative methods.

in Table 2. However, AP_{50} is much lower than PolyWorld (Zorzi et al. 2022) and FFL (Girard et al. 2021), while for predictions with higher quality, considering AP_{75}, the performance matches the two methods, with the proposed method 0.8% and 0.9% lower respectively. It could suggest that the proposed method tends to output more poor quality polygons than the comparative methods, i.e. with lower IoU w.r.t. references: when more predicted polygons of poorer quality are present, a lower AP_{50} will be observed when comparing with other methods, while AP_{75} remains similar, indicating that the performance regarding better quality predictions is alike.

AR, on the other hand, reflects the performance regarding retrieval of true building polygons from references, and is also averaged over different IoU thresholds (from 50% to 95% with 5% intervals in this work). Overall, our proposed method is 1.8% lower in AR than PolyWorld (Zorzi et al. 2022) while it outperforms the other cited comparative methods in Table 2. It is the same for overall IoU, with 0.9% lower than PolyWorld (Zorzi et al. 2022) and 5.3% higher than FFL (Girard et al. 2021). The AR and IoU results suggest that the proposed method is able to retrieve most reference building polygons correctly with high quality regarding IoU criterion.

While PolyWorld (Zorzi et al. 2022) seems to be the best model among all mentioned methods in Table 2, it is reported to be less robust. The polygonization procedure in PolyWorld (Zorzi et al. 2022) is done through a learned adjacency matrix and a second stage based on an iterative assignment optimization algorithm. It is pointed out in the original paper that false connections between extracted vertices can be generated, which leads to collapsed incorrect polygons, and this is not controlled in any way. We suspect that this is why the AP_{75} is slightly lower than that of FFL (Girard et al. 2021): when a building polygon collapses from a rectangle into, for example, a triangle, it might be considered as a true positive sample at IoU threshold of 50%, but as a false negative sample at 75%. From a practical point of view, these incorrect polygons are not desired. In this sense, our proposed method achieved matching results in a more controlled manner which produces building polygons that do not differ as drastically from reality.

When comparing with the FFL based method (Girard et al. 2021), we get better results except for AP_{50} and AP_{75}. We believe that this is partly because of smaller objects. It is more difficult to extract smaller objects than larger ones, especially for segmentation based methods. In the FFL based method, this issue is addressed by the active skeleton model, where the whole image scene is divided into polygonized parcels, including background, followed by polygon filtering. Theoretically, no true building polygons are filtered out in this way. In contrast, our proposed method generates initial polygons based on the predicted footprint maps solely. If the predicted building footprint response is too low, it is highly possible that this building would be missing in our final predicted polygons.

For polygon-based evaluation protocols, our method has worse performance when comparing with the two competitive methods. This is due to the reconstructed number of vertices in the predicted polygons. We will address this issue in the following with ablation study.

The UBC dataset is more complicated than the CrowdAI dataset for being individual building oriented and its complicated scenes with high-rising buildings. This is the main reason why the evaluation results are worse for the UBC dataset comparing with the CrowdAI dataset for both methods shown in Table 3. It suggests that the CBs were not satisfyingly extracted, especially when the buildings are densely distributed. Another reason could be the limited amount of data, which leads to fast over-fitting in training.

Specifically, our proposed method relies highly on the building vertices, especially in the building polygonization procedure. It means that our network requires correct reference vertex maps as supervision to extract real building vertices. However, the annotations in the UBC dataset do not guarantee such correctness on building vertices (e.g. Fig. 6). This could lead to wrongly extracted building vertices, as well as wrongly split buildings.

The individual building polygons depend on the network predictions and on the proposed enhanced building polygonization and adjustment algorithm. Poorer performance for individual buildings suggests that the extracted building edges (including CBs) must have higher quality, e.g. less disconnected and isolated CBs. More accurate extraction of building vertices, especially for the shared non-corner vertices on the building block edges, must be achieved for higher quality of individual building polygons. Nevertheless, our proposed method outperformed FFL (Girard

Fig. 6 Example of an annotation in the UBC dataset. For this building, not only building corners are recorded but also vertices along building edges. This leads to false building vertices, which influences our model

Table 4 Reported IoU based on the Inria dataset. "Stack 1", "stack 2" and "stack 3" are the model variants of different numbers of stacked MBMs, i.e. stack of 1, 2 and 3 MBMs respectively. Full leader-board is available online (see main text). We report specifically the result of the comparative method based on FFL (Girard et al. 2021)

Model	stack 1	stack 2	stack 3	FFL
IoU (%)	71.3	70.4	71.0	74.0

et al. 2021) regarding the four evaluation criteria reported in Table 3.

Due to the nature of the assignment optimization algorithm in PolyWorld (Zorzi et al. 2022) where each vertex is only allowed to be matched with at most one vertex, it does not work for individual buildings, where each vertex could be shared with more than one building. In this sense, the FFL based method (Girard et al. 2021) and our proposed method are more flexible.

For the Inria dataset, the proposed method did not outperform FFL (Girard et al. 2021), as is shown in Table 4. This can be explained by the fact that the pseudo-references of building vertices are detrimental to our network training. When wrong references are fed to the building vertices branch, wrong information is then further propagated to the other two branches, leading to performance drop. With increased number of stacks, this harmful information is possibly to become more dominant, resulting in a decrease of IoU with 3 stacks of MBMs when comparing with the variant with only 1 MBM. It further confirms the dependence of high quality reference data for the network training in our proposed method. Nevertheless, for the task of building footprint extraction, the proposed method achieved adequate results with the highest overall IoU being 71.3%.

6.2 Qualitative evaluations

The first row in Fig. 7 shows an example of raster predictions from the CrowdAI dataset. It is observed that most of the building vertices are correctly extracted. The predicted building edges are straight and have sharp corners at intersections, despite some false negative pixels where line pixels are disconnected. The predicted building footprints are more irregular, which proves that the enhanced building polygonization and adjustment procedure is necessary to obtain higher quality polygons.

The second row in Fig. 7 shows an example of the three predictions from the UBC dataset. In the predicted vertex density map Fig. 7f, the influence of the non-building corners in reference (Fig. 6) is clearly observed, where peak values appear very often on building edges. Nevertheless, peaks at true building corners and at shared CB vertices (highlighted in red rectangles) are observed, which proves that the proposed method is also able to handle individual buildings, despite of the polluted training data.

Figure 8 shows three examples from the CrowdAI dataset for qualitative evaluation. We visualize two possible polygon adjustment methods in our enhanced building polygonization procedure: the first method adjusts each vertex to

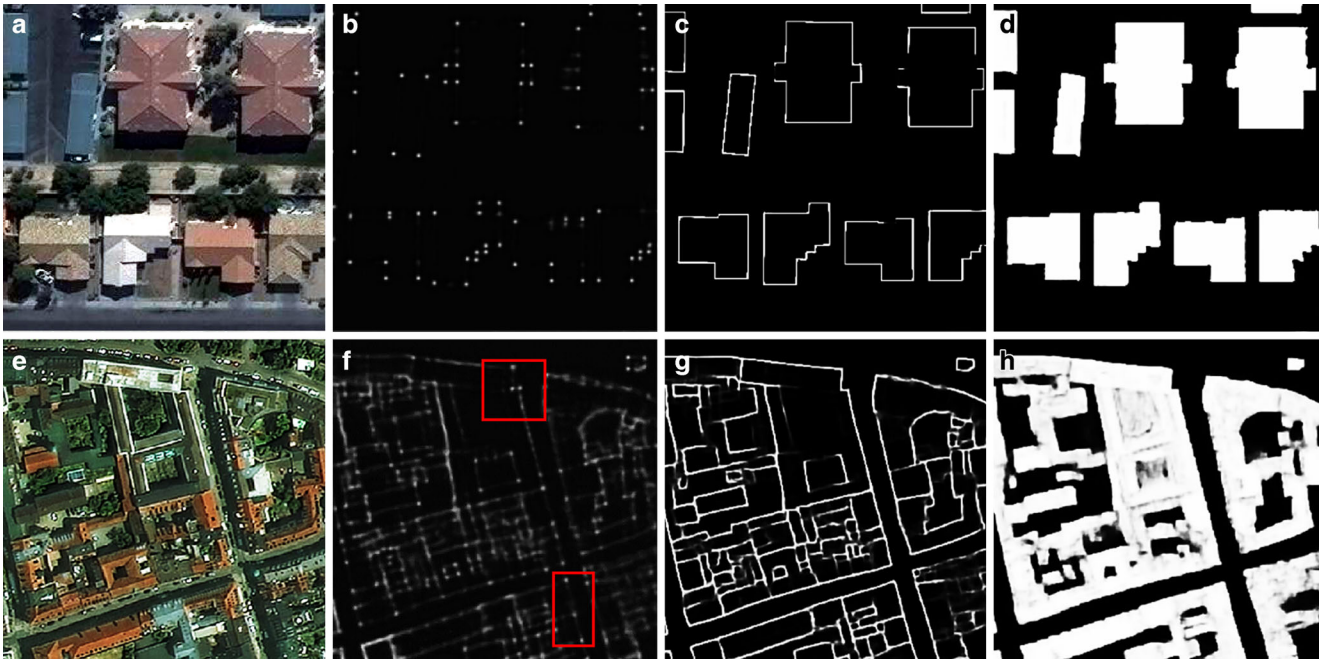

Fig. 7 Two examples of the final raster predictions from the CrowdAI (first row) and the UBC (second row) dataset. **a**, **e** RGB input image, **b**, **f** predicted building vertex density, **c**, **g** predicted building edges, **d**, **h** predicted building footprints. All predictions are visualized as gray images. Areas of interest are highlighted in red rectangles, where peaks of vertex density appear at true building vertex locations

Fig. 8 Examples of the CrowdAI dataset results. Each column from left to right: **a** RGB image, **b** reference annotation, **c** adjustment based on the vertex density, and **d** adjustment based on the combined vertex density and edge response. Green rectangles show sharp building corners from the adjustment based on the vertex density, while red rectangle shows where this method fails and the polygons collapse. Blue rectangle shows robustness of the adjustment based on the combined vertex density and edge response, which is able to reconstruct the building shape correctly. Color enhancement is applied to each image for better visualization

the highest predicted vertex density, and the second method adjusts each vertex to the highest combined predicted vertex density and edge response. The results of these two methods are shown in the third and the fourth column respectively.

Overall, both methods are capable of producing fairly satisfying polygons. However, the first method, which is based on the predicted vertex density only, produces better details and sharper corners, shown in green rectangles. Without addition of the predicted edge response, non-corner vertices are suppressed, and polygons appear more regularized with sharp corners, without zigzag shapes introduced by additional false non-corner building vertices. That being said, this method highly relies on the quality of the predicted vertex density, and is more sensitive in the choice of the adjustment threshold θ_d. If the corresponding predicted vertex density is too low to surpass θ_d, the whole polygon will collapse, leading to wrong building edges, shown in red rectangles.

In contrast, the second method based on the combined predicted vertex density and edge response seems more robust, but with loss in the overall quality of polygons. For example, in the last example within the blue rectangle in Fig. 8, the overall shape of the building is detected with the second method, while the polygon collapses into the building itself using the first method. In all, the choice of the adjustment methods is a trade-off of robustness and the final quality of polygons.

Figure 9 shows three examples from the UBC dataset. Since the annotations in the UBC dataset for building corners are not guaranteed to be correct, we only consider the adjustment strategy based on the combined predicted vertex density and edge response. We compare our results with the method based on FFL (Girard et al. 2021).

In general, our model works better and outputs meaningful building polygons, with several example areas highlighted with yellow rectangles in Fig. 9 where the FFL

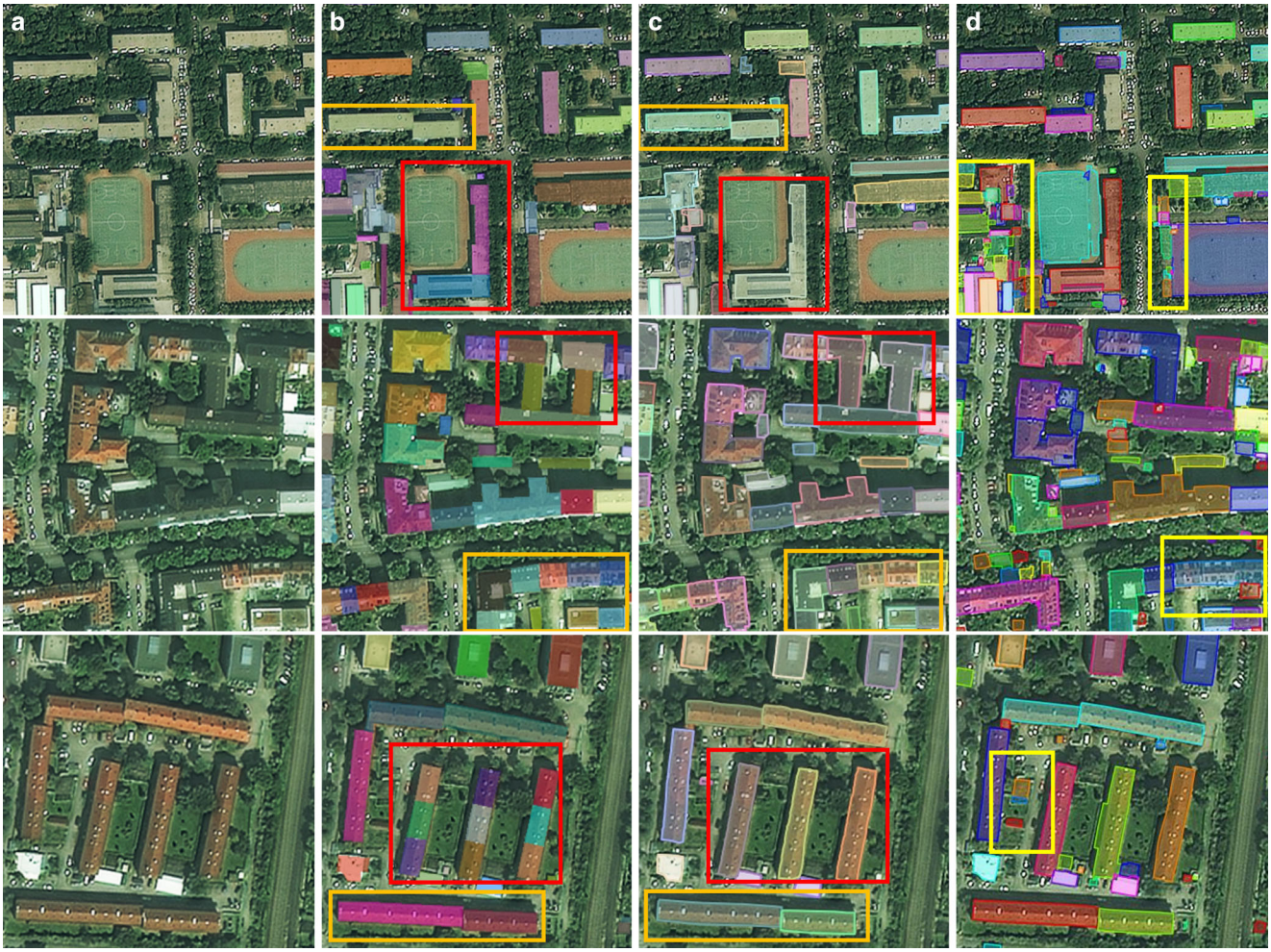

Fig. 9 Examples of the UBC dataset results. Each column from left to right: **a** RGB image, **b** reference annotation, **c** results of our model, and **d** results of the FFL based method. Orange rectangles in columns **b** and **c** show that our model works fairly well in some areas where individual buildings are nicely separated, while red rectangles show examples where our model fails to separate individual buildings. Generally speaking, our method outperforms the method based on FFL by visual inspection, exemplified with areas highlighted in yellow rectangles in column **d**

based method (Girard et al. 2021) failed, e.g. false positive predictions, missing individual buildings. When transferring from building blocks to individual buildings, our model is still able to extract individual buildings, even when they are directly connected with each other, shown in orange rectangles. However, there are cases where our model fails to separate different individual buildings, shown in red rectangles. We suspect that it is mainly due to our post-processing for splitting individual buildings. It is possible that not all CBs are correctly extracted and prolonged by Pseudocode 1, leading to merged individual buildings.

Moreover, the final polygons tend to have more zigzagging edges than those from the CrowdAI dataset, since we lost the information of building vertices in the UBC dataset. This is considered a limitation of our model that it has specific requirements on the reference annotations.

Figure 10 presents some examples from the Inria test dataset results. Since for this dataset, we only have pseudo-references for building edges and building vertices, we only show the predicted footprints in raster format. Generally speaking, the proposed method was able to extract most of the buildings correctly. It proves that the proposed method performs well on very high resolution aerial images (0.3 m for the Inria dataset) and in different scenarios.

6.3 Ablation study based on the CrowdAI dataset

With focus on polygon-based evaluation, we present quantitative evaluation results in Table 5 for ablation study based on the CrowdAI dataset. We compare different numbers of stacks of MBMs, as well as different polygonization base, i.e. adjustment (a) based on the predicted vertex density maps only and (b) based on the combined predicted vertex density maps and predicted edge maps.

The upper part of Table 5 presents the evaluation results where we replace each vertex in the initial polygons

Fig. 10 Examples of the Inria test dataset results. RGB images are overlaid with the corresponding predicted building footprints

with the location with the highest predicted vertex density. Although with number of stacks of MBMs equal to 3 the highest AP and AR are observed, the differences of all evaluation results are similar for each variant. Additionally, RN values are roughly 0.25 smaller than the optimal value 1, i.e. all polygons are oversimplified. It means that not all building vertices were correctly extracted, possibly due to shadows, vegetation obstruction, etc. Furthermore, the predicted polygons also highly depend on the thresholding of the vertex adjustment θ_d. A simple global threshold might not lead to optimally adjusted polygons with given predictions, resulting in oversimplified polygons.

The lower part of Table 5 shows the evaluation results where each vertex in the initial polygons is replaced by the highest predicted vertex density plus predicted edge response. This method shows higher performance concerning raster-based evaluation results, but is not necessarily better regarding polygonal evaluation protocols. In general, the RNs differ largely from the optimal value of 1, and $PD_{p,t}$ is also very high for each variant. This is intuitively understandable, since vertices that are not building corners are also included in the predicted edge maps, and these extra vertices are hard to be filtered out simply by thresholding. However, lower $PD_{t,p}$ is observed. This suggests that even for building vertices extraction, line segments also play an important role. Intersection of line segments helps identify true building vertices, but brings risks of introducing more false non-corner vertices.

Regarding the comparison between the previous feature fusion design proposed by Batra et al. (2019) ("stack 1*" in Table 5), our proposed method achieved higher AP for vertex density based and combined vertex density and edge response based adjustment. Interestingly, for the adjustment based on combined vertex density and edge response, it seems that "stack 1*" extracted polygon vertices very close to reference building vertices: RN value is very close to 1, and $PD_{p,t}$ is much smaller than the other model variants. We believe that this is caused by the randomness in multi-task learning.

Overall, the network targets at different tasks in different branches. During training, it could happen that one branch is more dominant than the others. For example, although all model variants achieved similar results in extracting building vertices (similar vertex density based evaluation

Table 5 Ablation study. We tested variants of our method based on number of stacks of MBMs and different polygonization base at the second stage. The upper part of the table is adjusting each initial polygon based on the predicted vertex density. The lower part refers to the adjustment based on the combined predicted vertex density and edge response. "Stack 1*" refers to the original feature fusion proposed by Batra et al. (2019) while the rest of the model variants are named consistently as in Table 4, indicating different numbers of stacks of MBMs. The hyperparameters in the enhanced building polygonization step were kept constant for each variant. For the upper part and the lower part of the table, the best values are highlighted in bold respectively

Vertex density based						
Variant	**AP** (%) ↑	**AR** (%) ↑	**RN** − 1	**C-IoU** (%) ↑	**PD**$_{t,p}$ (pixels) ↓	**PD**$_{p,t}$ (pixels) ↓
stack 1*	58.1	70.1	0.78	68.1	**4.26**	**2.06**
stack 1	59.2	70.0	**0.79**	67.8	4.32	2.43
stack 2	57.9	68.6	0.77	66.2	4.46	2.43
stack 3	**60.8**	**72.3**	0.75	**68.2**	4.35	2.11
Combined vertex density and edge response based						
Variant	**AP** (%) ↑	**AR** (%) ↑	**RN** − 1	**C-IoU** (%) ↑	**PD**$_{t,p}$ (pixels) ↓	**PD**$_{p,t}$ (pixels) ↓
stack 1*	56.9	69.1	**1.08**	**65.1**	3.65	**4.92**
stack 1	59.9	71.0	1.61	58.8	3.17	7.06
stack 2	58.5	69.5	1.43	60.0	3.61	5.50
stack 3	**62.7**	**73.6**	1.71	58.3	**2.91**	7.22

Table 6 Model inference time in seconds based on the CrowdAI dataset. "Inference" refers to the average network inference time for a single image. "Polygonization" refers to the average processing time of the enhanced building polygonization and adjustment. Each input image size was 320×320 pixels. Test device was NVIDIA GeForce RTX 2080 with 11,019 MB memory

Variant	Inference	Polygonization
stack 1	0.55 s	0.80 s
stack 2	0.58 s	0.73 s
stack 3	0.56 s	0.62 s

Table 7 Model size and FLOPs for comparison. Generally speaking, less parameters and smaller FLOPs indicate less model complexity. "M" and "G" denote million and gillion respectively. Each input RGB image size was 320×320 pixels

Model	# of parameters ↓	FLOPs ↓
FFL	76.69 M	79.56 G
PolyWorld	39.44 M	181.23 G
MBM-BP, stack 1	76.99 M	88.73 G
MBM-BP, stack 2	114.93 M	108.02 G
MBM-BP, stack 3	152.87 M	127.30 G

results), "stack 1*" was able to extract the building edges better than the rest of the variants (better polygon quality with added edge response), but with poorer building extraction performance (lower AP and AR). This is also why the variant with 2 stacks of MBMs achieved slightly lower AP and AR but higher polygon quality than the variant with 1 stack of MBM. This could suggest that weighting different branches might lead to better performance, which we include in our future outlooks. Nevertheless, for the task of building extraction, our proposed feature enhancement is able to achieve better building extraction results.

The polygon-wise evaluation criteria penalize the mismatch of polygon complexity between the prediction and the reference concerning RNs. When this mismatch is large, the C-IoU decreases accordingly. Due to high RNs presented in Table 5, our proposed method shows much lower C-IoU in Table 2 than PolyWorld (Zorzi et al. 2022) and FFL (Girard et al. 2021). This further confirms that, despite of its simplicity, the proposed enhanced building polygonization and adjustment must be improved to extract more true building vertices and reduce the number of non-corner vertices.

Table 6 reports network inference time and processing time of the enhanced building polygonization and adjustment. Despite of larger amount of parameters with more stacks of MBMs, the network inference time is similar for all the three variants. Differences are more obvious for the second stage for building polygonization: with increase in number of stacks of MBMs, decrease of processing time is observed. This could be explained by the fact that, with number of stacks equal to 3, the predictions of geometry primitives are more accurate, with less noise that need to be filtered out. Thus, it is easier for the enhanced building polygonization module to obtain the optimal polygons with given predictions.

Other than the inference time which depends highly on computation environments, we report number of parameters and Floating Point Operations (FLOPs) to better evaluate model complexity, as is shown in Table 7.

With three parallel branches, our models have more learnable parameters, with the smallest variant having 76.99 million parameters while PolyWorld (Zorzi et al. 2022) has 39.44 million parameters and the FFL model (Girard et al. 2021) has 76.69 million parameters. It means that our model requires more memory in practice.

Despite of that, our models have lower computational complexity, with the largest FLOPs being 127.30 gillion while PolyWorld (Zorzi et al. 2022) has over 180 gillion. Although the FFL based model (Girard et al. 2021) has the least model complexity in sense of number of parameters and FLOPs, our model with 1 stack achieved matching results on the CrowdAI dataset, with AP being 59.7% and 61.3%, and AR being 71.4% and 64.9% for our proposed method and the FFL model (Girard et al. 2021) respectively. Therefore, we can conclude that our method achieved matching performance of previous state-of-the-art CNN models with less model complexity.

It is worth pointing out that, due to limitations of resources (with batch size of 2 with input RGB images of size 320×320 pixels, the model variant with 3 stacks of MBMs occupies roughly 10,120 MB GPU memory during training), we were not able to further increase the number of stacks. We add this into our future outlooks for further inspection.

6.4 Distributions of evaluation criteria

Figure 11 shows the histograms of RN, $\mathrm{PD}_{t,p}$ and $\mathrm{PD}_{p,t}$ for each model variant. For RN, stack of 2 achieved the best performance, with majority of samples closer to the optimal value of 1. Stack of 3 achieved better results in sense of $\mathrm{PD}_{t,p}$ and $\mathrm{PD}_{p,t}$: it has more samples with bidirectional PDs equal and closer to 0. In general, based on the evaluation histograms, we can further confirm the conclusions we had from the averaged evaluation results.

7 Conclusion

In this paper, we propose a two-stage method to extract building polygons from remote sensing images. The first stage adapts a CNN with stacked MBMs to extract the three geometry primitives, which are the essential parts of each building polygon. The MBMs provide learned in-

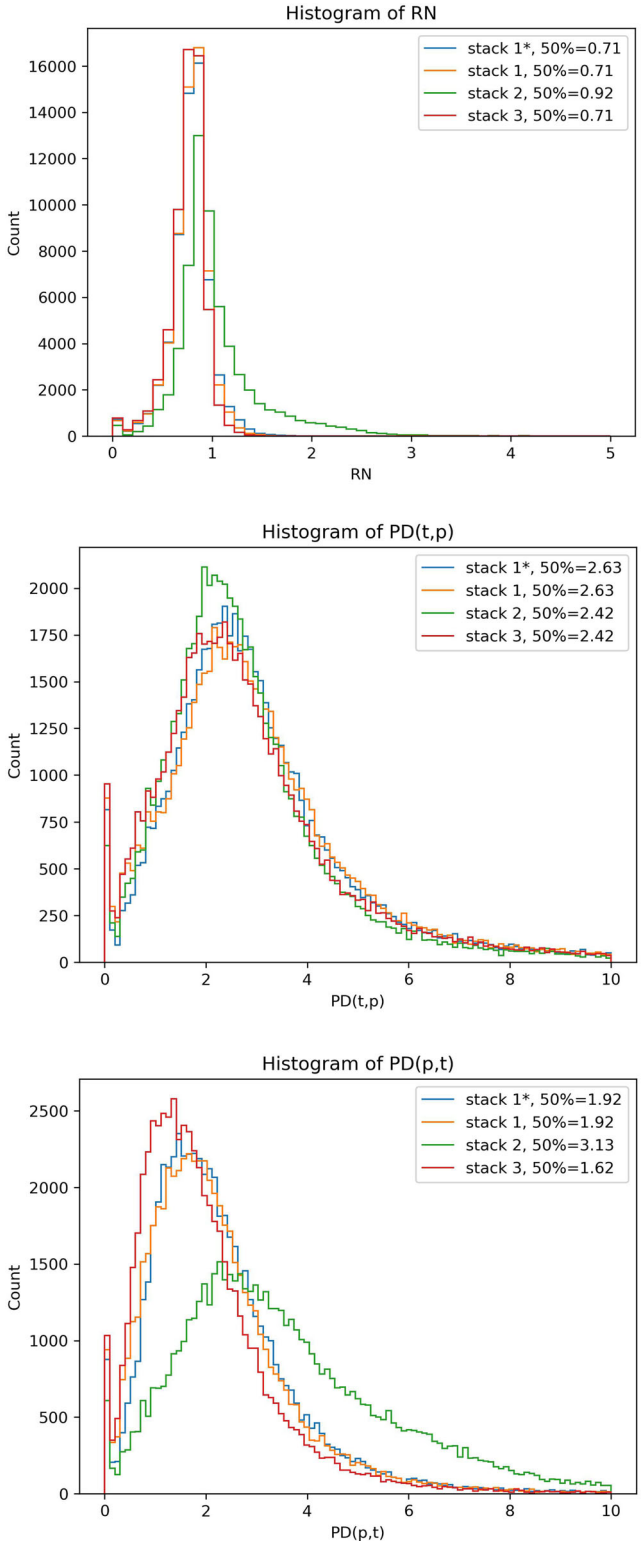

Fig. 11 Histograms of polygonal evaluation protocols. The distributions of different measures help better evaluate and compare different variants. "50%=" indicates the measured value when cumulative counts reach half of the total number of samples. The overall conclusions match the conclusions based on the averaged quantitative results. We report here only the histograms for the adjustment based on the vertex density for its better polygonization performance

teraction and information fusion across different geometry primitive branches, which improves the final predictions. At the second stage, we propose an efficient building polygonization algorithm to reconstruct building polygons based on the three predicted geometry primitives. Our algorithm produces visually pleasing building polygons with straight building edges as well as sharp corners. Moreover, our algorithm is capable of handling both building blocks and individual buildings.

Experiments show that our designed multi-branch CNN is well suited for building extraction using remote sensing images. With less computational complexity, our proposed method achieved competitive results comparing with previous state-of-the-art CNN models.

However, there are still limitations in our work which could be the reasons why the proposed method did not outperform the previous state-of-the-art method proposed by Zorzi et al. (2022). Firstly, our method depends on well-annotated reference labels, especially for the second stage. Secondly, the quality of the split individual buildings should be improved, and further studies on non-global thresholding should be considered. Lastly, number of stacks equal to 3 is not necessarily a global optimum and a balance of different branches need further study.

Funding This research received no external funding.

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and material Restrictions apply to the availability of these data. CrowdAI dataset was obtained from https://www.aicrowd.com/challenges/mapping-challenge. UBC dataset was obtained from https://github.com/AICyberTeam/UBC-dataset. Inria dataset was obtained from https://project.inria.fr/aerialimagelabeling. Datasets were used with the permission of the corresponding data distributors.

Code availability Due to institute restrictions, codes are not allowed to be distributed.

Conflicts of interest/Competing interests The authors declare no conflict of interest.

References

Alidoost F, Arefi H, Hahn M (2020) Y-shaped convolutional neural network for 3d roof elements extraction to reconstruct building models from a single aerial image. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences V-2-2020:321–328. https://doi.org/10.5194/isprs-annals-V-2-2020-321-2020

Bahmanyar R, Vig E, Reinartz P (2019) Mrcnet: Crowd counting and density map estimation in aerial and ground imagery. BMVC's Workshop on Object Detection and Recognition for Security Screenin (BMVC-ODRSS), pp 1–12

Batra A, Singh S, Pang G, Basu S, Jawahar C, Paluri M (2019) Improved road connectivity by joint learning of orientation and segmentation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 10377–10385. https://doi.org/10.1109/CVPR.2019.01063

Castrejón L, Kundu K, Urtasun R, Fidler S (2017) Annotating object instances with a polygon-rnn. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4485–4493. https://doi.org/10.1109/CVPR.2017.477

Drozdzal M, Vorontsov E, Chartrand G, Kadoury S, Pal C (2016) The importance of skip connections in biomedical image segmentation. In: Carneiro G, Mateus D, Peter L, Bradley A, Tavares JMRS, Belagiannis V, Papa JP, Nascimento JC, Loog M, Lu Z, Cardoso JS, Cornebise J (eds) Deep Learning and Data Labeling for Medical Applications. Springer International Publishing, Cham, pp 179–187

Girard N, Tarabalka Y (2018) End-to-end learning of polygons for remote sensing image classification. In: IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, pp 2083–2086. https://doi.org/10.1109/IGARSS.2018.8518116

Girard N, Smirnov D, Solomon J, Tarabalka Y (2021) Polygonal building extraction by frame field learning. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 5887–5896. https://doi.org/10.1109/CVPR46437.2021.00583

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In, vol 2016. IEEE, Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp 770–778

He K, Gkioxari G, Dollar P, Girshick R (2017) Mask r-cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp 2980–2988. https://doi.org/10.1109/ICCV.2017.322

Hu Y, Wang Z, Huang Z, Liu Y (2023) Polybuilding: Polygon transformer for building extraction. ISPRS Journal of Photogrammetry and Remote Sensing 199:15–27. https://doi.org/10.1016/j.isprsjprs.2023.03.021

Huang X, Ren L, Liu C, Wang Y, Yu H, Schmitt M, Hänsch R, Sun X, Huang H, Mayer H (2022) Urban building classification (ubc) – a dataset for individual building detection and classification from satellite imagery. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp 1412–1420. https://doi.org/10.1109/CVPRW56347.2022.00147

Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations

Li Z, Wegner JD, Lucchi A (2019) Topological map extraction from overhead images. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp 1715–1724 https://doi.org/10.1109/ICCV.2019.00180

Li Z, Xin Q, Sun Y, Cao M (2021) A deep learning-based framework for automated extraction of building footprint polygons from very high-resolution aerial imagery. Remote Sensing 13(18):3630. https://doi.org/10.3390/rs13183630

Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer Vision. ECCV, vol 2014. Springer, Cham, pp 740–755

Lin TY, Goyal P, Girshick R, He K, Dollár P (2020) Focal loss for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 42(2):318–327. https://doi.org/10.1109/TPAMI.2018.2858826

Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: 2018. IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 8759–8768. https://doi.org/10.1109/CVPR.2018.00913

Liu W, Yang M, Xie M, Guo Z, Li E, Zhang L, Pei T, Wang D (2019) Accurate building extraction from fused dsm and uav images using a chain fully convolutional neural network. Remote Sensing 11(24):2912. https://doi.org/10.3390/rs11242912

Loshchilov I, Hutter F (2018) Decoupled weight decay regularization. In: International Conference on Learning Representations

Maggiori E, Tarabalka Y, Charpiat G, Alliez P (2017) Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In: 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp 3226–3229. https://doi.org/10.1109/IGARSS.2017.8127684

Mahmud J, Price T, Bapat A, Frahm JM (2020) Boundary-aware 3d building reconstruction from a single overhead image. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 438–448. https://doi.org/10.1109/CVPR42600.2020.00052

Mohanty SP, Czakon J, Kaczmarek KA, Pyskir A, Tarasiewicz P, Kunwar S, Rohrbach J, Luo D, Prasad M, Fleer S, Göpfert JP, Tandon A, Mollard G, Rayaprolu N, Salathe M, Schilling M (2020) Deep learning for understanding satellite imagery: An experimental survey. Frontiers in Artificial Intelligence 3. https://doi.org/10.3389/frai.2020.534696

Newell A, Yang K, Deng J (2016) Stacked hourglass networks for human pose estimation. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer Vision– ECCV 2016. Springer International Publishing, Cham, pp 483–499

Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in pytorch

Qian Y, Zhang H, Furukawa Y (2021) Roof-gan: Learning to generate roof geometry and relations for residential houses. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 2795–2804 https://doi.org/10.1109/CVPR46437.2021.00282

Ramer U (1972) An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing 1(3):244–256. https://doi.org/10.1016/S0146-664X(72)80017-0

Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on medical image computing and computer-assisted intervention. Springer, pp 234–241

Schuegraf P, Schnell J, Henry C, Bittner K (2022) Building section instance segmentation with combined classical and deep learning methods. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences V-2-2022:407–414. https://doi.org/10.5194/isprs-annals-V-2-2022-407-2022

Sun W, Wang R (2018) Fully convolutional networks for semantic segmentation of very high resolution remotely sensed images combined with dsm. IEEE Geoscience and Remote Sensing Letters 15(3):474–478. https://doi.org/10.1109/LGRS.2018.2795531

Wang K, Frahm JM (2017) Single view parametric building reconstruction from satellite imagery. In: 2017 International Conference on 3D Vision (3DV), IEEE, pp 603–611

Wang Y, Zorzi S, Bittner K (2021) Machine-learned 3d building vectorization from satellite imagery. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),

pp 1072–1081. https://doi.org/10.1109/CVPRW53098.2021.00118

Wu Y, Xu L, Wang L, Chen Q, Chen Y, Clausi DA (2023) Multi-task edge detection for building vectorization from aerial images. IEEE Geoscience and Remote Sensing Letters 20:1–5. https://doi.org/10.1109/LGRS.2023.324413

Xie S, Tu Z (2015) Holistically-nested edge detection. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp 1395–1403. https://doi.org/10.1109/ICCV.2015.164

Xu Y, Schuegraf P, Bittner K (2023) Vertex aided building polygonization from satellite imagery applying deep learning. In: 2023 Joint Urban Remote Sensing Event (JURSE), pp 1–4. https://doi.org/10.1109/JURSE57346.2023.10144146

Zhao K, Kang J, Jung J, Sohn G (2018) Building extraction from satellite images using mask r-cnn with building boundary regularization. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp 242–2424. https://doi.org/10.1109/CVPRW.2018.00045

Zhao K, Kamran M, Sohn G (2020) Boundary regularized building footprint extraction from satellite images using deep neural networks. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences V-2-2020:617–624. https://doi.org/10.5194/isprs-annals-V-2-2020-617-2020

Zhao W, Persello C, Stein A (2022) Extracting planar roof structures from very high resolution images using graph neural networks. ISPRS Journal of Photogrammetry and Remote Sensing 187:34–45. https://doi.org/10.1016/j.isprsjprs.2022.02.022

Zorzi S, Bittner K, Fraundorfer F (2021) Machine-learned regularization and polygonization of building segmentation masks. 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, pp 3098–3105

Zorzi S, Bazrafkan S, Habenschuss S, Fraundorfer F (2022) Polyworld: Polygonal building extraction with graph neural networks in satellite images. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 1938–1947. https://doi.org/10.1109/CVPR52688.2022.00189