*remote sensing*

*Article*

# Building Extraction from Airborne Multi-Spectral LiDAR Point Clouds Based on Graph Geometric Moments Convolutional Neural Networks

**Dilong Li [1,2], Xin Shen [1,2,*], Yongtao Yu [3], Haiyan Guan [4], Jonathan Li [5], Guo Zhang [1,2] and Deren Li [1,2]**

1 State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China; scholar.dll@whu.edu.cn (D.L); guozhang@whu.edu.cn (G.Z.); drli@whu.edu.cn (D.L.)
2 Collaborative Innovation Center of Geospatial Technology, Wuhan University, Wuhan 430079, China
3 Faculty of Computer and Software Engineering, Huaiyin Institute of Technology, Huaian 223003, China; allennessy@hyit.edu.cn
4 School of Remote Sensing and Geomatics Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China; guanhy.nj@nuist.edu.cn
5 Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada; junli@uwaterloo.ca
* Correspondence: xinshen@whu.edu.cn; Tel.: +86-27-687-79-098

Received: 5 August 2020; Accepted: 25 September 2020; Published: 29 September 2020

**Abstract:** Building extraction has attracted much attentions for decades as a prerequisite for many applications and is still a challenging topic in the field of photogrammetry and remote sensing. Due to the lack of spectral information, massive data processing, and approach universality, building extraction from point clouds is still a thorny and challenging problem. In this paper, a novel deep-learning-based framework is proposed for building extraction from point cloud data. Specifically, first, a sample generation method is proposed to split the raw preprocessed multi-spectral light detection and ranging (LiDAR) data into numerous samples, which are directly fed into convolutional neural networks and completely cover the original inputs. Then, a graph geometric moments (GGM) convolution is proposed to encode the local geometric structure of point sets. In addition, a hierarchical architecture equipped with GGM convolution, called GGM convolutional neural networks, is proposed to train and recognize building points. Finally, the test scenes with varying sizes can be fed into the framework and obtain a point-wise extraction result. We evaluate the proposed framework and methods on the airborne multi-spectral LiDAR point clouds collected by an Optech Titan system. Compared with previous state-of-the-art networks, which are designed for point cloud segmentation, our method achieves the best performance with a correctness of 95.1%, a completeness of 93.7%, an *F-measure* of 94.4%, and an intersection over union (*IoU*) of 89.5% on two test areas. The experimental results confirm the effectiveness and efficiency of the proposed framework and methods.

**Keywords:** building extraction; airborne multi-spectral LiDAR point clouds; graph geometric moments; convolutional neural networks

## 1. Introduction

Building extraction from remotely sensed data is a prerequisite for many applications, such as three-dimensional (3D) building modeling, city planning, disaster assessment, and updating of digital maps and GIS databases [1–5]. Airborne light detection and ranging (LiDAR) data have been widely used for building extraction because of their high accuracy, large area coverage, fast acquisition, and additional information. Due to the lack of comprehensive spectral information of LiDAR data, many studies integrated LiDAR data with high spatial resolution multi-spectral images to improve the performance of building extraction [6,7]. They try to fuse the two different data sources to compensate for their weaknesses. However, how to accurately register different data sources to the same spatial coordinate system is still an open problem [8].

With the development of sensor technology, many institutes and companies have successfully developed the prototypes of multi-spectral LiDAR systems. For example, Teledyne Optech's Titan, the first commercial multi-spectral LiDAR system, was released in Canada in December 2014. Multi-spectral LiDAR data provide more comprehensive and consistent spectral information without data fusion, which has obvious advantages for building extraction tasks.

At the approach level, although there are recent advances in LiDAR data processing, several challenges are still to be resolved, especially in the areas of massive data processing, approach universality, and processing automation. Traditionally, classical machine learning methods are still considered as a mainstream tool in this field. The paradigmatic architectures first convert the raw point clouds into other forms to extract various features, which is usually called "feature representation." Then, to map the features into desired outputs, the architectures select and design a serial of classifiers to map the features into the desired outputs. Typical techniques include support vector machines (SVMs) [9], conditional Markov random fields [10], region-growing [11], k-means [12], and graph cut algorithms [13]. However, the extraction performance of these methods is highly affected by the parameters and adopted features, which are usually content and/or application-dependent [14].

In recent years, the success of deep convolutional neural networks (CNNs) for image processing has motivated the data-driven approaches to extract buildings from airborne LiDAR data. In current studies, CNNs were applied to the existing architectures [15,16] or simply served as a powerful classifier [14]. Nevertheless, due to the unstructured properties of point clouds, these CNN-based methods had to convert the raw point clouds into other data forms or the chosen feature representations from the raw point clouds, which still did not completely solve the drawbacks of the traditional data-driven methods and did not make full use of the inference ability of CNNs. The key challenges of introducing deep learning methods into building extraction from airborne LiDAR data are still to be resolved, not to mention building extraction from multi-spectral LiDAR data.

To address these issues, in this paper, we propose a novel deep learning-based framework for building extraction from multi-spectral point cloud data. With this framework, the multi-spectral LiDAR data are directly used for building extraction without transforming them into other data forms, e.g., the multi-view projected images, digital surface model (DSM), or digital terrain model (DTM). Besides, the universality of the framework allows us to handle any size of scenes and any shape of buildings without beforehand limitations or assumptions. In addition, the flexibility of the framework allows to replace the model (CNNs) freely.

The main contributions of this paper are listed as follows:

- We propose a deep learning-based framework for building extraction from multi-spectral LiDAR data, which only inputs raw multi-spectral point clouds and directly outputs point-wise building extraction results.
- We propose a sample generation method to generate the samples from raw multi-spectral LiDAR data, which have structured data to meet the input requirement of CNNs and achieve the full coverage of the original input point clouds.
- We propose a novel learn-from-geometric-moments convolution operator, called GGM convolution, that can explicitly encode the local geometric structure of a point set.
- We propose a hierarchical architecture equipped with the GGM convolution, called GGM convolutional neural networks, which achieves the best performance of building extraction by comparing with previous state-of-the-art networks.

The rest of this paper is organized as follows: Section 2 presents the related work. Section 3 introduces the study area and the preprocessing of multi-spectral LiDAR data. Section 4 details the proposed framework and model. Sections 5 and 6 present and discuss the experimental results, respectively. Section 7 provides the concluding remarks and the future work.

## 2. Related Work

To the best of our knowledge, there are no previous studies about building extraction directly from multi-spectral LiDAR data. Thus, we review the previous works related to only LiDAR data. Correspondingly, in terms of input data, the methods are grouped into two categories: the only raw LiDAR data and the integration of raw LiDAR data and additional remotely sensed data. In terms of approaches to be used, generally, building extraction methods using LiDAR data are grouped into model-driven and data-driven methods. The former estimates the buildings by fitting the input data to a hypothetical model library [12,17], e.g., flat- and gable-roof buildings; thus, the building extraction results are always topologically correct and relatively robust. However, for a complex building, the respective model may not be present in the model library [18]. In contrast, the latter has no limitations on the building appearance and can recognize buildings with any shape. Because deep-learning-based methods belong to the data-driven approaches, we will review and discuss the most representative data-driven methods in terms of their inputs.

### *2.1. The Use of Only Raw LiDAR Data*

Maas and Vosselman [19] presented two approaches for the automatic derivation of building models from laser altimetry data. For the first approach, they utilized the invariant moments of point clouds to model the shape of buildings and calculated the closed solution for the parameters of different kinds of building types to extract building points. After constructing the triangular meshes of the extracted building points, the second approach detected the planar roof faces by clustering the Delaunay triangulation and determined the roof outline with a Douglas–Peucker-like algorithm. Both of these two approaches required no additional data, such as 2D GIS data, and do not need to convert the original 3D data points into other data forms. However, the first extraction approach can only model the limited types of buildings, which constrained its applications.

Dorninger and Pfeifer [12] proposed a comprehensive approach for automated determination of 3D city models from airborne laser scanning (ALS) data. They used the results of mean shift segmentation as the initial roof area. Then, the roof points were extracted by applying an iterative region growing process. The approach can generate detailed 3D building models with rooftop overhangs. However, manual interventions were required during the preprocessing and post-processing steps. Besides, for the complex building rooftop structures, the interior structure lines cannot be well extracted.

Zhou and Neumann [20] proposed an automatic algorithm that reconstructed building models from LiDAR data of urban areas. First, they utilized the SVM algorithm to classify the vegetation from other urban objects. Then, they identified the ground points and roof points by using a distance-based region-growing algorithm. However, some tree points with similar heights to the buildings might not be eliminated clearly, and some of the ground points were divided by the roads into patches, both of which caused the outlier points of building extraction result.

Poullis and You [21] proposed a method for the rapid reconstruction of photorealistic large-scale virtual environments. Based on the segmented data, they used the region-growing algorithm to detect the buildings and applied a polygonal Boolean operation to refine the boundaries. For the complex and nonlinear surfaces buildings, some control points were needed to be specified. Obviously, the building extraction results of this approach depended on the segmentation algorithm, and the requirement of interactive operation limited its application.

Sampath and Shan [13] presented a solution framework for the segmentation and reconstruction of polyhedral building roofs from LiDAR data. First, they applied eigen analysis to every point to exclude the nonplanar points. Then, they used the fuzzy k-means algorithm to cluster the planar points, instead of the commonly used region-growing based methods. Finally, the clustered points were merged by the breakline into the integrated rooftop. For this approach, although the feature elements of the most sampled rooftops could be obtained by adjacency matrix, the complex rooftop models, e.g., Dutch gable rooftop, would not be generated correctly.

Zou et al. [22] proposed a method based on a strip strategy to filter the building points and extract the edge point set from LiDAR data. First, they divided the point clouds into several data strips and filtered building points from them with an adaptive-weight polynomial. Then, the building edges were extracted by a modified scanline method. However, this method was only suitable for urban area with dense buildings.

Santos et al. [23] proposed a building roof boundary extraction method from LiDAR data. The method overcame the limitation of the original alpha-shape algorithm by applying an adaptive strategy. A local parameter $\alpha$ was estimated for each edge, instead of using a global parameter. With this approach, the extracted boundaries showed better consistency.

Most of the aforementioned methods follow the traditional building extraction architecture and are equipped with the classical machine learning methods as the classifier. There are three main aspects of their common drawbacks. First, the handcrafted computed or selected features are always required for point separation or classification, which leads to additional manual interventions and the assumptions or limitations of building shape or research area. Second, because the handcrafted computed or selected features are just low-level features, such as heights and normal vectors, their distinguishability and representability are not good enough; for example, the ground and non-ground points could not be separated clearly by heights in many cases. Third, the classical machine learning methods need to specify the parameters, but how to adjust the parameters requires professional experience, which limits the universality and application of these approaches.

### *2.2. The Fusion of Raw LiDAR and Additional Data*

In contrast to the aforementioned building extraction approaches, which only use the raw LiDAR data as the input data, there are methods using additional data, e.g., DSM, DTM, orthoimage, and multi-spectral orthoimage, to enhance the extraction performance.

Mohammad et al. [7] proposed a method for automatic 3D roof extraction through an integration of LiDAR data and multi-spectral orthoimage. First, they generated the DEM from raw point clouds normalized difference vegetation index (NDVI) image from multi-spectral orthoimage and an entropy image from greyscale orthoimage, respectively. Then, the ground height from the DEM was used to separate the ground points and non-ground points, and the structural lines extracted from greyscale orthoimage were classified into various classes by using the NDVI image and entropy image. Finally, the lines belonging to the "building" class were used to fit the planes and boundaries. Their further work [24] added the texture information from the orthoimage and iteratively applied a region-growing technique to extract the complete roof plane. Compared with their works [25,26], which only used LiDAR data as the input data, this method has further enhanced the building extraction effectiveness. However, they mainly used LiDAR data as the source data of the DEM, which did not make full use of the precious spatial geometry information contained in the LiDAR data.

Gilani et al. [27] proposed a method to extract and regularize the buildings using features from LiDAR data and orthoimage. Similar to Awrangjeb et al. [7], first, they generated the building mask, height difference, NDVI image, entropy image, image lines from LiDAR data and orthoimage. Then, they detected the candidate building regions by using connected component analysis and estimated the boundaries using the Moore–Neighborhood tracing algorithm. Finally, through a series of processing, the boundaries and building regions were associated and refined to a complete roof. However, this approach had the same drawbacks as in Awrangjeb et al. [7] and inevitably lost information during the data generation.

Sohn and Dowman [28] proposed an approach for automatic extraction of building footprints from the integration of multi-spectral image and LiDAR data. First, they recognized the isolated building by the height information from LiDAR data and NDVI from IKONOS imagery. Then, to obtain the better building boundaries, they used both data-driven and model-driven methods to compensate their weakness. Finally, they merged the convex polygons to obtain the complete building outlines. One of the characteristics in this approach was to use both data-driven and model-driven methods to compensate their weakness, which contributed to a better extraction result and less information loss.

Nguyen et al. [29] presented a super-resolution-based snake model (SRSM) to extract buildings using LiDAR data and optical image. First, they also preliminary extracted the candidate building points by using the DTM and NDVI derived from LiDAR data and optical image, respectively. Then, they generated the Z-image by the super-resolution of LiDAR data. Finally, they used the SRSM to extract the building boundary points. With the SRSM, this approach achieved better performance of building boundary generation than the basic snake and previous modified snake model. However, this approach only used the height information of LiDAR data, and the extraction results mainly depended on the SRSM.

The approaches taking the fused data as input are facing two main problems. First, the data collected from different sensors have different data format, projection, resolution and collection time. Thus, the errors are inevitably produced during the data fusion process. Next, because most of these approaches extract buildings by using DEM, DTM, NDVI, or other information derived from the input data, only a small part of input spatial and spectral information is utilized. The precious spatial information contained in LiDAR data is usually ignored.

## *2.3. The Deep-Learning-Related Methods*

With the success of deep convolutional neural networks for image processing, many researchers have tried to apply CNNs to extract buildings on airborne LiDAR data. However, this is still a primeval field to research. To the best of our knowledge, there are few deep-learning-related approaches developed to extract buildings from LiDAR data.

Bittner et al. [15] proposed a method to automatically generate a building mask out of a DSM using a fully convolution network (FCN) architecture. Unlike the previous methods, which generated the mask to separate the ground and non-ground points from the DSM and NDVI, they utilized an FCN to learn a binary building mask from the DSM. Besides, they combined Conditional Random Field (CRF) and FCN as a comparison version to achieve the better performance. However, this approach needed to convert the LiDAR data into the DSM and used the DSM as the input fed into the FCN. That means only a part of the LiDAR data (mainly height) has been used.

Nahhas et al. [16] proposed a building detection approach based on deep learning using the fusion of LiDAR data and orthophotos. Compared with the previous methods, this approach went beyond using the DSM or the NDVI to create the mask by fusing various low-level features derived from the orthophoto and LiDAR data, e.g., the spectral information, DSM, DEM, nDSM. Then, they utilized the powerful learning ability of CNN to extract the high-level features from the input low-level features to recognize the building points. Nevertheless, this method needed to convert the LiDAR data and caused information loss during the data or feature fusion.

Maltezos et al. [14] proposed a building extraction method using LiDAR data by applying deep CNNs. Unusually, they did not simply derive the DSM from LiDAR data but extracted the features from raw LiDAR data with a physical interpretation. The seven extracted features (or information) included entropy, height variation (HV), intensity, distribution of the normal vectors, number of returns (NR), planarity, and standard deviation (STD). Then, the features were fed into a CNN to learn an optimal classification result. Like the former two methods, this approach did not use the raw LiDAR data as the inputs to the neural networks, and there was information loss during the feature extraction stage inevitably. Actually, the CNN was just taken as a powerful classifier in this approach.

In conclusion, the existing deep-learning-related approaches still cannot directly handle the raw point clouds as the input. Although the current approaches try to avoid the information loss as much as possible during the data transformation or data fusion stage, but it seems still an open problem.

Thus, in this paper, we propose a deep-learning-based framework and model to extract buildings directly from the raw multi-spectral LiDAR data.

## 3. Study Area and Data Preprocessing

### *3.1. Study Area*

As shown in Figure 1, the study area is a small town located in Whitchurch-Stouffville, Ontario, Canada. The area of the whole study area is about 3.2 km$^2$, and the latitude and longitude of the center position are 43°58′00″ and 79°15′00″, respectively. We selected 13 typical scenes as the training and test scenes, which are rendered with red boxes (training scenes) and blue boxes (test scenes) in Figure 1. Each selected scene contains many types of ground objects, such as roads, trees, grass, buildings, and soil, which contribute to our method to study in a real-world complex scene. Table 1 shows the size and total number of points for the selected scenes.

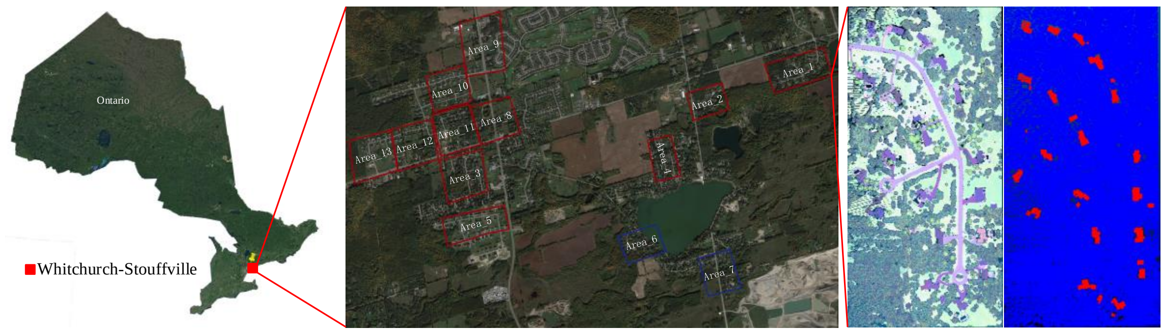

**Figure 1.** The study area, the general view of the selected scenes and a sample of the corresponding labeled data.

**Table 1.** The size and total number of points in each selected scene.

| | Area_1 | Area_2 | Area_3 | Area_4 | Area_5 | Area_6 | Area_7 | Area_8 | Area_9 | Area_10 | Area_11 | Area_12 | Area_13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Size (m$^2$) | 176,938 | 98,813 | 178,668 | 104,882 | 153,575 | 108,009 | 129,332 | 149,907 | 241,053 | 149,838 | 163,088 | 165,978 | 162,742 |
| Points | 697,838 | 425,409 | 747,342 | 418,220 | 556,183 | 325,924 | 598,398 | 695,190 | 887,487 | 653,780 | 864,581 | 758,588 | 626,285 |

The experimental data were collected by using an airborne Titan multi-spectral LiDAR system, produced by Teledyne Optech (Toronto, ON, Canada). The detailed specifications of the multi-spectral LiDAR system are presented in Table 2. Radiometric calibration [30] of the Titan multi-spectral LiDAR system has been done before data collection. Since we cannot obtain the system parameters of Titan and the trajectories of data collection, we directly used the three channels of intensities from the LiDAR outputs without intensity correction. As for the geometric correction, similarly, because we have no ground control points or reference points, we just simply used the iterative closest points (ICP) to register these strips. Therefore, we chose the training and test scenes from the same strip, to avoid the problems caused by intensity correction or geometric correction.

**Table 2.** Specifications of the Titan Airborne System.

| Parameters | Channel 1 | Channel 2 | Channel 3 |
|---|---|---|---|
| Wavelength (nm) | 1550 (SWIR) | 1064 (NIR) | 532 (GREEN) |
| Deflection Angle (°) | 3.5 (forward) | nadir | 7 (forward) |
| Flight Altitude (m) | ~1000 | ~1000 | ~1000 |
| Point Density (m$^2$) | 3.6 | 3.6 | 3.6 |

### *3.2. Data Preprocessing*

As we can see in Table 2, the original acquired raw multi-spectral LiDAR data contains three channels of individual spatial coordinates and spectral values. Thus, we have to preprocess the original individual data into the fused data firstly. Because data preprocessing is not the main focus of this paper, here, we adopt a simple multi-spectral LiDAR data preprocessing strategy proposed in [31].

The three laser channels of the Titan multi-spectral LiDAR system are 1550 nm, 1064 nm, and 532 nm. The preprocessing is to fuse the three independent spectral values with three different points into one single point with three spectral values. First, we select one of the three single-wavelength point clouds as the reference data. Then, we find the nearest neighbor point from the other two single-wavelength point clouds at the same scanning. Finally, we fuse the other two spectral values of the nearest neighbor points into the reference data. In this way, the original acquired three-wavelengths point clouds are fused into the multi-spectral point clouds with a third number of points.

## **4. Methodology**

### *4.1. Framework Overview*

Similar to [32], we propose a framework (Figure 2) designed for building extraction from multi-spectral LiDAR data, which also contains the common process steps.

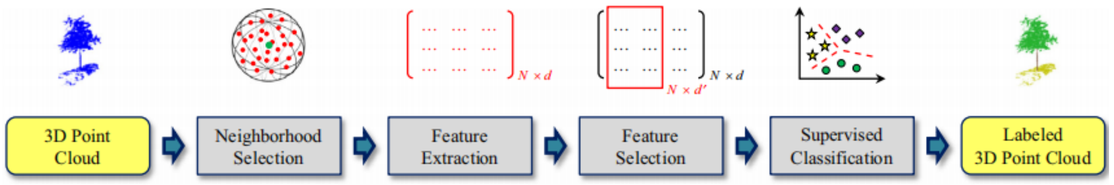

**Figure 2.** The overview of the point cloud semantic interpretation framework.

After data preprocessing, we obtain the available multi-spectral LiDAR data. As a supervised method, we have to manually label each of the selected training and test areas before we feed them into the framework. As shown in Figure 3, our proposed deep learning-based building extraction framework consists of two main stages. First, we feed the labeled training scenes into the GGM convolutional neural networks. Then, we use the trained model to recognize the building points from the input test scenes. Remarkably, the framework requires only point cloud data as the input and directly outputs the labels of each point in the test scenes. There are no limitations about the number of training and test scenes and the size of each input scene. The framework requires no assumptions of the shape and size of the buildings. Furthermore, the model used for training and test is replaceable. That is, any networks, only if they can output the required data form, can be applied in this framework.

During the sample generation stage, the training and test scenes are split into individual samples with a fixed size. Thus, the sampled data can be directly fed into the neural networks, and the input scenes are completely covered by the sampled data at the same time. The details are illustrated in Section 4.2.

For the building points recognition task, we designed a convolution operator, called GGM convolution, which learns local geometric features from geometric moments representation of a local point set. Then, a hierarchical architecture equipped with the GGM convolution contributes to our model, called GGM convolutional neural networks. The related details are illustrated in Section 4.3.

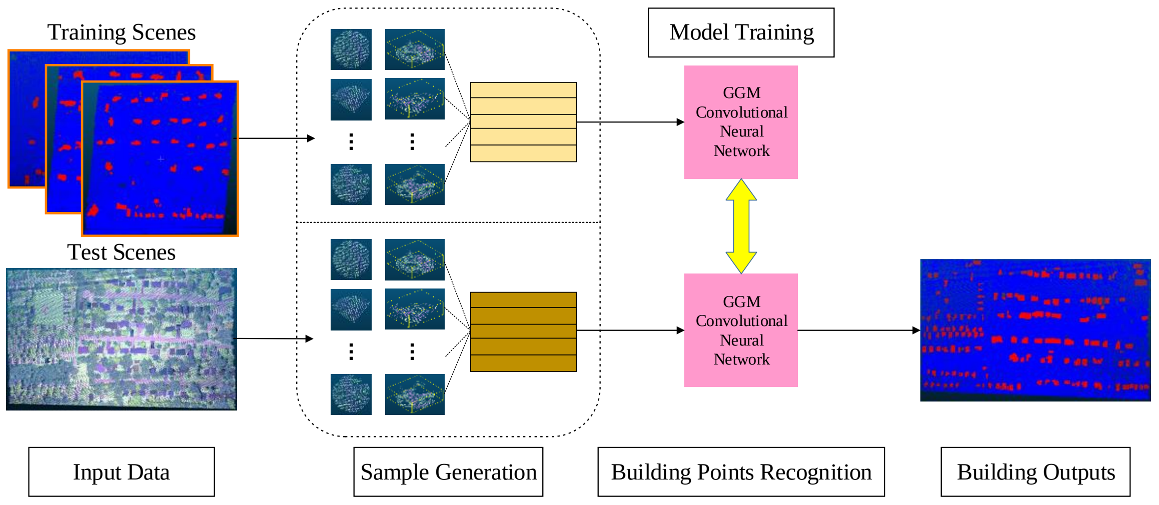
Training Scenes
Test Scenes
Model Training
GGM Convolutional Neural Network
GGM Convolutional Neural Network
Input Data
Sample Generation
Building Points Recognition
Building Outputs

**Figure 3.** Deep-learning-based framework for building extraction.

## *4.2. Sample Generation*

To keep the scene integrity and ensure every point in the original scene being labeled, inspired by RandLA-Net [33], we propose a farthest point sampling–k nearest neighbors (FPS-KNN) sample generation method to generate the training and test samples for neural networks. The samples generated by the FPS-KNN both satisfy the input data form requirement of standard convolutional neural networks and obtain the full coverage of the scene. Figure 4 shows the data processing workflow with the FPS-KNN method. The details of the FPS-KNN sample generation method are carried out as follows:

Step 1: For a given scene, we duplicate an identical point set as the evaluation point set. We randomly choose one point in the evaluation point set as the seed point and search its k nearest neighbors in the original point set. The value of k is set depending on the sample size. For example, if each sample contains 4096 points, then the value of k is configured as 4096.

Step 2: We calculate the distance from the rest points in the evaluation point set to the seed point and select the most distant point as the next seed point. The seed point and its k nearest neighboring points are saved as one sample and removed from the evaluation point set.

Step 3: We iteratively search the farthest point as the seed point in the evaluation point set, search its k nearest neighbors in the original point set, and remove the sampled points from the evaluation point set, until the evaluation point set is empty.

Thus, we obtain numerous samples with the fixed number of points from the given scene, which can be directly fed into a standard convolutional neural network. At the same time, we can ensure that every point in the scene is contained in some samples, which means the full coverage of the scene. We also notice that some samples are inevitably overlapped. For the points within the overlapped part, we choose the most predicted label as its final predicted label.

In this way, theoretically, for any scene, the generated samples can be directly fed into the neural networks by the proposed FPS-KNN sample generation method to obtain the predicted label for every point in the scene.

## *4.3. Graph Geometric Moments Convolutional Neural Networks*

### 4.3.1. Geometric Moments

Moments and functions of moments have been widely utilized as pattern features in pattern recognition [34–36], edge detection [37,38], image segmentation [39], texture analysis [40], and other domains of image analysis [41,42] and computer vision [43,44].

The general two-dimensional $p+q$th order moments of a density distribution function $f(x,y)$ is defined as follows:

$$m_{pq} = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} x^p y^q f(x,y)dxdy \tag{1}$$

where $p, q = 0, 1, 2, \ldots$. The lower order moments (small values of $p$ and $q$) have well defined geometric interpretations. For example, $m_{00}$ is the area of the region, ${}^{m10}/{}_{m00}$ and ${}^{m01}/{}_{m00}$ give the $x$ and $y$ coordinates of the centroid of the region, respectively [38]. Similarly, the three-dimensional geometric moments of $p+q+r$th order of a 3D object is defined as follows [39]:

$$m_{pqr} = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} x^p y^q z^r f(x,y,z)dxdydz, \tag{2}$$

where $p, q, r = 0, 1, 2, \ldots$. The discrete implementation of the moments of a 3D homogeneous object could be defined as follows [38]:

$$m_{pqr} = \sum_{\mathbb{R}^3} x^p y^q z^r f(x,y,z), \tag{3}$$

where $\mathbb{R}^3$ is a 3D region. For the 10 low order 3D moments (order up to 2), we have:

$$\begin{aligned}
m_{000} &= \sum_{\mathbb{R}^3} f(x,y,z)\\
m_{100} &= \sum_{\mathbb{R}^3} x \cdot f(x,y,z)\\
m_{010} &= \sum_{\mathbb{R}^3} y \cdot f(x,y,z)\\
m_{001} &= \sum_{\mathbb{R}^3} z \cdot f(x,y,z)\\
m_{110} &= \sum_{\mathbb{R}^3} x \cdot y \cdot f(x,y,z)\\
m_{101} &= \sum_{\mathbb{R}^3} x \cdot z \cdot f(x,y,z)\\
m_{011} &= \sum_{\mathbb{R}^3} y \cdot z \cdot f(x,y,z)\\
m_{200} &= \sum_{\mathbb{R}^3} x^2 \cdot f(x,y,z)\\
m_{020} &= \sum_{\mathbb{R}^3} y^2 \cdot f(x,y,z)\\
m_{002} &= \sum_{\mathbb{R}^3} z^2 \cdot f(x,y,z)
\end{aligned} \tag{4}$$

For a raw point cloud, we define its geometric moments representation referring to [45] as follows:

$$M_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, M_2 = \begin{bmatrix} xy \\ xz \\ yz \\ x^2 \\ y^2 \\ z^2 \end{bmatrix}, \tag{5}$$

where $M_1$ and $M_2$ are the first and second order geometric moments of a point cloud, respectively. Usually, parameters fitting ability is one of the most fundamental and powerful abilities of neural networks. The different order of geometric moments representation of point clouds could be seen as the spatial distribution function with the different variables and orders. Thus, it can upgrade the neural networks to learn more precise features by feeding the geometric moments representation of point clouds.

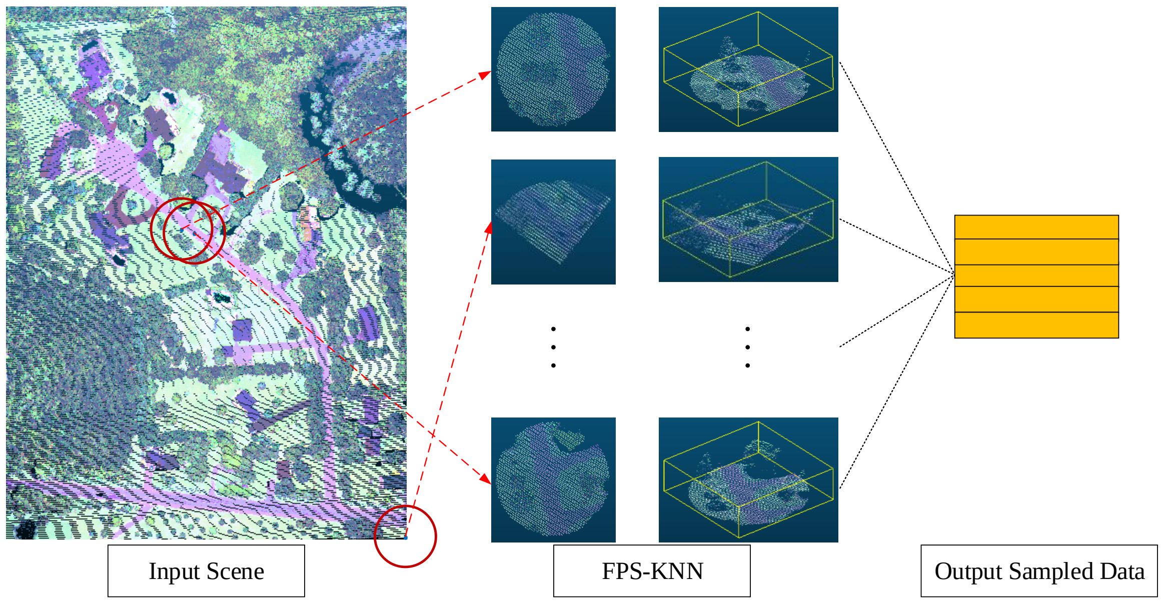

**Figure 4.** Sample generation workflow with the FPS-KNN method.

Besides, the moment-based methods have advantageous qualities like translation and rotation invariance, both of which are important properties for feature descriptors. Translation invariance is obtained by using the central moments for which the origin is at the centroid of the density function [40]. For 3D objects, the translation invariance is obtained by using the central moments $\mu_{pqr}$ defined in the same way as for 2D objects [34]. The central moments $\mu_{pqr}$ is defined as follows:

$$\mu_{pqr} = \sum_{\mathbb{R}^3} (x-\overline{x})^p (y-\overline{y})^q (z-\overline{z})^r f(x,y,z), \tag{6}$$

where $(\overline{x}, \overline{y}, \overline{z})$ is the centroid of the object, which can be obtained from the first order moments,

$$\overline{x} = \frac{m_{100}}{m_{000}} \quad \overline{y} = \frac{m_{010}}{m_{000}} \quad \overline{z} = \frac{m_{001}}{m_{000}}. \tag{7}$$

The higher order moments represent more detailed shape characteristics [40], which means more comprehensive geometric features in deep learning. Mo-Net [45] first utilizes the second order geometric moments representation of point clouds as the input features. Compared with PointNet [46], which only considers the first order geometric moments, Mo-Net validates the function of higher order geometric moments. Inspired by that, we design our network to learn features from the geometric moments representation of point clouds.

### 4.3.2. Graph Generation

Since the graph neural networks (GNNs) proposed by [47], it has been widely used in learning on unstructured data. GNNs apply neural networks for walks on the graph structure, propagating node representations until a fixed point is reached. The resulting node representations are then used as features in classification and regression problems [48]. To apply the graph neural network to the point cloud, first, we need to convert it to a directed graph.

A graph $G$ is a pair $(P, E)$ with $P = \{p_1, \ldots, p_n\}$ denoting the set of vertices and $E \subseteq P \times P$ representing the set of edges. As the consideration of computational complexity, most of the networks would rather construct a k-nearest neighbors (KNN) than a fully connected edges for the whole point cloud.

As shown in Figure 5, we utilize the k-nearest neighbors of each point to construct a local directed graph. In this local directed graph, point $p_i$ is a central node, and $e_{ij}$ are the edges between the central node and its k-nearest neighbors, which are calculated as follows:

$$\begin{aligned} &G = (P, E) \\ &P = \{p_i \big| i = 1, 2, \ldots, n\} \\ &E = \left\{e_{ij} \middle| j = 1, 2, \ldots, k\right\} \\ &e_{ij} = p_{ij} - p_i \end{aligned} \tag{8}$$

where $p_{ij}$ are the neighbors of the central point $p_i$.

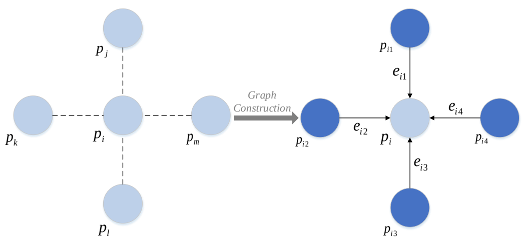

**Figure 5.** Graph construction of a point cloud. The $p_i$, $p_j$, $p_k$, $p_l$, $p_m$ are the points in the point cloud. The $p_j$, $p_k$, $p_l$, $p_m$ on the left and $\{p_{i1}, \ldots, p_{i4}\}$ on the right are the nearest neighbors of $p_i$. The directed edges $\{e_{i1}, \ldots, e_{i4}\}$ are the edges from the neighbors to the central point.

### 4.3.3. GGM Convolution

In this section, first, we detailed introduce the design of the GGM convolution, which acts as the core module of our model, and then analyze the reason why we design in this way.

Generally, the core module could be seen as an encoder with the given input and output dimension. Meanwhile, to fit various network architectures, the core module should be designed with sufficient universality and pluggability, such as the EdgeConv in DGCNN and relation-shape convolution in RS-CNN. Inspired by the previous core modules, we design the GGM convolution by using the geometric moments representation of point clouds, Figure 6 shows the details of the GGM convolution.

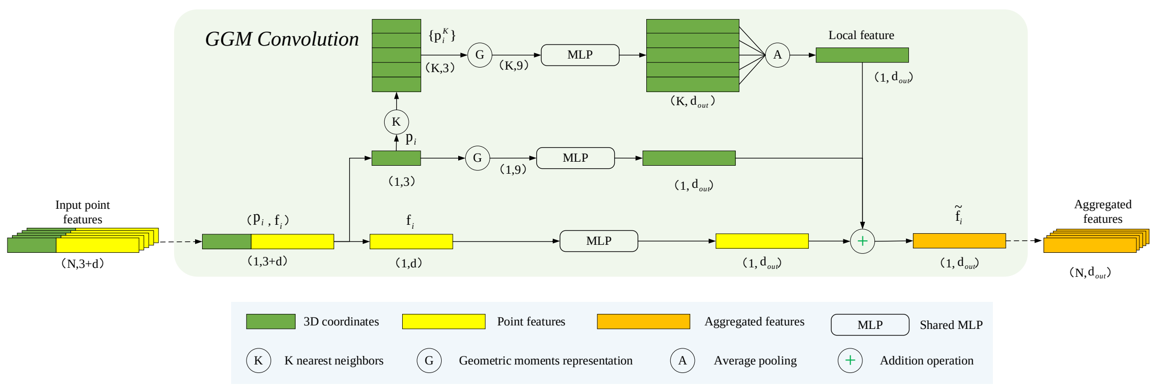

**Figure 6.** Architecture of the graph geometric moments (GGM) convolution.

Given a $d$-dimensional point cloud with $n$ points, denoted by $X = \{p_1, \ldots, p_n\} \subseteq R^F$. For the initial input, $d = 3$, which indicates the spectral values of three channels. In a hierarchical neural network, the subsequent layer operates on the output of the previous layer, so more generally the dimension $d$ represents the feature dimension of a given layer [49], which indicates as the point features in Figure 6.

As show in Figure 6, the point features are combined with its 3D coordinates as the input to the GGM Convolution, and the GGM Convolution contains two main branches. The bottom branch indicates the input point features directly fed into a multi-layer perceptron (MLP), which is a skip connection like residual block. The other branch is designed to extract the local features of each point. Firstly, we construct a local directed graph by searching its k-nearest neighbors and calculate the first and second order geometric moments representations of the point and its local directed edges, respectively. Then, they were separately fed into two independent MLPs, and the output of the MLP on the top branch is aggregated by the average-pooling operation. Finally, an addition operation is utilized to fuse all the outputs.

The analysis of local features aggregation strategy: The reason why we use the average-pooling operation instead of the max-pooling operation to aggregate the extracted local features is that we want to obtain the local feature as the compensation of the point feature. The max-pooling operation takes only the max value at each feature channel, which tends to capture the most "special" features and shows less representativeness. To guarantee the extracted compensation feature is sufficiently reliable, the more reasonable local feature should be the average of all local features extracted from the edges.

The analysis of global features aggregation strategy: Although the concatenation and multiplication operations are quite commonly used in related methods. For example, PointNet++ [50] and DGCNN [49] fuse features by using concatenation operation, RS-CNN [51] and GACNet [52] fuse features by using multiplication operation. Here, we choose the addition operation to fuse features. The main reasons are as follows: (1) the concatenation operation is effective to fuse the multiscale features, and the multiplication operation is commonly used in attention mechanism methods. However, we are fusing the features extracted from higher order geometric moments of original coordinates, which contain different forms of underlying geometric information. Thus, we cannot use the concatenation or multiplication operations roughly here. (2) Essentially, the feature space in deep learning is a kind of probability space, the convolution could be viewed as the filter. The value in different channel of the output feature shows the probability that passes the filter with specific parameters. The addition operation could highlight the befitting filters and restrain the improper filters, which effectively refine the point feature.

### 4.3.4. Network Architecture

Figure 7 shows the detailed architecture of the GGM Convolutional Neural Networks. The network follows the widely used hierarchical structure. After sample generation, the point clouds of each test area are split into many batches, and each batch contains 4096 points. Through the GGM Convolutional Neural Networks, the input points, which contains spatial coordinates and spectral values of three channels, are labeled with their predicted labels, e.g., 1 indicates the building point and 0 indicates the background point. The details of the GGM Convolutional Neural Networks are illustrated as follows:

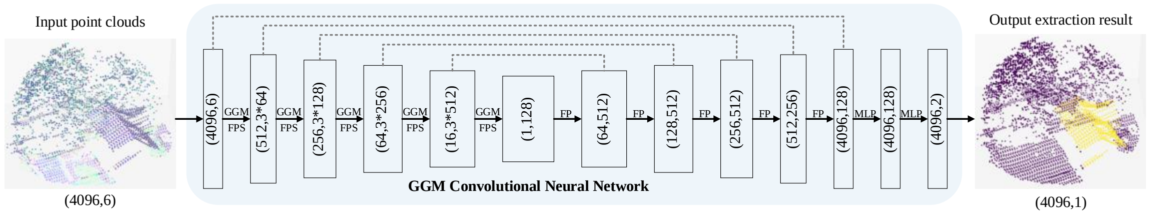

**Figure 7.** GGM convolutional neural networks architecture. $(N, D)$ represents the number of points and feature dimension respectively. GGM: graph geometric moment convolution, FPS: farthest point sampling, *FP*: feature propagation, MLP: multi-layer perceptron.

Hierarchical structure: Our hierarchical structure is referenced from PointNet++. The hierarchical structure is composed of a number of set abstraction levels. The set abstraction level is made of the following two key layers: the sampling layer and the GGM convolution layer. The sampling layer selects a set of points from the input points via the Farthest Point Sampling (FPS) algorithm, which defines the centroids of local regions. The GGM convolution layer is illustrated in Section 4.3.3, which combines local feature extraction and grouping function. A set abstraction level takes an $N \times (d + C)$ matrix as input that is from $N$ points with $d$-dimensional coordinates and $C$-dimensional point feature. It outputs an $N' \times (d + C')$ matrix of $N'$ subsampled points with $d$-dimensional coordinates and new $C'$-dimensional feature vectors summarizing local features.

Farthest point sampling (FPS): In the sampling layer, we utilize iterative farthest point sampling (FPS) to choose a subset of points. Given the input points $\{x_1, x_2, \ldots, x_n\}$, firstly, the FPS randomly picks one point $x_1$ as the seed point, then, calculates the distance from the input points to the seed point and selects the most distant point as the next seed point. The selected points will be removed from the input points. Finally, all the selected seed points constitute the subset of input points with a specified size. In this way, the selected subset of input points could have good coverage of the entire input points.

Multi-scale grouping (MSG): Inspired by PointNet++, we implement the MSG strategy to make our model more robust. For every set abstraction level, we apply a GGM convolution with three different scales, e.g., we set the k-nearest neighbors of 16, 32, and 48 for the first set abstraction level. Then, the features at different scales are concatenated to form a multi-scale feature. Thus, as shown in Figure 7, we use 3*D to indicate the number of scales and the dimension of features at different scales, respectively.

Feature propagation (*FP*): To predict the labels for all the original points, we need to propagate features from subsampled points to the original points. Here, we choose a hierarchical propagation strategy similar to PointNet++. Firstly, we find one nearest neighboring point for each point, whose point feature set is up-sampled through a nearest-neighbor interpolation. Then, the up-sampled features are concatenated with the intermediate feature produced by set abstraction layers through skip connections, which is indicated by the dotted lines in Figure 6. Finally, we apply a shared MLP and ReLU layer to the concatenated features to update each point's feature vector.

Final label prediction: The final label of each point is obtained through two shared MLP with 128 and two output dimensions. After a softmax operation, the max value of the two channels indicates the final predicted label.

## 5. Experimental Results

### 5.1. Implementation Details

Our training strategy is the same as in [49]. We used the cross entropy as the loss function. We used the stochastic gradient descent (SGD) optimizer with 0.1 as the initial learning rate in our network, and the learning rate declined fifty percent after each thirty iterations. Since we applied the MSG strategy in our model, the number of the nearest neighbors $k$ varied from 16 to 64 in different set abstraction levels. The training iteration was configured as 200. The number of input points, batch size, and momentum were 4096, 16, and 0.9, respectively. For every MLP layer, we used the LeakyReLU with 0.2 negative slope as the activation function and applied Batch normalization. After training the whole network, we saved the best performance training variables of the network and set it as the input in the retraining process. We adjusted the hyper-parameters during the retraining process. Furthermore, we trained our model on a NIVIDIA 2080 TI GPU (2788 San Tomas Expressway Santa Clara, CA 95051, US).

### 5.2. Accuracy Evaluation Metrics

To assess the quality of the proposed method, we adopted the commonly used metrics for sematic segmentation and binary classification task. Let *TP*, *FP*, and *FN* denote the total numbers of true positives, false positives, and false negatives, respectively. Then, the precision/correctness, recall/completeness are calculated as following:

$$Precision = \frac{TP}{TP + FP}, \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

where the *Precision* is the proportion of the true positives over the extracted building points, and the *Recall* is the proportion of true positives with regard to the labeled ground-truth building points. The higher these metrics, the better the performance of the method.

Besides, we used the *F-measure* derived from the precision and recall values for the point-based overall evaluation, which is defined as follows:

$$F\text{-}measure = \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP}. \tag{11}$$

For simplicity, we set $\beta = 1$.

In addition, we also employed the intersection over union (*IoU*) as one of the metrics. *IoU* is the proportion of intersection of prediction and ground truth points over the union of them, which is calculated as following:

$$IoU = \frac{TP}{TP + FP + FN}. \tag{12}$$

## 5.3. Parameter Sensitivity

### 5.3.1. Spectral Information

To investigate the effect of the input feature selection, e.g., spectral information, we trained our model based on the five sets of input data. Since the main characteristic of our model is learning local features from geometric moments, we considered the spatial coordinates as the essential feature. The first model was trained using 3D coordinates only. To better evaluate the effect of each single spectral value of three channels, the next three models were trained by using three individual single-wavelength, i.e., 3D coordinates and one spectral values of each channel, respectively. The last model was trained using both 3D coordinates and spectral information (spectral values of three channels) for each point.

We evaluated our model on area_6 and area_7. After sample generation, these two test scenes were split into 257 and 474 samples, respectively. As we mentioned in Section 4.2, for the overlapped part between samples, we counted the predicted labels from different samples of the same point and chose the most predicted label as its final predicted label. After we obtained the predicted label for each point in the test scenes, we evaluated the extracted point-wise results by the four metrics mentioned above. Here, we defined the point-based evaluation result of the combination of the test scenes as the comprehensiveness result, instead of the commonly used average result.

As shown in Table 3, the last model achieved best performance on Area_6, Area_7, and comprehensiveness for each metric. This suggests that combining additional features could improve the accuracy of the results. From second to fourth models evaluated the effect of different spectral values from different wavelength (channel). Obviously, with the spectral values collected from channel 1 and 2, the accuracies of building extraction had been improved significantly. That motivated us to find the corresponding wavelength spectral images and check what it is the visualization of building and background on the images. As shown in Figure 8, on the visual effects, the intensity image from channel 2 (1064 nm) shows the best separability of building from background, and the intensity image from channel 3 (532 nm) shows the worse separability than the others. The separability could be seen as how “helpful” these additional features are for the building extraction task. Remarkably, the results of comparison experiments show the same trend with the visual effects. That validated the reasonability and interpretability of the proposed method from another side. The results also validate the powerful geometric feature learning ability of our model. The results are quite promising even by only using 3D coordinates as input.

**Table 3.** A comparison between training with the different input feature.

| Input | Area | *Precision* | *Recall* | *F-measure* | *IoU* |
|---|---|---|---|---|---|
| Coordinates | Area_6 | 86.0 | 85.0 | 85.5 | 74.7 |
| | Area_7 | 85.0 | 85.3 | 85.1 | 74.1 |
| | **comprehensiveness** | 86.6 | 86.1 | 86.3 | 76.0 |
| Coordinates and channel 1 (1550 nm) | Area_6 | 93.9 | 88.6 | 91.2 | 83.8 |
| | Area_7 | 93.6 | 90.3 | 91.9 | 85.1 |
| | **comprehensiveness** | 94.1 | 90.0 | 92.0 | 85.2 |
| Coordinates and channel 2 (1064 nm) | Area_6 | 96.7 | 94.2 | 95.4 | 91.3 |
| | Area_7 | 94.2 | 93.6 | 93.9 | 88.5 |
| | **comprehensiveness** | 94.5 | 93.5 | 94.0 | 88.7 |
| Coordinates and channel 3 (532 nm) | Area_6 | 93.0 | 88.3 | 91.1 | 83.7 |
| | Area_7 | 88.8 | 88.2 | 90.1 | 82.0 |
| | **comprehensiveness** | 90.5 | 89.7 | 90.8 | 83.2 |
| Coordinates and three spectral values | Area_6 | 97.4 | 91.6 | 94.4 | 89.4 |
| | Area_7 | 94.7 | 94.9 | 94.8 | 90.2 |
| | **comprehensiveness** | 95.1 | 93.7 | 94.4 | 89.5 |

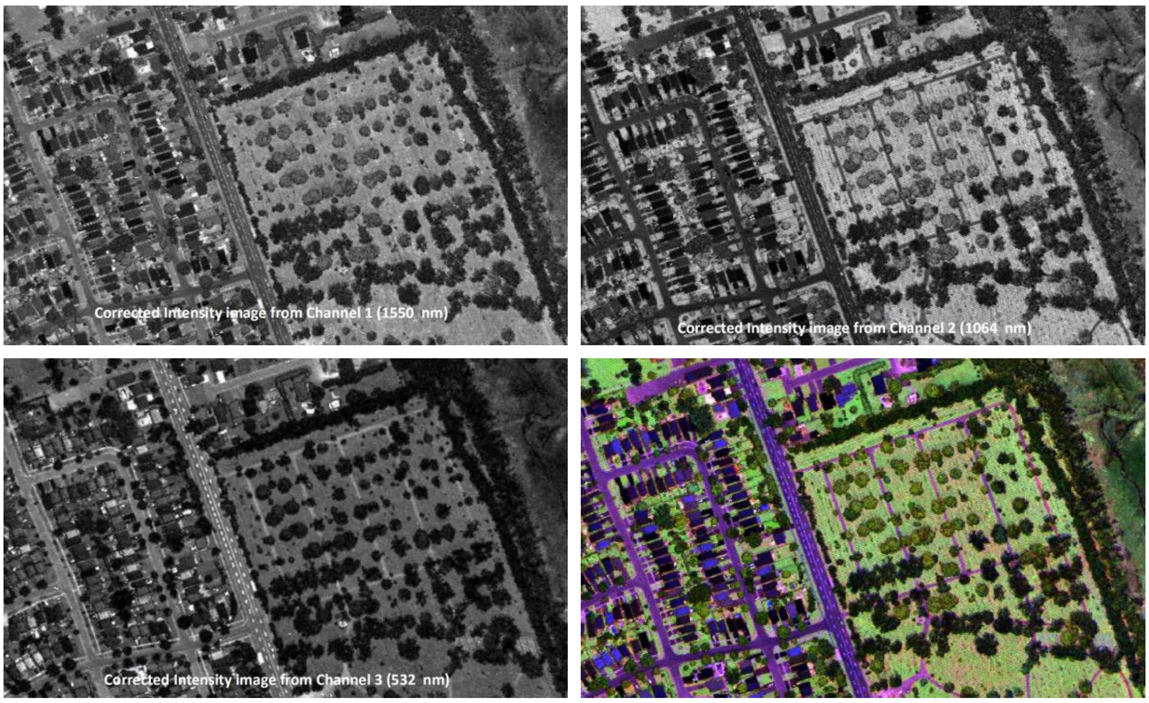
Corrected Intensity image from Channel 1 (1550 nm)
Corrected Intensity image from Channel 2 (1064 nm)
Corrected Intensity image from Channel 3 (532 nm)

**Figure 8.** The corrected intensity images from different channel and the fused false colored image.

### 5.3.2. Sample Size

Furthermore, we investigated the effect of sample size by training our model based on three different sample sizes. As we mentioned in Section 4.2, during the sample generation stage, we can set the number of points each sample contained. Considering the limitation of GPU memory, we set the maximum number of points as 4096, and the other two were set as 2048 and 1024. All the models were trained using the same input features (coordinates and spectral values).

In Table 4, "#points" indicates the number of points in each sample. As we can see, the larger scale performed better than the smaller scale. For deep learning methods, the larger scale input sample provides the more comprehensive information and the better geometric continuity of objects in the scene, which decides "how good" feature the model can learn from. And that is the reason why the larger scale performed better. The results also confirmed our speculation.

**Table 4.** A comparison between training with different sample sizes.

| **#points** | **Area** | ***Precision*** | ***Recall*** | ***F-measure*** | ***IoU*** |
|---|---|---|---|---|---|
| 1024 | Area_6 | 85.6 | 85.7 | 85.6 | 74.9 |
| | Area_7 | 89.2 | 85.4 | 87.2 | 77.4 |
| | **comprehensiveness** | 88.2 | 86.3 | 87.2 | 77.3 |
| 2048 | Area_6 | 87.2 | 85.4 | 86.3 | 75.9 |
| | Area_7 | 92.3 | 86.0 | 89.1 | 80.3 |
| | **comprehensiveness** | 90.6 | 84.9 | 87.6 | 78.0 |
| 4096 | Area_6 | 97.4 | 91.6 | 94.4 | 89.4 |
| | Area_7 | 94.7 | 94.9 | 94.8 | 90.2 |
| | **comprehensiveness** | 95.1 | 93.7 | 94.4 | 89.5 |

## 5.4. Results and Comparisons

Since there is no previous method proposed for building extraction from multi-spectral LiDAR data fitting for our framework, to better evaluate our method, we compared our model with the previous state-of-the-art networks designed for semantic segmentation on point clouds. The compared networks include PointNet [46], KCNet [53], DGCNN [47], and RS-CNN [51].

Table 5 shows the point-based evaluation comparison results for the two test scenes. For all experiments, the input data size was set to 4096 points, and the features included coordinates and three spectral values. As shown in Table 5, our model, GGM convolutional neural networks, achieved significantly better performance than the other networks, especially on the metrics of *Recall* and *IoU*. The PointCNN achieved slightly higher accuracy in area_7 and comprehensiveness, but the other three metrics were observably lower than ours. Hence, for the overall extraction quality, our model achieved a better performance, which was also demonstrated by the following visualization results.

**Table 5.** Point-based building extraction comparison results on test scenes.

| **Model** | **Area** | ***Precision*** | ***Recall*** | ***F-measure*** | ***IoU*** |
|---|---|---|---|---|---|
| PointNet | Area_6 | 74.4 | 55.1 | 63.3 | 46.3 |
| | Area_7 | 72.3 | 56.5 | 63.4 | 46.4 |
| | **comprehensiveness** | 72.6 | 56.1 | 63.3 | 46.3 |
| PointNet++ | Area_6 | 80.5 | 64.0 | 70.5 | 51.1 |
| | Area_7 | 74.4 | 64.9 | 69.1 | 52.8 |
| | **comprehensiveness** | 78.2 | 64.3 | 70.1 | 51.5 |
| PointCNN | Area_6 | 96.7 | 81.3 | 88.3 | 79.1 |
| | Area_7 | 95.8 | 81.7 | 88.2 | 78.9 |
| | **comprehensiveness** | 96.2 | 81.9 | 88.5 | 79.4 |
| KCNet | Area_6 | 96.0 | 77.8 | 85.9 | 75.3 |
| | Area_7 | 92.8 | 78.6 | 85.1 | 73.9 |
| | **comprehensiveness** | 93.6 | 78.0 | 85.3 | 74.1 |
| DGCNN | Area_6 | 79.5 | 76.2 | 77.8 | 63.7 |
| | Area_7 | 79.5 | 73.4 | 76.4 | 61.8 |
| | **comprehensiveness** | 79.3 | 73.6 | 76.4 | 61.8 |
| RS-CNN | Area_6 | 80.9 | 77.8 | 79.3 | 65.8 |
| | Area_7 | 87.3 | 78.6 | 82.7 | 70.6 |
| | **comprehensiveness** | 85.0 | 79.2 | 82.0 | 69.5 |
| Ours | Area_6 | 97.4 | 91.6 | 94.4 | 89.4 |
| | Area_7 | 94.7 | 94.9 | 94.8 | 90.2 |
| | **comprehensiveness** | 95.1 | 93.7 | 94.4 | 89.5 |

Figure 9 shows the visualization of the overall comparison results on Area_7. For each model, we selected the same test area to show its overall extraction result. As reflected by the overall results, most of the models recognized all buildings in object-level regardless of building size, even the small-size buildings (less than 5 m$^2$) were recognized with a part of points. This demonstrated the powerful inference capability of deep learning methods. Our model achieved a more complete building extraction result with less misrecognition points. For example, the PointNet and RS-CNN misrecognized some powerline points as the building points, because they have the similar altitudes, which was indicated by the black circle in Figure 9a,d.

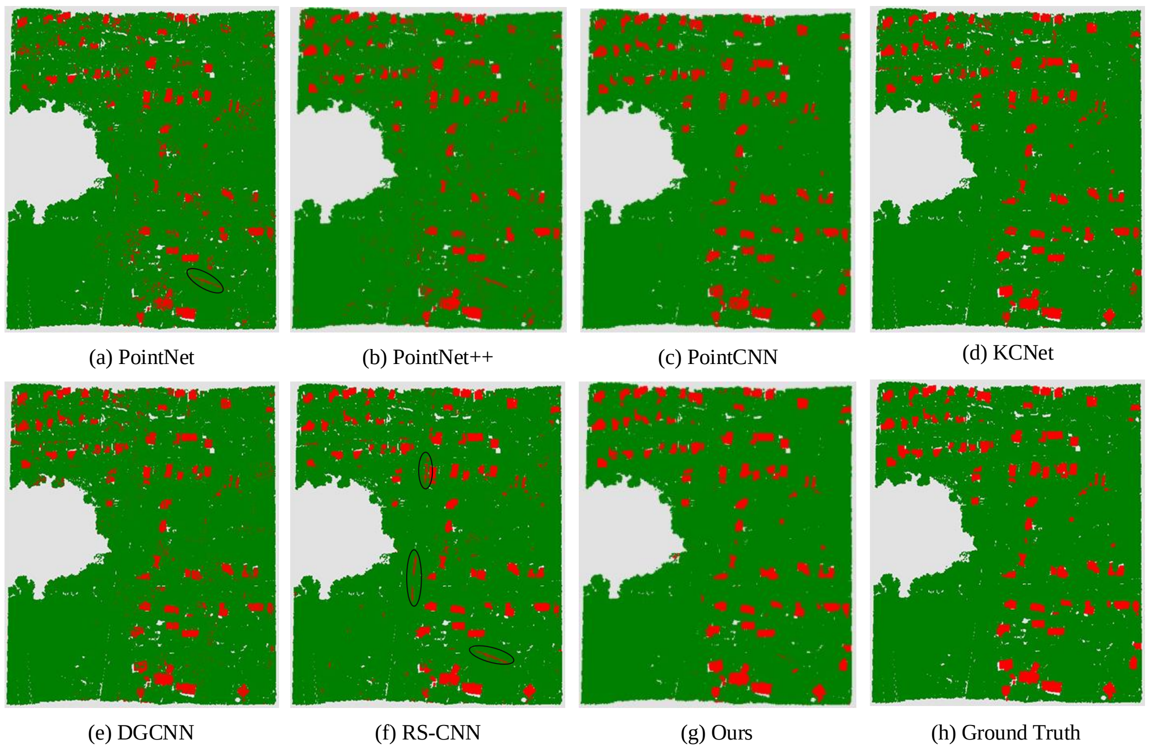

**Figure 9.** The visualization of overall extraction comparison results. The green colored points are the background (non-building) points, and the red colored points are the recognized or labeled building points. The black circles in (**a**) and (**d**) indicate the misrecognized building points from powerline points.

To compare the extraction results of these models in detail, we chose three typical buildings to represent the extraction difficulty in three levels, easy, normal and hard.

As shown in Figure 10, in the easy case, the building structure is simple, and the surrounding environment is clear (only flat grass). Our model completely recognized all the building points and separated them from the grass points clearly. The other models failed to recognize part of the building points.

As shown in Figure 11, in the normal case, two buildings with different sizes and heights are combined, and they are surrounded by tall trees. Although it is much harder than the easy case, our model also completely recognized all the building points, but misrecognized three tree points as the building points. Similarly, the performance of our model was obviously better than the others.

As shown in Figure 12, in the hard case, the building is a multi-story building with irregular rooftops, which has more complex structure than the former two cases. The PointCNN achieved relatively good performance, which recognized more than half of the building points. Our model completely recognized the main rooftop and one side rooftop, but only few building points of the other side rooftop with chimney were misrecognized. As for the other models, only some cracked pieces were recognized, which recognized less than half of the building points.

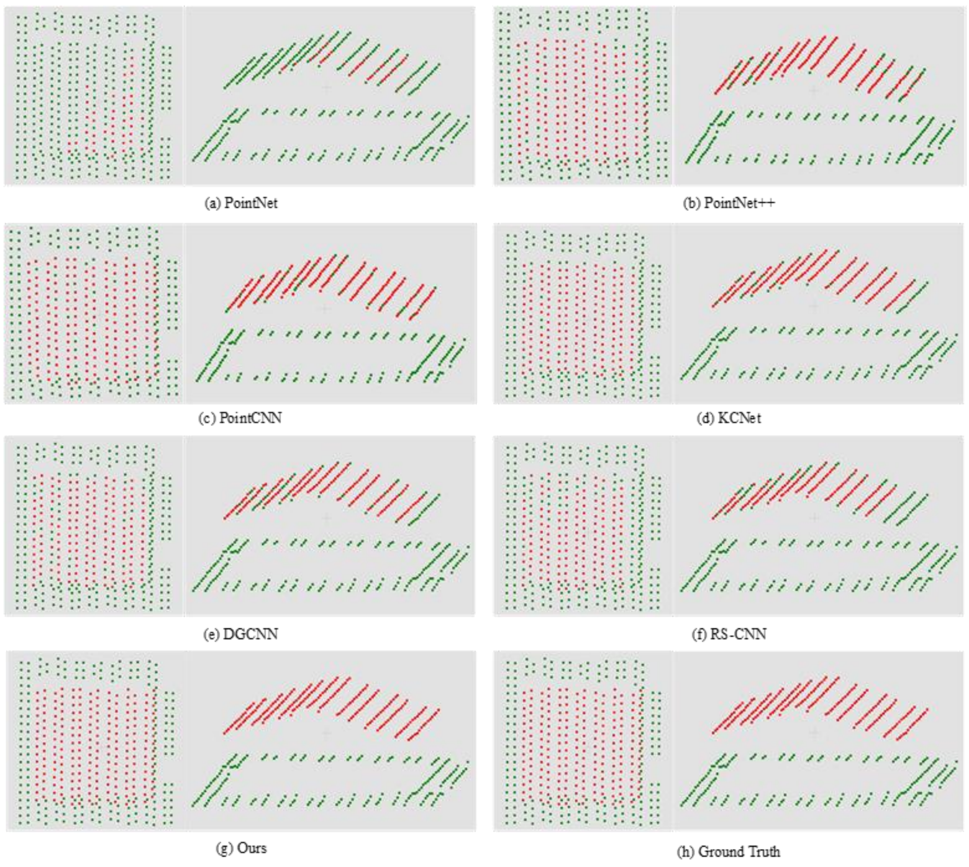

**Figure 10.** The details visualization of easy level comparison results. For each model, the **left** image is the vertical view, and the **right** image is the side view.

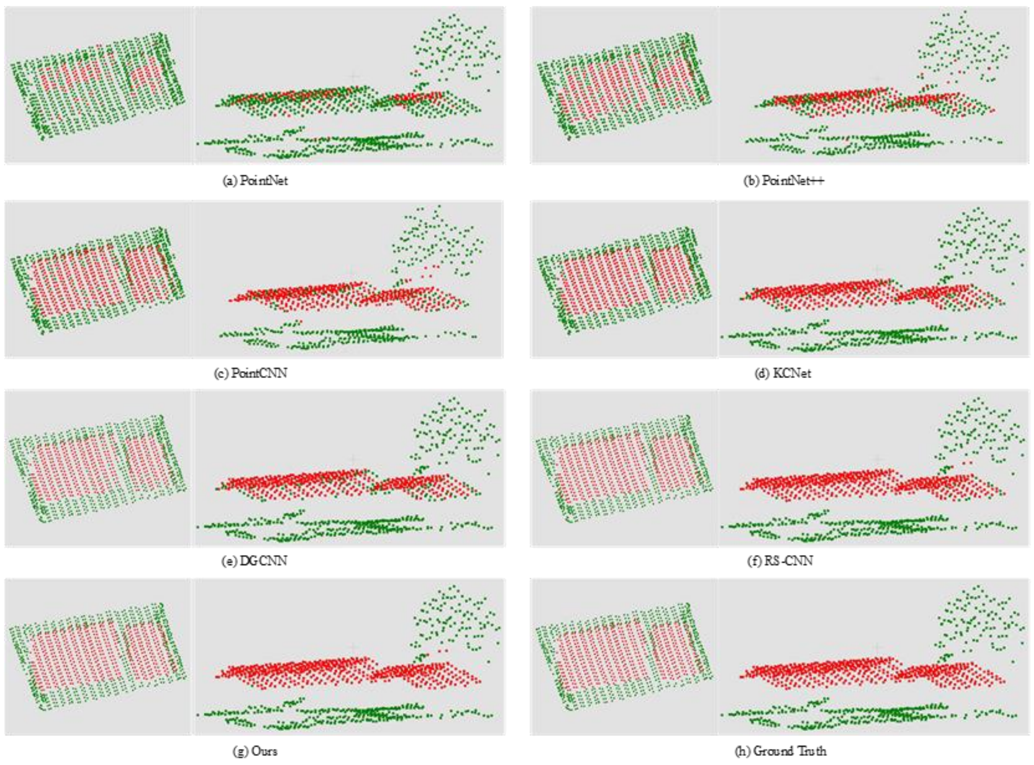

**Figure 11.** The details visualization of normal level comparison results. For each model, the **left** image is the vertical view, and the **right** image is the side view.

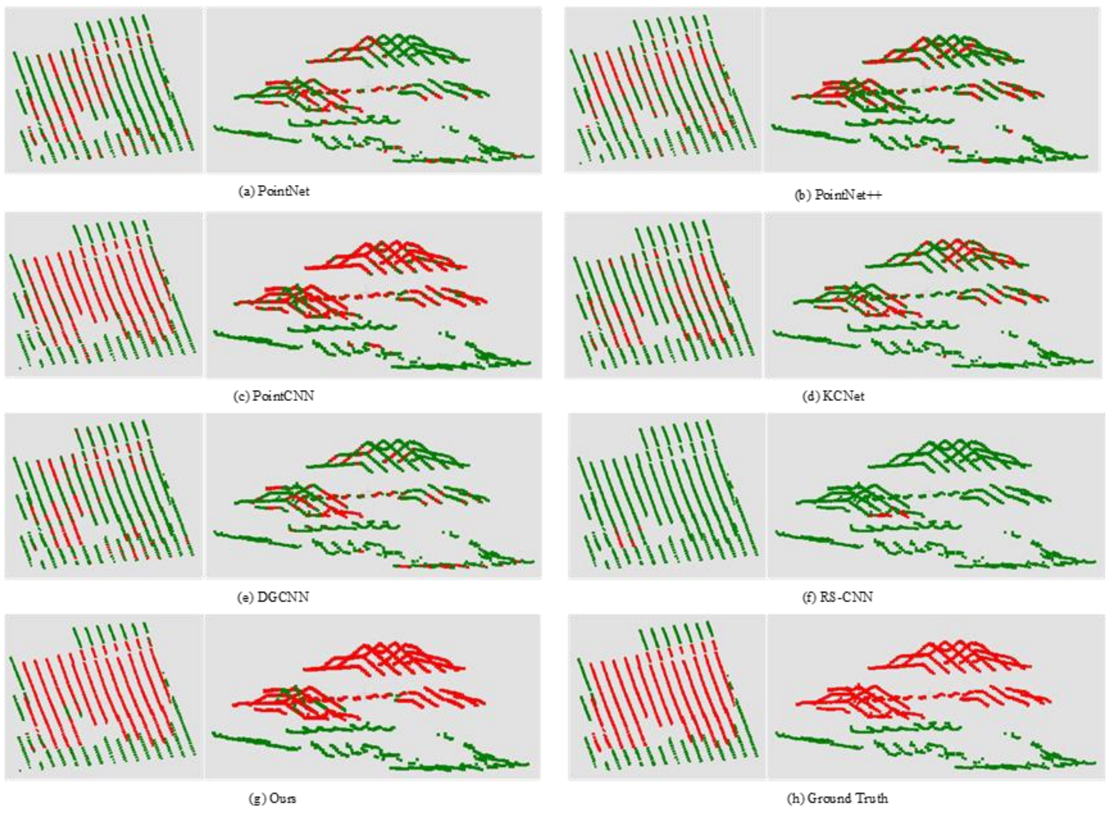

**Figure 12.** The details visualization of hard level comparison results. For each model, the **left** image is the vertical view, and the **right** image is the side view.

To better show the competitive extraction results of the proposed method, we specially chose several complicated scenes. As show in Figure 13, case (a) is a small and low building nearby the tall trees, many approaches might misrecognize it as the background points or concatenate the nearby tree points as a building. In the case (b), the building is surrounded by the tall trees, from another view (c), we can find it is also at the bottom of a small hill. Apparently, in this case, the building points and ground points cannot be separated by the height like many approaches did. From the margin side view, as case (d) shows, the buildings are spread on the uneven surface and surrounding by the dense forest, which dramatically increase the difficulty for building extraction. For all of these cases, as shown in Figure 12, the buildings are extremely difficult extracted with the existing approaches, but the proposed method handle these thorny cases well.

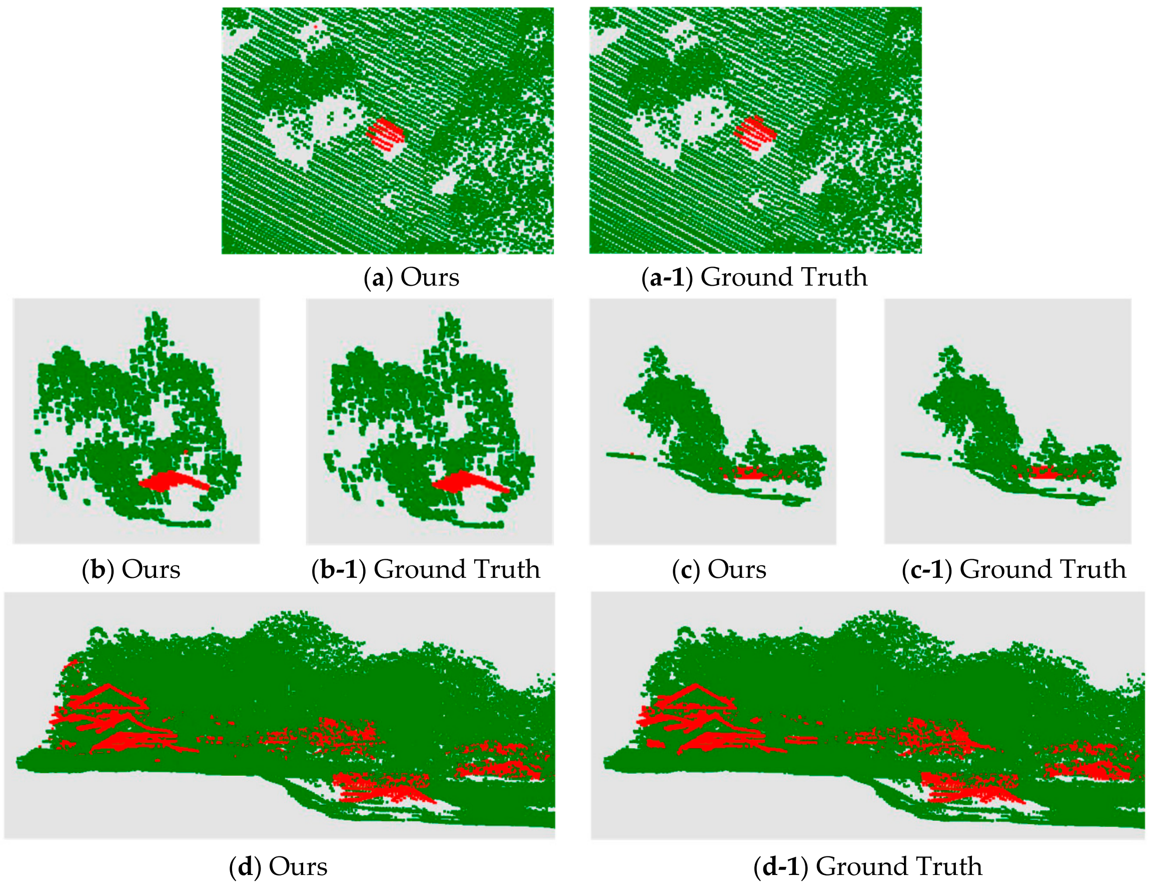

**Figure 13.** The visualization of complicated scenes.

## 6. Discussion

Building extraction is a basic task to the applications of land use survey and population change evaluation. According to the statistics in [2], 40% building extraction methods utilizes the additional data as the compensation of LiDAR data. Although the LiDAR data have been long-term considered as a promising data source, which contain the precious geometric information of the real world, its potential is still not fully realized by the existing methods. The newly emerged multi-spectral LiDAR data provide rich spectral information without the regular data fusion, which inspired us trying to explore the extraction effects with this new data.

As we mentioned before, those common drawbacks of the traditional building extraction approaches (classical machine learning-based methods) might fail to extract buildings from LiDAR data. We think these problems or drawbacks are basically coming from two stages, feature extraction and feature classification. For early studies, because there is no good feature extractor or encoder, they have to extract features manually, which limits the distinguishability and representability of the extracted features. Apparently, without a "good" feature, there is no "good" extraction result. Moreover, the traditional classifiers, which are based on classical machine learning methods, are suitable for small data sets and low-level features, have difficulty handling the large data volume and high-level features. Additionally, the traditional classifiers need to specify the parameters, the researchers are always beset by which classifier should be used and how to adjust its parameters. With the powerful inference ability of CNNs, the deep-learning-based methods automatically learn the high-level features from low-level features, like original coordinates and spectral values, and also learn the suitable or useful features automatically. Besides, the deep learning-based methods could be used as a classifier naturally, and the parameters contained in the network are adjusted by the optimizer automatically.

Compared with the traditional approaches, the deep learning-based methods could effectively solve the main drawbacks without extra assumptions or limitations.

Although the deep-learning-based methods have obvious advantages for building extraction tasks, the existing approaches and frameworks still have room for improvement. Due to the unstructured properties of point clouds, the characteristics of point clouds in sparsity, permutation invariance, and transformation invariance, are the thorny problems for standard convolution implementations [54]. In previous studies, many researchers transformed the point cloud data into multi-view projected images or voxel grids before feeding them to a standard convolutional neural network. Few researchers separated the whole scene into many cuboid regional subsets, and utilized the down-sampling and up-sampling techniques to meet the data form requirement of the standard convolutional neural networks. However, the number of points in unit area is not fixed and the sampling techniques damage the scene integrity, which cannot ensure that every point in the original scene could be labeled. Unlike the existing deep-learning-based point classification frameworks, whose main purpose is to evaluate the accuracy of the model, the building extraction is a practical task, in which every point in the test scenes needs to be predicted for the future real-world applications. That motivated us to develop a new framework. Fortunately, we eventually address these issues by introducing the FPS-KNN sample generation method into the framework. Obviously, the proposed framework also could be applied to the other similar practical tasks, e.g., land cover classification. The limitations of the proposed framework mainly reflect in two ways: the hardship of manually labeling sample and the requirement of massive training data. However, these are the common drawbacks of deep-learning-based frameworks or methods, and there are still no good solutions so far.

Compared with the other state-of-the-art networks, we introduced the geometric moments into our model to extract the local geometric feature more efficiently. Because the special geometry of the buildings or rooftops, e.g., they are normally composited by a set of planar faces, the geometric moments representation contributes to distinguish the building points from the background points significantly. Although the other general models may achieve higher accuracies on benchmarks, our designated model achieves better performance on building extraction tasks. Through the experiments in Section 5.4, the accuracies and visualization results demonstrated the effectiveness and efficiency of the proposed framework and methods. It is worth mentioning that the test scenes we used are more complicated than the commonly used urban areas, which dramatically increase the difficulty for building extraction tasks. The point-based evaluation we used has higher resolution, which means the stricter evaluation way, compared with pixel-based and object-based evaluations.

Besides, we specially designed the experiments in Section 5.3.1. On the one hand, we want to investigate and evaluate the effect of additional spectral information. On the other hand, we want to confirm the universality of our model and framework for its future potential extensions. The results confirm our speculations. This motives us to explore the combination of the other additional information, e.g., normal vectors, in our future work.

Since the large volume of point clouds, the inefficient massive data processing of the existing methods has been the obstruction of practical applications. Compared with the classical machine learning based methods, the deep-learning-based methods show significant improvement in this respect, but still limited. For example, with the limitation of GPU memory, by using our model, we can only set the maximum sample size and batch size as 4096 and 16, respectively. It is far from the requirement of building extraction from large scale scenes. In addition, the experiment we designed in Section 5.3.2 demonstrated that the larger sample size contributes to the better performance. Thus, how to improve the architecture of our model with larger sample size to handle the large-scale building extraction tasks would be our next work.

## 7. Conclusions

In this paper, we proposed a novel deep-learning-based framework for building extraction from multi-spectral point cloud data. Meanwhile, a sample generation method, a convolution operator and a convolutional neural network implemented in the framework were proposed. The proposed framework provided a novel architecture for the better application of deep learning methods in this research field. Besides, with the characteristic of good universality, theoretically, the proposed framework could handle any point sets and be implemented in any networks, which could greatly promote the practical applications of the proposed framework. As for the point-based evaluation we used in this paper, obviously, it is more difficult to achieve the same accuracy compared with the traditionally used pixel-based and object-based evaluation. However, it has higher resolution and reflects the direct connection with the real world, which is of greater practical significance. Compared with the other state-of-the-art networks, our method achieved the best comprehensive performance with regard to the four metrics. In addition, the corresponding visualization results showed the strong capacity of our model, especially for the difficult cases such as the buildings surrounded by tall trees and the multi-story buildings with complex structure rooftops, our model still achieved outstanding performance than the others. In future work, we will test the influence of adding the other additional features to our method and try to process the larger area scenes by using our method in our framework.

**Author Contributions:** Methodology, D.L. (Dilong Li); Validation, D.L. (Dilong Li). and X.S.; Writing—original draft preparation, D.L. (Dilong Li); Writing—review and editing, Y.Y., G.Z., and D.L. (Deren Li); resources, H.G. and J.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported in part by the National Natural Science Foundation of China under Grants 41671454 and 41971414, and in part by the Six Talent Peaks Project in Jiangsu Province under Grant XYDXX-098.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Khoshelham, K.; Nardinocchi, C.; Frontoni, E.; Mancini, A.; Zingaretti, P. Performance evaluation of automated approaches to building detection in multi-source aerial data. *ISPRS J. Photogramm. Remote Sens.* **2010**, *65*, 123–133. [CrossRef]
2. Tomljenovic, I.; Höfle, B.; Tiede, D.; Blaschke, T. Building extraction from airborne laser scanning data: An analysis of the state of the art. *Remote Sens.* **2015**, *7*, 3826–3862. [CrossRef]
3. Groger, G.; Plumer, L. CityGML—Interoperable semantic 3D city models. *ISPRS J. Photogramm. Remote Sens.* **2012**, *71*, 12–33. [CrossRef]
4. Khoshelham, K.; Oude Elberink, S.; Xu, S. Segment-Based classification of damaged building roofs in aerial laser scanning data. *IEEE Geosci. Remote Sens. Lett.* **2013**, *10*, 1258–1262. [CrossRef]
5. Sirmacek, B.; Taubenboeck, H.; Reinartz, P.; Ehlers, M. Performance evaluation for 3-D city model generation of six different DSMs from air- and spaceborne sensors. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **2012**, *5*, 59–70. [CrossRef]
6. Awrangjeb, M.; Ravanbakhsh, M.; Fraser, C.S. Automatic building detection using LiDAR data and multispectral imagery. In Proceedings of the 2010 International Conference on Digital Image Computing: Techniques and Applications, Sydney, NSW, Australia, 1–3 December 2010; pp. 45–51.
7. Awrangjeb, M.; Zhang, C.; Fraser, C.S. Automatic reconstruction of building roofs through effective integration of LiDAR and multispectral imagery. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *1*, 203–208. [CrossRef]
8. Pan, S.; Guan, H.; Yu, Y. A comparative land-cover classification feature study of learning algorithms: DBM, PCA, and RF using multispectral LiDAR data. *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **2019**, *12*, 1314–1326. [CrossRef]
9. Zhang, J.; Lin, X.; Ning, X. SVM-Based classification of segmented airborne LiDAR point clouds in urban areas. *Remote Sens.* **2013**, *5*, 3749–3775. [CrossRef]
10. Verdie, Y.; Lafarge, F.; Alliez, P. LOD generation for urban scenes. *ACM Trans. Graph.* **2015**, *34*, 30. [CrossRef]
11. Peter, D.; Norbert, P. A comprehensive automated 3D approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors* **2008**, *8*, 7323–7343.
12. Sampath, A.; Shan, J. Segmentation and reconstruction of polyhedral building roofs from aerial LiDAR point clouds. *IEEE Trans. Geosci. Remote Sens.* **2010**, *48*, 1554–1567. [CrossRef]

13. Ural, S.; Shan, J. A min-cut based filter for airborne LiDAR data. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* **2016**, *41*, 395–401. [CrossRef]
14. Maltezos, E.; Doulamis, A.; Doulamis, N.; Ioannidis, C. Building extraction from LiDAR data applying deep convolutional neural networks. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 155–159. [CrossRef]
15. Bittner, K.; Cui, S.; Reinartz, P. Building extraction from remote sensing data using fully convolutional networks. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. ISPRS Arch.* **2017**, *42*, 481–486. [CrossRef]
16. Hamed, N.F.; Shafri, H.Z.M.; Ibrahim, S.M.; Pradhan, B.; Mansor, S. Deep learning approach for building detection using LiDAR-Orthophoto fusion. *J. Sens.* **2018**, *2018*, 1–12.
17. Sohn, G.; Huang, X.; Tao, V. Using a binary space partitioning tree for reconstructing polyhedral building models from airborne LiDAR data. *Photogramm. Eng. Remote Sens.* **2008**, *74*, 1425–1440. [CrossRef]
18. Zhou, G.; Song, C.; Simmers, J. Urban 3D GIS from LiDAR and digital aerial images. *Comput. Geosci.* **2004**, *30*, 345–353. [CrossRef]
19. Maas, H.G.; Vosselman, G. Two algorithms for extracting building models from raw laser altimetry data. *ISPRS J. Photogramm. Remote Sens.* **1999**, *54*, 153–163. [CrossRef]
20. Zhou, Q.; Neumann, U. Fast and extensible building modeling from airborne LiDAR data. In Proceedings of the International Conference on Advances in Geographic Information Systems, Irvine, CA, USA, 5–7 November 2008; p. 7.
21. Poullis, C.; You, S. Photorealistic large-scale urban city model reconstruction. *IEEE Trans. Vis. Comput. Graph* **2009**, *15*, 654–669. [CrossRef]
22. Zou, X.; Feng, Y.; Li, H. An adaptive strips method for extraction buildings from light detection and ranging data. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 1–5. [CrossRef]
23. Santos, R.C.D.; Galo, M.; Carrilho, A.C. Extraction of building roof boundaries from LiDAR data using an adaptive alpha-shape algorithm. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 1–5. [CrossRef]
24. Awrangjeb, M.; Zhang, C.; Fraser, C.S. Automatic extraction of building roofs using LIDAR data and multispectral imagery. *ISPRS J. Photogramm. Remote Sens.* **2013**, *83*, 1–18. [CrossRef]
25. Awrangje, M.; Fraser, C.S. Rule-based segmentation of LIDAR point cloud for automatic extraction of building roof planes. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2013**, *2*, 1–6.
26. Mohammad, A.; Clive, F. Automatic segmentation of raw LIDAR data for extraction of building roofs. *Remote Sens.* **2014**, *6*, 3716–3751.
27. Syed, G.; Mohammad, A.; Guojun, L. An automatic building extraction and regularisation technique using LiDAR point cloud data and orthoimage. *Remote Sens.* **2016**, *8*, 27.
28. Sohn, G.; Dowman, I. Data fusion of high-resolution stellite imagery and LiDAR data for automatic building extraction. *ISPRS J. Photogramm. Remote Sens.* **2007**, *62*, 43–63. [CrossRef]
29. Nguyen, T.H.; Daniel, S.; Guériot, D.; Sintès, C.; Le Caillec, J.-M. Super-Resolution-Based Snake Model—An Unsupervised Method for Large-Scale Building Extraction Using Airborne LiDAR Data and Optical Image. *Remote Sens.* **2020**, *12*, 1702. [CrossRef]
30. Briese, C.; Pfennigbauer, M.; Lehner, H.; Ullrich, A.; Wagner, W.; Pfeifer, N. Radiometric calibration of multi-wavelength airborne laser scanning data. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *7*, 335–340. [CrossRef]
31. Pan, S.; Guan, H.; Chen, Y.; Yu, Y.; Gonçalves, W.N.; Junior, J.M.; Li, J. Land-cover classification of multispectral LiDAR data using CNN with optimized hyper-parameters. *ISPRS J. Photogramm. Remote Sens.* **2020**, *166*, 241–254. [CrossRef]
32. Weinmann, M.; Jutzi, B.; Hinz, S.; Mallet, C. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS J. Photogramm. Remote Sens.* **2015**, *105*, 286–304. [CrossRef]
33. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In Proceedings of the Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 11108–11117.
34. Hu, M. Visual pattern recognition by moment invariants. *IEEE Trans. Inf. Theory* **1962**, *8*, 179–187.
35. Belkasim, S.O.; Shridhar, M.; Ahmadi, M. Pattern recognition with moment invariants: A comparative study and new results. *Pattern Recognit.* **1991**, *24*, 1117–1138. [CrossRef]
36. Flusser, J.; Suk, T. Pattern recognition by affine moment invariants. *Pattern Recognit.* **1993**, *26*, 167–174. [CrossRef]

37. Luo, L.M.; Hamitouche, C.; Dillenseger, J.L.; Coatrieux, J.L. A moment based three-dimensional edge operator. *IEEE Trans. Biomed. Eng.* **1993**, *40*, 693–703. [CrossRef]
38. Liu, S.T.; Tsai, W.H. Moment-preserving corner detection. *Pattern Recognit.* **1990**, *23*, 441–446. [CrossRef]
39. Yokoya, N.; Levine, M.D. Range image segmentation based on differential geometry: A hybrid approach. *IEEE Trans. Pattern Anal. Mach. Intell.* **1989**, *11*, 643–649. [CrossRef]
40. Tuceryan, M. Moment-based texture segmentation. *Pattern Recognit. Lett.* **1994**, *15*, 659–668. [CrossRef]
41. Teh, C.H.; Chin, R.T. On image analysis by the methods of moments. *IEEE Trans. Pattern Anal. Mach. Intell.* **1988**, *10*, 496–513. [CrossRef]
42. Teague, M.R. Image analysis via the general theory of moments. *J. Opt. Soc. Am.* **1980**, *70*, 920–930. [CrossRef]
43. Lo, C.H.; Don, H.S. 3-D moment forms: Their construction and application to object identification and positioning. *IEEE Trans. Pattern Anal. Mach. Intell.* **1989**, *11*, 1053–1064. [CrossRef]
44. Abu-Mostafa, Y.S.; Psaltis, D. Recognitive aspects of moment invariants. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, *6*, 698–706. [CrossRef]
45. Joseph-Rivlin, M.; Zvirin, A.; Kimmel, R. *Mo-Net: Flavor the Moments in Learning to Classify Shapes*; Computer Vision Foundation: New York, NY, USA, 2018.
46. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep learning on point sets for 3D classification and segmentation. In Proceedings of the Computer Vision and Pattern Recognition, Honolulu, HI, USA, 22–25 July 2017; pp. 652–660.
47. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw. Learn. Syst.* **2009**, *20*, 61–80. [CrossRef] [PubMed]
48. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning Convolutional Neural Networks for Graphs. *arXiv* **2016**, arXiv:1605.05273.
49. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.* **2018**, *38*, 146. [CrossRef]
50. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv* **2017**, arXiv:1706.02413.
51. Liu, Y.; Fan, B.; Xiang, S.; Pan, C. Relation-shape convolutional neural network for point cloud analysis. In Proceedings of the Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 1–10.
52. Wang, L.; Huang, Y.; Hou, Y.; Zhang, S.; Shan, J. Graph attention convolution for point cloud semantic segmentation. In Proceedings of the Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 10296–10305.
53. Shen, Y.; Feng, C.; Yang, Y.; Tian, D. Mining point cloud local structures by kernel correlation and graph pooling. In Proceedings of the Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4548–4557.
54. Li, D.; Shen, X.; Yu, Y.; Guan, H.; Wang, H.; Li, D. GGM-Net: Graph Geometric Moments Convolution Neural Network for Point Cloud Shape Classification. *IEEE Access.* **2020**, *8*, 124989–124998. [CrossRef]