Continuous Conditional Random Field Convolution for Point Cloud Segmentation

Fei Yang, Franck Davoine, Huan Wang, Zhong Jin

Continuous Conditional Random Field Convolution for Point Cloud Segmentation

Fei Yang[a,b], Franck Davoine[c], Huan Wang[b], Zhong Jin[a,b,*]

[a]*Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing 210094, China*
[b]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[c]*Alliance Sorbonne Université, Université de technologie de Compiègne, CNRS, Heudiasyc Lab., CS 60319 - F-60203 Compiègne Cedex, France*

Abstract

Point cloud segmentation is the foundation of 3D environmental perception for modern intelligent systems. To solve this problem and image segmentation, conditional random fields (CRFs) are usually formulated as discrete models in label space to encourage label consistency, which is actually a kind of postprocessing. In this paper, we reconsider the CRF in feature space for point cloud segmentation because it can capture the structure of features well to improve the representation ability of features rather than simply smoothing. Therefore, we first model the point cloud features with a continuous quadratic energy model and formulate its solution process as a message-passing graph convolution, by which it can be easily integrated into a deep network. We theoretically demonstrate that the message passing in the graph convolution is equivalent to the mean-field approximation of a continuous CRF model. Furthermore, we build an encoder-decoder network based on the proposed continuous CRF graph convolution (CRFConv), in which the CRFConv embedded in the decoding layers can restore the details of high-level features that were lost in the encoding stage to enhance the location ability of the network, thereby benefiting segmentation. Analogous to the CRFConv, we show that the classical discrete CRF can also work collaboratively with the proposed network via another graph convolution to further improve the segmentation results. Experiments on various point cloud benchmarks demonstrate the effectiveness and robustness of the proposed method. Compared with the state-of-the-art methods, the proposed method can also achieve competitive segmentation performance.

Keywords: point cloud segmentation, conditional random fields, message passing, graph convolution, mean-field approximation

1. Introduction

A conditional random field (CRF) is a kind of probabilistic graphical model (PGM) that is widely employed for structure prediction problems in computer vision. In image segmentation, most previous studies have attempted to model the data affinity in label space with CRFs, where the CRF is formulated as a discrete model. Such a discrete model is usually NP-hard to solve. Nevertheless, some approximate methods have been proposed to solve discrete CRF models efficiently. The most widely used approximate methods in computer vision are α-expansion/$\alpha\beta$-swap[1] and mean-field approximation[2]. The former is a graph-cut-based method, and the latter belongs to the variational inference-based method. With the rapid development of deep learning techniques, mean-field approximation methods have attracted more attention in the community because the mean-field approximation can be easily embedded in a deep network to build an end-to-end architecture[3]. Despite these, discrete CRFs that encourage label consistency in the results can only smooth the object boundaries and remove some small noises. The misclassified object regions in the prediction stage are hard to correct by these kinds of postprocessing CRFs.

*Corresponding author
Email addresses: yangfei92516@163.com (Fei Yang), franck.davoine@hds.utc.fr (Franck Davoine), wanghuanphd@njust.edu.cn (Huan Wang), zhongjin@njust.edu.cn (Zhong Jin)

Convolutional neural networks (CNNs) have achieved significant success in image, speech, and natural language processing (NLP). However, with the explosion in 3D data such as point clouds, traditional CNNs that are designed for grid-like data can no longer meet all the needs. On the one hand, the point cloud data acquired by 3D sensors are usually large-scale, sparse, and unordered, which is difficult to process with traditional CNNs directly. On the other hand, point cloud segmentation is the foundation of 3D environmental perception in modern intelligent systems, such as intelligent robotics and autonomous driving cars. Therefore, segmenting point clouds with deep learning techniques is an urgent challenge in modern computer vision. Although many deep learning-based methods[4, 5, 6] have been proposed to segment point clouds, they focus on the design of the feature extractor to improve the representation ability of the point features. Segmentation consists of two parts: classifying (*what it is*) and locating (*where it is*). For classification tasks, we only need to encode the high-level features, and focus less on where the extracted features should be. The classification result can be inferred from only a single global vector. However, segmentation is a dense prediction task. We need to not only extract the high-level features (*what it is*) but also restore the correspondence between the encoded high-level features and each input point exactly (*where it is*). The restoration process can be understood as solving the problem of *where it is*. In other words, previous studies attempted to make the network understand *what it is* but failed to investigate the problem of *where it is*. Currently, the research on feature decoding is scarce in the literature. To fill this gap, we focus our study on feature decoding rather than feature encoding, considering that the feature encoding methods targeting a classification task are competent for a segmentation task.

To address the problems mentioned above, we reconsider the CRF in feature space to enhance the location ability of the network during feature decoding. Segmentation is a typical structure prediction problem. To address this problem, the CRF is usually formulated as a discrete model to encourage label consistency in the results, which can only smooth the object boundaries and remove some small noises. Inspired by the continuous CRF used for depth completion, we reconsider the CRF as a continuous model in feature space to restore the encoding features for point cloud segmentation. The CRF is enforced to participate in the feature extraction process of the point cloud and thus can fundamentally improve the representation ability of the features rather than just smoothing the labels by postprocessing. Specifically, previous methods usually adopt the kNN (k-nearest neighbors)-based interpolation method to decode high-level features. They roughly place the high-level features at their closest positions during upsampling but ignore the consistency of features in feature space. The details of the high-level features will be lost in such an upsampling process. There is an intuitive assumption that similar points in low-level feature space should also have similar high-level features. Therefore, we employ the CRF model to guide the feature upsampling procedure to restore the details of high-level input point features gradually. In our continuous CRF model, the low-level features in the encoding layers are used as guidance information to guide the high-level feature upsampling. This process has a similar effect to guided filtering, such as joint bilateral filters.

In this paper, we propose a continuous CRF graph convolution (CRFConv) to capture the feature structures during the feature extraction stage. More precisely, we propose using a continuous quadratic energy model to describe the point cloud in feature space, in that a continuous CRF model is mathematically equivalent to a quadric energy model that has a similar form to the diffusion model. Inspired by some diffusion-based graph convolution methods in the community, we formulate the solution process of this model as a message-passing graph convolution, by which the continuous CRF can be embedded in the network for feature extraction. From a CRF perspective, we also prove that the message passing of this graph convolution is equivalent to the mean-field approximation algorithm of a continuous Gaussian CRF model, which indicates that the proposed graph convolution is actually equivalent to a continuous CRF model. After that, we build an encoder-decoder network for point cloud segmentation, in which the decoding modules are carefully designed based on the proposed CRFConv. We use the proposed CRFConv to enhance the location ability (*where it is*) of the network as described above. To enlarge the receptive field, the extracted high-level features in the encoder lose details by successive pooling/downsampling operations. Previous methods only upsample the high-level features with simple linear interpolation in the decoding layers, while our method models the upsampling process with a structure model (i.e., the CRFConv), by which the details of the high-level point cloud features can be restored gradually. Furthermore, we show that our continuous CRF can also work collaboratively with the classical discrete CRF to further improve the segmentation results. The discrete CRF is formulated as another message-passing graph convolution by analogy to the CRFConv, by which the discrete CRF can be embedded in our segmentation network to implement an end-to-end dual CRF network. To the best of our knowledge, we are the first to consider CRF models both in feature and label space simultaneously in point cloud segmentation.

Experiments on various datasets, including object-level and scene-level segmentation tasks, demonstrate the ef-

fectiveness of the proposed method. The contributions of this paper can be summarized as follows:[1]

1. We propose a continuous CRF graph convolution (CRFConv) to model the data affinity in feature space instead of label space, which enforces the CRFConv to participate in the point cloud feature extraction process. The CRFConv can thus fundamentally improve the representation ability of the features rather than just smoothing the labels by postprocessing.
2. We design a point cloud segmentation network based on the proposed CRF convolution to enhance the location ability of the network. The CRFConv is embedded in the network decoder to model the upsampling process with a continuous CRF model, by which the details of the high-level point cloud features can be restored gradually during upsampling.
3. We further implement a dual CRF network by reformulating the classical discrete CRF as another graph convolution. The dual CRF network models the data affinity both in feature space and label space simultaneously to further improve the segmentation results.
4. We conduct experiments on various challenging point cloud segmentation benchmarks, including synthetic object data and large-scale indoor and outdoor scenes. The experimental results demonstrate the effectiveness and robustness of the proposed method.

The rest of this paper is organized as follows. In section 2, we review the literature and introduce some related basic knowledge about CRF. The continuous CRF graph convolution and theoretical analyses are presented in section 3. In section 4, we present the architecture of the segmentation network in detail and show how to combine the continuous CRF with the classical discrete CRF in our segmentation network. The experimental results, comparisons, and ablation analysis can be found in section 5. Section 6 concludes the work in this paper, in which the limitations and future works are also discussed.

2. Related Work

2.1. Point cloud segmentation

With the popularity of 3D sensors, 3D data are exploding in human daily life owing to their easy acquisition. One key application of 3D data is point cloud segmentation, which is the foundation of 3D environmental perception. Considering the success of CNNs in image processing, many researchers have attempted to employ deep learning techniques to segment point clouds[7]. In this section, we only review the point cloud segmentation methods based on deep learning. These methods can be divided into four categories: image-based, voxel-based, point-based, and graph-based methods.

Image-based methods convert the point cloud as an image-like representation and employ some image-based networks to process the point cloud. For example, Yang et al.[8] projected LiDAR (light detection and ranging) points to the image plane by the cross-calibration parameters. Instead of converting the LiDAR point cloud to the image plane, Wu et al.[9] projected the LiDAR point cloud to a sphere plane to form a dense and grid-based representation. An alternative solution for 3D point cloud representation is the multiview-based method[10]. In other words, the image-based methods convert the 3D point cloud to a 2D grid-like representation to cater to image-based networks. The structural information of the point cloud will inevitably be lost during such conversions.

Voxel-based methods borrow ideas from image-based CNNs. In these methods, the point cloud is first split into uniform voxels and then processed by deep 3D networks. Maturana et al. proposed VoxNet[11], an architecture to segment point clouds by integrating a volumetric occupancy grid representation with a supervised 3D CNN. However, the voxel-based presentation for a point cloud is usually time consuming and space wasting. To address this problem, Octree[12] and Kd-Tree[13]-based approaches have been proposed to save computations by skipping convolution in empty space. Nevertheless, voxel-based representation methods still cause information loss. In addition, it is difficult to balance computations and accuracy. How to determine the voxel resolution for the point cloud, which varies in density, remains a problem.

[1] Code is available at https://github.com/yangfei1223/CRFConv.

Point-based methods overcome the shortcomings of image-based and voxel-based methods. The unordered and sparse points can be directly processed by point-based methods without transformations. PointNet[4] was the first work to directly process the points in point clouds with deep learning techniques. Since Qi et al. pioneered this field, many point-based methods have emerged in the community. For example, Qi et al. further extended their PointNet to PointNet++[14], which is a hierarchical version of PointNet. Li et al.[5] learned a χ-transformation to transform the point cloud to a latent and potential canonical order, by which the typical convolution operator can be applied on the χ-transformed features. [15, 16] defined the convolution on point cloud domains using Monte Carlo. Thomas et al.[6] proposed the kernel point convolution (KPConv) inspired by the convolution operator on regular domains. Recently, the attention mechanism was also explored to extract the point features for point cloud segmentation in [17] and [18].

Graph-based methods have also been extended to process point clouds with the advent of graph neural networks (GNNs) in recent years. In these methods, they assume that a latent graph exists in the point cloud (e.g., kNN or radius graph). Landrieu et al.[19] first presegmented the point cloud to superpoints using an unsupervised cluster algorithm. Then they built a superpoint graph based on the superpoint features extracted by PointNet to segment the point cloud. Wang et al.[20] proposed EdgeConv, which is dynamically computed in each layer of the network, for high-level tasks on point clouds. DeepGCNs[21] extended graph convolutional networks (GCNs) to deep models by borrowing concepts from CNNs, specifically residual/dense connections and dilated convolutions. DeepGCNs have also shown their ability on large-scale 3D point cloud segmentation tasks.

Point-based and graph-based methods can both be categorized as geometric deep learning (GDL)[22] methods. GDL has shown its advances in processing point clouds. In this paper, the proposed method is also a kind of GDL method. However, previous methods focus on improving the classification ability of the network but ignore its location ability, which is crucial for segmentation. Our work focuses on the decoding rather than the encoding of the point features to enhance the location ability of the network, which is the most different point from existing methods.

2.2. Conditional random field

In this section, we first review some basic knowledge about CRF for the convenience of readers who are not familiar with this field. Then, we review the applications of CRFs in computer vision, including but not limited to point cloud processing.

Basic knowledge. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ and $\mathcal{E}$ represent the node and edge sets of the graph, respectively. Denote x_i as a random variable defined on node $i \in \mathcal{V}$. $e(i, j) \in \mathcal{E}$ can be understood as the relationship between the random variables x_i and x_j. Thus, $X = \{x_i | i \in \mathcal{V}\}$ can be viewed as a random field on $\mathcal{G}$. If the random variables of X in $\mathcal{G}$ satisfy the Markov property, then we call X a Markov random field (MRF) on $\mathcal{G}$. According to the Hammersley-Clifford theorem, the joint probability distribution of the MRF can be presented with a Gibbs distribution parameterized by the clique potentials defined on $\mathcal{G}$, that is:

$$P(X) = \frac{1}{Z} \prod_{c \in \mathcal{C}_\mathcal{G}} \phi_c(X_c), \tag{1}$$

where c and X_c denote a maximum clique (complete subgraph) of $\mathcal{G}$ and its associated random variable, respectively. $\mathcal{C}_\mathcal{G}$ is the set of all cliques. ϕ_c is the clique potential defined on c. Z is a partition function to ensure a distribution. Note that ϕ_c is constrained to be nonnegative in the Gibbs distribution. In general, we can transform $\phi_c(X_c)$ to its log space. Specifically, we let $\phi_c(X_c) = \exp(-\psi_c(X_c))$, where $\psi_c(X_c) = -\log \phi_c(X_c)$ denotes the energy function defined on c. Hence the Gibbs distribution can be simply rewritten as:

$$P(X) = \frac{1}{Z} \exp\big(-E(X)\big), \tag{2}$$

where $E(X) = \sum_{c \in \mathcal{C}_G} \psi_c(X_c)$ is the Gibbs energy function defined by all cliques of $\mathcal{G}$. Eq. (2) is a log linear model, which ensures a positive distribution.

The MRF can also be used to describe a conditional distribution. Given another random variable Y, if the random variable X on $\mathcal{G}$ is conditional on Y, we say the MRF determined by X is a CRF determined by X and Y. The conditional distribution $P(X|Y)$ of the CRF can also be represented with a Gibbs distribution, which has the same form as Eq. (2). Y and X can be viewed as the observed and latent variables in the CRF, respectively. In other words,

the CRF of X and Y can be understood as the MRF of X with an additional input Y. For notational convenience, we omit the condition Y in the rest of the paper when we refer to CRF.

Applications. CRFs/MRFs are widely applied to image segmentation for structure prediction as a kind of postprocessing. Krähenbühl et al.[23] proposed the fully connected CRF for image segmentation for the first time. The fully connected CRF also acts as a postprocessing step of a CNN in [24] to improve the localization accuracy. In addition, Zheng et al.[25] extended the fully connected CRF as a recurrent neural network (RNN) for object segmentation. Qiu et al[26] also integrated the CRF into a CNN to segment saliency objects. Although these methods allow us to jointly train the CNN and the CRF model end-to-end, the CRF still acts as postprocessing in the pipeline. In contrast to the existing approaches that use discrete CRF models, Vemulapalli et al.[27] proposed using a Gaussian CRF model for semantic segmentation instead, in which the CRF is still built in the label space to refine the labels. In addition to image segmentation, the application of CRFs/MRFs for point cloud segmentation is also explored. Wu et al. adopt a CRF model as in [24] to refine the output of their SqueezeNet[9].

Our continuous CRF model is different from discrete CRFs that usually act as postprocessing to refine the labels. We model the data affinity in feature space with a continuous CRF that essentially participates in the feature extraction process of the point cloud. Hence, our CRF can fundamentally improve the representation ability of the features rather than only smoothing the labels in postprocessing.

3. The continuous CRF graph convolution

In this section, we present the CRF graph convolution from a continuous quadratic energy model in the feature space of a point cloud. First, we present how to formulate its solution process as a message-passing algorithm in a graph defined on the point cloud. Then we define the continuous graph convolution based on this message-passing algorithm. Finally, we conduct a theoretical analysis of the proposed graph convolution from a CRF perspective.

3.1. The continuous quadratic energy model

Denote $\mathcal{P} = \{(\mathbf{p}_i, \mathbf{z}_i, \mathbf{x}_i)|i < N\}$ as a point cloud with N points. $\mathbf{p}_i \in \mathbb{R}^3$ is the position vector of point i in the 3D Euclidean space. $\mathbf{z}_i \in \mathbb{R}^d$ and $\mathbf{x}_i \in \mathbb{R}^d$ represent the observed and latent features of point i, respectively. Assume that the observed feature $\mathbf{z}_i$ is independent of each other. We infer the latent feature $\mathbf{x}_i$ from the observed feature $\mathbf{z}_i$ by considering the affinity in the feature space. Thus, we consider the following quadratic energy function:

$$E(X) = \underbrace{\sum_i (\mathbf{x}_i - \mathbf{z}_i)^\top (\mathbf{x}_i - \mathbf{z}_i)}_{\text{fidelity term}} + \underbrace{\sum_{ij} (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{w}_{ij} (\mathbf{x}_i - \mathbf{x}_j)}_{\text{smoothness term}}, \tag{3}$$

where $X = \{\mathbf{x}_i|i < N\}$ denotes the feature set of the latent features. $\mathbf{w}_{ij} \in \mathbb{R}^{d\times d}$ is the weight matrix between features $\mathbf{x}_i$ and $\mathbf{x}_j$, which encodes the relationship between two points. It can be computed as:

$$\mathbf{w}_{ij} = s_{ij}\mathbf{C}, \tag{4}$$

where $s_{ij} \geq 0$ is the scalar similarity between points i and j, which can be estimated from the observed feature $\mathbf{z}$.[2] $\mathbf{C} \in \mathbb{R}^{d\times d} \succ \mathbf{0}$ is a compatibility matrix to model the intrarelationship between feature channels and is assumed to be independent of the feature value. There is a similar concept of the compatibility matrix in the classic discrete CRF model. In that case, $\mathbf{C}$ describes the co-occurrence (or compatibility) of different object classes or labels. Since our model is built in feature space, $\mathbf{C}$ actually describes the channel/dimension compatibility of feature space here. In general, we can assume that the channels are independent of each other, and the messages pass in each channel individually in this case. Therefore, it is possible to directly define $\mathbf{C}$ as the identity matrix. Alternatively, it is also possible to automatically learn it from the data to further enhance the expression ability of the model. We cannot parameterize $\mathbf{C}$ directly because of its positive-define constraint. In this paper, we let $\mathbf{C} = \mathbf{c}^\top\mathbf{c} + \epsilon\mathbf{I}$ to ensure positive definition, where $\mathbf{c}$ is a learnable parameter initialized as $\mathbf{c} = \mathbf{I}$. ϵ is a tiny positive value.

[2] s_{ij} is task-relevant and can be learned from data. We specify its form in the next section.

Eq. (3) consists of two terms: a fidelity term and a smoothness term. It can be understood that we want the new representation $\mathbf{x}_i$ to be smoother but as close as possible to its original representation $\mathbf{z}_i$. Eq. (3) and its variants have been widely applied in the problems of denoising, recovery, and superresolution in computer vision. The latent feature X can be obtained by minimizing Eq. (3):

$$X^* = \underset{X}{argmin}\, E(X). \tag{5}$$

Eq. (5) is a quadratic convex optimization problem with a closed-form solution. Denote $\mathbf{Z} = [\mathbf{z}_1^\top, \mathbf{z}_2^\top, ..., \mathbf{z}_N^\top]^\top$ and $\mathbf{X} = [\mathbf{x}_1^\top, \mathbf{x}_2^\top, ..., \mathbf{x}_N^\top]^\top$ as the concatenated feature vector of the point cloud, where $\mathbf{Z}, \mathbf{X} \in \mathbb{R}^{(N\times d)}$. The compact form of Eq. (3) can be rewritten as:

$$E(\mathbf{X}) = (\mathbf{X} - \mathbf{Z})^\top(\mathbf{X} - \mathbf{Z}) + \mathbf{X}^\top(\mathbf{D} - \mathbf{W})\mathbf{X}, \tag{6}$$

where $\mathbf{W}$ is a $N \times N$ block matrix consisting of $\mathbf{w}_{ij}$, which can be computed as $\mathbf{W}(i, j) = \mathbf{w}_{ij}$. $\mathbf{D}$ is a $N \times N$ block diagonal matrix that can be computed as $\mathbf{D}(i, i) = \sum_j \mathbf{w}_{ij}$. The derivative of $E(\mathbf{X})$ is calculated with respect to $\mathbf{X}$ and $\partial E(\mathbf{X})/\partial \mathbf{X} = \mathbf{0}$. We can obtain the closed-form solution of $E(\mathbf{X})$:

$$\mathbf{X}^* = (\mathbf{I} + \mathbf{D} - \mathbf{W})^{-1}\mathbf{Z}, \tag{7}$$

where $\mathbf{I} + \mathbf{D} - \mathbf{W}$ is a $(N \times d) \times (N \times d)$ matrix.

To solve the linear system, we need to calculate the inverse of such a large matrix, which is computationally expensive, especially when N and d are both large. Alternatively, we adopt an iterative strategy instead of using the closed-form solution. Recalling Eq. (3), we calculate the partial derivation of $E(X)$ with respect to $\mathbf{x}_i$ and let $\partial E(X)/\partial \mathbf{x}_i = \mathbf{0}$. Then, we can obtain an iterative solution for $\mathbf{x}_i$ under the coordinate descent framework:

$$\mathbf{x}_i^* = (\mathbf{I} + \sum_j \mathbf{w}_{ij})^{-1}(\mathbf{z}_i + \sum_j \mathbf{w}_{ij}\mathbf{x}_j). \tag{8}$$

Since Eq. (3) is convex, the coordinate descent algorithm can guarantee the global minimum, which is validated in the experimental section. If we further assume that point i is only related to its neighbor points $\mathcal{N}(i) = \{j \big| \|\mathbf{p}_i - \mathbf{p}_j\|_2^2 \le r\}$ in the point cloud and normalize the similarity to $\hat{s}_{ij} = s_{ij}/\sum_{j\in\mathcal{N}(i)} s_{ij}$, Eq. (8) can be further simplified to:

$$\mathbf{x}_i^* = (\mathbf{I} + \mathbf{C})^{-1}(\mathbf{z}_i + \mathbf{C}\sum_{j\in\mathcal{N}(i)} \hat{s}_{ij}\mathbf{x}_j). \tag{9}$$

It is easy to find that Eq. (3) actually contains a graph structure implicitly. We denote the graph defined on a point cloud as $\mathcal{G}_\mathcal{P}(\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P})$, where $\mathcal{V}_\mathcal{P}$ is the node set that contains all points ($|\mathcal{V}_\mathcal{P}| = N$). $\mathcal{E}_\mathcal{P}$ represents the edge set that indicates whether two points are related to each other. Therefore, the update equation Eq. (9) can be described as a message-passing process in $\mathcal{G}_\mathcal{P}$, as shown in Algorithm 1.

Algorithm 1 The message-passing process of Eq. (3)

Input: Point cloud graph $\mathcal{G}_\mathcal{P}(\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P})$, observed feature $Z = \{\mathbf{z}_i | i \in \mathcal{V}_\mathcal{P}\}$, maximum iteration steps T.

Initialize: $\mathbf{x}_i \leftarrow \mathbf{z}_i$ for $i \in \mathcal{V}_\mathcal{P}$
Compute the normalized similarity: $\hat{s}_{ij}$ for $e(i, j) \in \mathcal{E}_\mathcal{P}$
for $t = 1 : T$ **do**
  for $i \in \mathcal{V}_\mathcal{P}$ **do**
    Message passing: $msg_{j\to i} = \sum_{e(i,j)\in\mathcal{E}_\mathcal{P}} \hat{s}_{ij}\mathbf{x}_j$
    Compatibility transformation: $msg'_{j\to i} = \mathbf{C}msg_{j\to i}$
    Add unary: $\mathbf{x'}_i = \mathbf{z}_i + msg'_{j\to i}$
    Normalize: $\mathbf{x}_i = (\mathbf{I} + \mathbf{C})^{-1}\mathbf{x'}_i$
  end for
end for

Output: Latent feature $X = \{\mathbf{x}_i | i \in \mathcal{V}_\mathcal{P}\}$.

3.2. The message-passing graph convolution

Gilmer et al. proposed a universal GNN paradigm named message-passing neural networks (MPNNs) in [28]. The forward pass of MPNN has two phases: message passing and readout. In fact, most graph convolutional networks previously proposed[29, 30, 31] follow such a message-passing paradigm. Accordingly, we can also reformulate Algorithm 1 as a message-passing graph convolution.

Denote $\mathbf{x}_i^l$ as the feature vector of node i in the l_{th} layer of the network. Our message-passing graph convolution can be defined as:

Initial step:

$$\mathbf{h}_i^{(l,0)} = \mathbf{z}_i^{(l)} = \text{MLP}(\mathbf{x}_i^{(l-1)}), \tag{10}$$

Message-passing step:

$$\text{Message:} \quad \mathbf{m}_i^{(l,t)} = \mathbf{C} \sum_{j \in \mathcal{N}(i)} \hat{s}_{ij} \mathbf{h}_j^{(l,t-1)}, \tag{11}$$

$$\text{Update:} \quad \mathbf{h}_i^{(l,t)} = (\mathbf{I} + \mathbf{C})^{-1}(\mathbf{h}_i^{(l,0)} + \mathbf{m}_i^{(l,t)}), \tag{12}$$

Readout step:

$$\mathbf{x}_i^l = \sigma(\mathbf{h}_i^{(l,T)}), \tag{13}$$

where $\text{MLP}(\cdot)$ denotes the multilayer perceptron which is used for the pointwise feature transformation. $\mathbf{m}_i^{(l,t)}$ and $\mathbf{h}_i^{(l,t)}$ denote the message and the hidden state of node i in the l_{th} layer after t message-passing steps, respectively. $\sigma(\cdot)$ acts as a nonlinear activation function in the read out step. The data flow of the message-passing step is illustrated in Figure 1. Unlike most one-step graph convolution methods, the message-passing phase of our graph convolution, which consists of a message function Eq. (11) and an update function Eq. (12), runs T times.

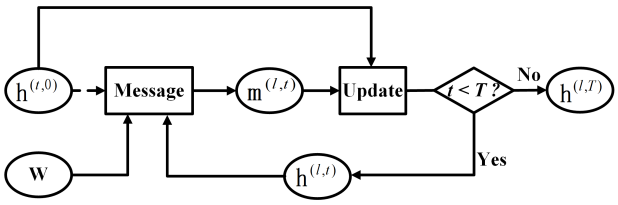


Figure 1: The data flow of the message-passing module of the proposed graph convolution.

3.3. The CRF Perspective

Consider a Gibbs distribution in Eq. (2) with the energy function given in Eq. (3). According to the Hammersley-Clifford theorem, in this case, Eq. (2) actually determines a continuous Gaussian CRF model defined on $\mathcal{G}_{\mathcal{P}}(\mathcal{V}_{\mathcal{P}}, \mathcal{E}_{\mathcal{P}})$. Considering CRF, the maximization of $P(X)$ is also equivalent to the minimization of the energy function $E(X)$.

Theorem 1. *The coordinate decent of Eq. (3) is equivalent to the mean-field approximation of Eq. (2) when the approximate distribution* $Q_i \sim N(\mu_i, \mathbf{\Sigma}_i)$.

Proof. The goal of the mean-field approximation algorithm is to find a distribution $Q(X)$ that can be factorized as $Q(X) = \prod_i Q_i(\mathbf{x}_i)$ to approximate the original distribution $P(X)$, where we have $Q_i(\mathbf{x}_i) \sim N(\mathbf{x}_i; \mu_i, \mathbf{\Sigma}_i)$. Then, we can obtain $Q(X)$ by minimizing the Kullback-Leibler divergence $KL(Q\|P)$ between $Q(X)$ and $P(X)$. We obtain $\{(\mu_i^*, \mathbf{\Sigma}_i^*)|i \in \mathcal{V}_{\mathcal{P}}\}$ by solving the following problem:

$$\begin{aligned} \{(\mu_i^*, \mathbf{\Sigma}_i^*)|i \in \mathcal{V}_{\mathcal{P}}\} &= \underset{\{(\mu_i, \mathbf{\Sigma}_i)|i \in \mathcal{V}_{\mathcal{P}}\}}{argmin} \; KL(Q\|P) \\ &= \underset{\{(\mu_i, \mathbf{\Sigma}_i)|i \in \mathcal{V}_{\mathcal{P}}\}}{argmin} \; F[Q(X; \{(\mu_i, \mathbf{\Sigma}_i)|i \in \mathcal{V}_{\mathcal{P}}\}), E(X)], \end{aligned} \tag{14}$$

where $F[\cdot]$ is an energy functional about the energy function $E(X)$ and the approximation distribution $Q(X)$. Since the forms of $E(X)$ and $Q(X)$ are determined in our case, F can be viewed as a function of $\{(\mu_i, \mathbf{\Sigma}_i)|i \in \mathcal{V}_{\mathcal{P}}\}$ directly.

Q_i is supposed to be independent and identically distributed. Without losing generality, we consider each μ_i and $\mathbf{\Sigma}_i$ individually. By setting $\partial F/\partial \mu_i = 0$ and $\partial F/\partial \mathbf{\Sigma}_i = 0$, we can obtain the update equations of μ_i and $\mathbf{\Sigma}_i$, respectively:

$$\mu_i^* = (\mathbf{I} + \sum_{j \in \mathcal{N}(i)} \mathbf{w}_{ij})^{-1} (\mathbf{z}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{w}_{ij} \mu_j). \tag{15}$$

$$\mathbf{\Sigma}_i^* = \frac{1}{2} (\mathbf{I} + \sum_{j \in \mathcal{N}(i)} \mathbf{w}_{ij})^{-1}. \tag{16}$$

Actually, we are only interested in the mean vector μ_i of Q_i in that $max(Q_i(\mathbf{x}_i)) = Q_i(\mu_i)$. We find that the update equation of μ_i in the mean-field algorithm is equivalent to the update equation of $\mathbf{x}_i$ in the coordinate descent algorithm. In other words, the coordinate descent of Eq. (3) is equivalent to the mean-field approximation of a continuous CRF (Eq. (2)), by which Theorem 1 can be proved. Hence, we name the proposed message-passing graph convolution continuous CRF graph convolution (CRFConv). For clarity reasons, we only provide a brief proof here. The detailed proof of Theorem 1 can be found in the supplemental material.

4. The point cloud segmentation network

In this section, we show how to utilize the proposed CRFConv for point cloud segmentation. We design a point cloud segmentation network that includes two parts: an encoder and a decoder. An overview of the network architecture is shown in Figure 2. The CRFConv is embedded in the decoder to decode the high-level features. Specifically, we model the upsampling process with the CRF model rather than linear interpolation, by which the lost details of high-level features caused by the encoder can be restored gradually during feature decoding.

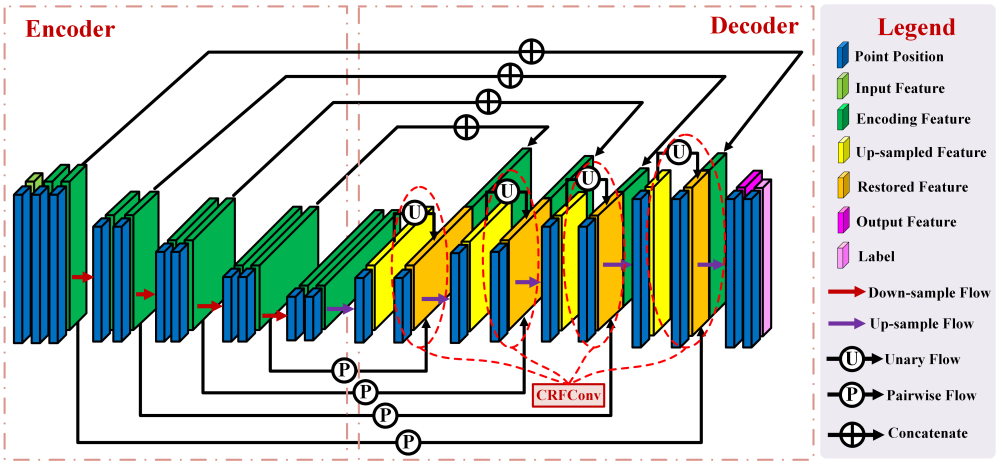


Figure 2: Overview of the proposed point cloud segmentation network. The red dotted circles indicate where the CRFConv was embedded.

4.1. The encoder

Because point clouds are unordered, sparse, and continuous in 3D Euclidean space, the convolution on a point cloud is different from that on an image. Analogous to image convolution, we adopt the convolution defined on irregular domains to extract the point features in the encoder. The standard convolution operation is defined as:

$$f * g(\mathbf{x}) = \int_{-\infty}^{+\infty} f(\mathbf{y}) g(\mathbf{x} - \mathbf{y}) d\mathbf{y} = \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} f(\mathbf{y}) g(\mathbf{x} - \mathbf{y}), \tag{17}$$

where f is the signal to be convolved, and g is the convolution kernel. In a regular domain (e.g., image), the offset $(\mathbf{x} - \mathbf{y})$ is fixed with respect to different kernel centers, which allows us to directly parameterize g with shared parameters. However, for an irregular domain (e.g. ,point cloud), the offset $(\mathbf{x} - \mathbf{y})$ and the neighborhood size $|\mathcal{N}(\mathbf{x})|$ may

vary at different positions, which leads to the fact that we cannot parameterize g directly. In the literature, researchers have attempted different methods to define the convolution on irregular domains[5, 6, 32, 15, 16]. In this paper, we follow [32, 15, 16] and parameterize the convolution kernel g as:

$$g(\mathbf{x}-\mathbf{y}) = \mathrm{MLP}(\mathbf{x}-\mathbf{y};\mathbf{\Theta}), \tag{18}$$

where $\mathbf{\Theta}$ is the learnable weights of the MLP. Therefore, the point convolution (PointConv) used in this paper can be written as:

$$f * g(\mathbf{x}) = \frac{1}{|\mathcal{N}(\mathbf{x})|}\sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x})} f(\mathbf{y})\mathrm{MLP}(\mathbf{x}-\mathbf{y};\mathbf{\Theta}), \tag{19}$$

in which normalization is necessary to make it robust to various neighbor sizes.

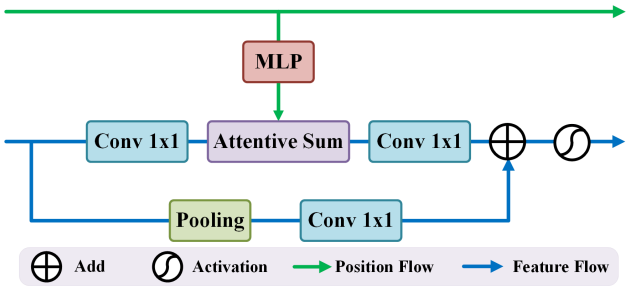


Figure 3: The PointConv used in the encoding layers. Note that in the residual path, pooling and 1×1 convolution are adopted to align the point cloud size or feature dimension if necessary.

d_{in} and d_{out} are denoted as the input and output feature dimensions, respectively. For each PointConv, the kernel $g(\cdot)$ is evaluated $N \times |\mathcal{N}| \times d_{\text{in}} \times d_{\text{out}}$ times. This is computationally expensive in practice, especially when both the number of points and the feature dimension are large. Inspired by the depthwise separable convolution of image networks, we separate canonical PointConv into a depthwise point convolution and several 1×1 convolutions. Figure 3 illustrates our PointConv layer in detail. Specifically, we first use a 1×1 convolution to reduce the dimension of the input feature from d_{in} to d_{in}/r with a proportion of r. Then, we evaluate $N \times |\mathcal{N}| \times (d_{\text{in}}/r)$ kernels for depthwise point convolution. Finally, another 1×1 convolution with $(d_{\text{in}}/r) \times d_{\text{out}}$ parameters is adopted to produce the output features. Moreover, batch normalization (BN) and residual connections are also adopted for PointConv. Dilated kNN neighbors are constructed for dilated PointConv to increase the receptive field of the encoder, which is proven to be helpful in image segmentation[24].

4.2. The decoder

In the literature, successive pooling operations or strided convolutions are adopted to increase the receptive field of the encoder. In the decoding stage, they simply adopt linear interpolation to upsample the high-level features, by which the details of these features are lost. Such linear interpolation only considers the positions of the features but ignores their continuity in feature space. The high-level features become increasingly coarser during decoding. Previous methods skip connections to relieve this situation. However, no previous study has fundamentally attempted to solve this problem. To address this problem, we propose using the CRFConv to restore the details of the high-level features during decoding. In our method, not only the point positions but also the distribution of the low-level features are considered for feature decoding. CRFConv is utilized to model the data affinity in feature space to generate high-resolution and detailed high-level features.

Denote $\mathbf{x}^{(l-1)}$ and $\mathbf{x}^{l}$ as the high-level feature to be upsampled (the input feature in the decoder) and the upsampled high-level feature (the output feature in the decoder), respectively. Let $\mathbf{x}'^{(l)}$ be the corresponding lower-level feature in the encoder that has the same resolution as $\mathbf{x}^{(l)}$. Figure 4(a) gives the architecture of our decoding layer. The high-level feature is initially upsampled by kNN interpolation. Then, a *unary net* is immediately adopted to extract the unary feature, which produces $\mathbf{z}^{(l)}$ as described in Eq.(10). Then, we dynamically compute the similarity s_{ij}. We adopt a *pairwise net* that takes the corresponding feature $\mathbf{x}'^{(l)}$ in the encoding layer to compute s_{ij}. The CRFConv

takes these two inputs to generate the restored feature via the message-passing module, as shown in Figure 1. Once the message-passing completes, the restored feature is further concatenated with the corresponding lower-level feature $\mathbf{x}'^{(l)}$ in the encoding layer to feed the next decoding layer after activation.

Unary Net. We simply take an MLP as the unary net, which conducts a pointwise feature transformation for the point features. The unary net plays the roles of dimension reduction and feature fusion in the decoding layers.

Pairwise Net. The pairwise net is used to compute the similarity s_{ij} of each point pair (i, j). For each high-level point feature pair $(\mathbf{x}_i, \mathbf{x}_j)$, we compute its similarity with the corresponding point feature pair $(\mathbf{x}'_i, \mathbf{x}'_j)$ in a lower-level feature space. We adopt the Mahalanobis distance in the low-level feature space to compute the similarity s_{ij}. The vanilla Mahalanobis distance is computed as:

$$d^2(\mathbf{x}'_i, \mathbf{x}'_j) = (\mathbf{x}'_i - \mathbf{x}'_j)^\top \mathbf{M}(\mathbf{x}'_i - \mathbf{x}'_j) = (\mathbf{x}'_i - \mathbf{x}'_j)^\top \mathbf{P}\mathbf{P}^\top(\mathbf{x}'_i - \mathbf{x}'_j) = \|\mathbf{P}^\top \mathbf{x}'_i - \mathbf{P}^\top \mathbf{x}'_j\|_2^2, \tag{20}$$

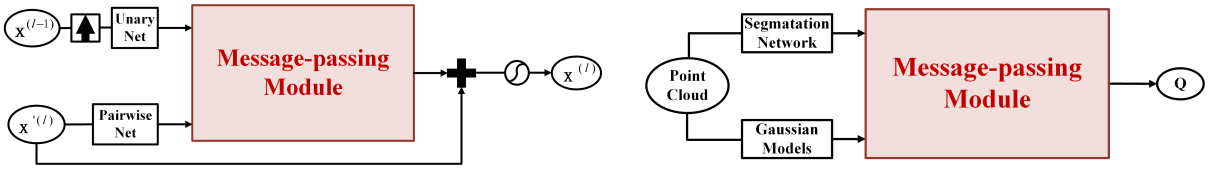
where $\mathbf{x}'_i, \mathbf{x}'_j \in \mathbb{R}^d$, $\mathbf{M} \in \mathbb{R}^{d\times d} \succ \mathbf{0}$, $\mathbf{P} \in \mathbb{R}^{d\times d'}$. The Mahalanobis distance is actually the Euclidean distance on a projected space determined by the linear projection matrix $\mathbf{P}$, which may fail to model nonlinear data effectively. In this paper, we reformulate $\mathbf{P}$ as a nonlinear projection to better model nonlinear data. We let the network learn a measure function automatically with an MLP. Thus, the improved Mahlanobis distance used in this paper can be computed as:

$$d'^2(\mathbf{x}'_i, \mathbf{x}'_j) = \|\mathrm{MLP}(\mathbf{x}'_i) - \mathrm{MLP}(\mathbf{x}'_j)\|_2^2. \tag{21}$$

Accordingly, the normalized similarity $\hat{s}_{ij}$ is given as:

$$\hat{s}_{ij} = \frac{\exp(-d'^2(\mathbf{x}'_i, \mathbf{x}'_j))}{\sum_{j\in\mathcal{N}(i)} \exp(-d'^2(\mathbf{x}'_i, \mathbf{x}'_j))}, \tag{22}$$

which can be easily implemented with a softmax layer in the network.



(a) The continuous CRF convolution layer.

(b) The discrete CRF convolution layer.

Figure 4: The continuous and discrete CRF convolution layer used in the point cloud segmentation network. The data flow of the message-passing modules here can be represented with Figure 1.

4.3. Classical discrete CRF as graph convolution

Most previous studies utilize the discrete CRF model to refine the labels in segmentation tasks, while our CRF is a continuous model defined in feature space. Hence our continuous CRF can work collaboratively with the classical discrete CRF to further improve the segmentation performance. In image segmentation, the energy function of the discrete CRF is usually defined as a pairwise model[23]:

$$E(X) = \sum_i \psi_\mathrm{u}(x_i) + \sum_i \sum_{j\in\mathcal{N}(i)} \psi_\mathrm{p}(x_i, x_j), \tag{23}$$

where $x_i \in \mathcal{L}$ presents the label of pixel (node) i. Here $\mathcal{L}$ is a discrete label set. $\psi_\mathrm{u}(x_i) \geq 0$ is the unary potential, which gives a penalty when each pixel is assigned a label. In the postprocessing CRFs, the unary potential can be simply computed as $\psi_\mathrm{u}(x_i) = -\log p(x_i)$, where $p(x_i)$ is the probability predicted by a discriminative classifier. $\psi_\mathrm{p}(x_i, x_j) = \mu(x_i, x_j)K(\mathbf{f}_i, \mathbf{f}_j)$ is the pairwise potential, which penalizes two connected pixels with different labels. Here, $\mu(\cdot,\cdot)$ measures the compatibility between different classes. $K(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ is a positive function

defined on a feature space of the pixels. The update equation of the approximate distribution Q_i in the mean-field approximation of Eq. (23) is given in [23]:

$$Q_i^*(x_i) = \frac{1}{Z_i} \exp\Big\{ -\psi_{\mathrm{u}}(x_i) - \sum_{l \in \mathcal{L}} \mu(x_i, l) \sum_{j \in \mathcal{N}(i)} K(\mathbf{f}_i, \mathbf{f}_j) Q_j(l) \Big\}. \tag{24}$$

We show that Eq. (23) can also be used for point cloud segmentation. For each point i, let $\mathbf{p}_i$ be the output probability vector of the segmentation network and $\mathbf{q}_i$ be the vector version of Q_i. Eq. (24) can be compactly rewritten as:

$$\mathbf{q}_i^* = \mathrm{Softmax}\Big(\log(\mathbf{p}_i) - \mathbf{C} \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{q}_j \Big), \tag{25}$$

where we denote $w_{ij} = K(\mathbf{f}_i, \mathbf{f}_j)$. $\mathbf{C} \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}|}$ is a matrix that encodes the compatibility of labels. Here, we have no constraint on $\mathbf{C}$ and learn it from the data. If we set $\mathbf{C} = \mathbf{I}$, it will degenerate to the Potts model, that is, $\mu(x_i, x_j) = 0$ if $x_i = x_j$ otherwise 1, which is widely adopted in most previous image segmentation CRFs.

Algorithm 2 Message passing of the discrete CRF

Input: Point cloud graph $\mathcal{G}_{\mathcal{P}} = (\mathcal{V}_{\mathcal{P}}, \mathcal{E}_{\mathcal{P}})$, initial distribution $P = \{\mathbf{p}_i | i \in \mathcal{V}_{\mathcal{P}}\}$, maximum iteration step T.

```
Initialize: q_i ← p_i for i ∈ V_P
Compute weights: w_ij = K(f_i, f_j) for e(i, j) ∈ E_P
for t = 1 : T do
    for i ∈ V_P do
        Message-passing: msg_{j→i} = Σ_{e(i,j)∈E_P} w_ij q_j
        Compatibility transformation: msg'_{j→i} = C msg_{j→i}
        Add unary: q̃_i = log(p_i) − msg'_{j→i}
        Normalize: q_i = Softmax(q̃_i)
    end for
end for
```

Output: Approximate distribution $Q = \{\mathbf{q}_i | i \in \mathcal{V}_{\mathcal{P}}\}$.

Eq. (25) can be formulated as another message-passing algorithm on the point cloud graph $\mathcal{G}_{\mathcal{P}}(\mathcal{V}_{\mathcal{P}}, \mathcal{E}_{\mathcal{P}})$, as shown in Algorithm 2. Analogous to CRFConv, we can further reformulate Algorithm 2 as another message-passing graph convolution for point cloud segmentation:

Initial step:

$$\mathbf{h}_i^{(0)} = \mathbf{p}_i, \tag{26}$$

Message-passing step:

$$\text{Message:} \quad \mathbf{m}_i^t = \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{h}_j^{(t-1)}, \tag{27}$$

$$\text{Update:} \quad \mathbf{h}_i^t = \mathrm{Softmax}\Big(\log(\mathbf{p}_i) - \mathbf{C}\mathbf{m}_i^t \Big), \tag{28}$$

Readout step:

$$\mathbf{q}_i = \mathbf{h}_i^T. \tag{29}$$

Following [23], we set $w_{ij} = \sum_{m=1}^{M} \omega_m k_m(\mathbf{f}_i, \mathbf{f}_j)$ as the linear combination of multiple Gaussian, where $k_m(\mathbf{f}_i, \mathbf{f}_j) = \exp\big(-(\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{\Sigma}_m^{-1} (\mathbf{f}_i - \mathbf{f}_j) \big)$. Instead of adopting artificially designed parameters such as [23], we learn ω_m and $\mathbf{\Sigma}_m$ of each Gaussian component from the data automatically. To ensure positive definition and avoid the inverse operation, we let $\mathbf{\Sigma}_m^{-1} = \mathbf{P}\mathbf{P}^\top$, where $\mathbf{P} \in \mathbb{R}^{d \times d'}$, which can be implemented with a linear layer having d' outputs. Similarly, the weight sum operation can also be implemented with another linear layer having only one output. The data flow of the discrete CRF convolution layer is shown in Figure 4(b). It can be added at the end of our segmentation network as an additional graph convolution layer to form a dual CRF network. In addition, the whole network can be trained end-to-end.

5. Experiments

The experimental platform is a PC equipped with an Intel Core i7 9700K CPU, 16 GB RAM, and an Nvidia TITAN RTX GPU (24 GB). The algorithm is implemented with *PyTorch*[33] and *PyTorch Geometric*[34] on Ubuntu 16.04.

5.1. Datasets

The experiments are conducted on various 3D point cloud segmentation datasets, including ShapeNet[35] for object-level evaluation and S3DIS[36] and Semantic3D[37] for indoor and outdoor scene-level evaluation.

ShapeNet. ShapeNet is a 3D object model dataset that contains 16,880 object models from 16 categories. Each 3D object consists of approximately 2k points. ShapeNet Parts is the part segmentation benchmark for the ShapeNet models, in which each model annotates 2 to 6 parts with a part label (50 parts in total).

S3DIS. S3DIS is a large-scale indoor 3D scene segmentation dataset that was collected on six large-scale indoor areas from 3 different buildings. The scene is annotated with 13 categories (including clutter). Following [4], we organize the dataset by rooms (271 rooms in total). In our experiments, we split the training and test sets according to area.

Semantic3D. Semantic3D is a large-scale outdoor point cloud segmentation online benchmark. This benchmark provides a large labeled 3D point cloud dataset of natural scenes with over 4 billion points in total, which covers a range of diverse urban scenes. We conduct our experiments on the reduced-8 benchmark, which contains 15 scenes for training and 4 reduced scenes for testing with 8 semantic categories.

5.2. Experimental Settings

The encoder of our segmentation network contains five convolution blocks, each of which consists of two successive PointConv layers. The cloud resolution is reduced by the farthest point sampling (fps) with a ratio of r after each convolution block; additionally, the feature dimension d is doubled. In our experiment, we set $r = [1.0, 0.25,\ 0.375,\ 0.375,\ 0.375]$ and $d = [64,\ 128,\ 256,\ 512,\ 1024]$. The decoder consists of four corresponding CRFConv layers to restore the cloud resolution to the original input size gradually. We use kNN to search the neighbors for each point. The kernel (neighbor) size k is set to 32 in the first layer and 16 in the remaining layers. The dilated kNN is only available in the encoder. We set the dilation rate $dil = [1,\ 2,\ 4,\ 4,\ 2]$ in the five PointConv blocks. Two fully connected layers follow the last decoding layer to obtain per-point labels. The discrete CRF is added after the last fully connected layer if necessary. To make fair comparisons with the previous methods, the discrete CRF convolution is not adopted in the comparison experiments. The performance of the discrete CRF convolution is evaluated in the ablation study. The detailed network architecture and more experimental details can be found in the supplementary material.

5.3. Object-Level Evaluation

We conduct a shape part segmentation experiment on the ShapeNet Parts benchmark. Instead of training each category individually, we train a model with all part labels (50 in total). The normalized x, y, z position and the normal vector n_x, n_y, n_z of each point are taken as the input features. Because our network supports various input point cloud sizes, we do not sample the points as in previous methods[4, 14, 5]. The intersection over union (IoU) of the object parts is used as the metric to evaluate the performance. The instance part IoU (pIoU) and mean class pIoU (mpIoU) are given in Table 1. Compared with nearly twenty current popular methods, the performance of the proposed method exceeds most previous methods. From Table 1, we can find that KPConv[6] performs best overall, which may be due to the better feature encoding ability. Even compared with state-of-the-art methods, the proposed method can achieve competitive performance. Please see the supplementary material for some visualization results.

Table 1: Performance of the proposed method on the ShapeNet Part segmentation benchmark.

	Methods	**pIoU**	**mpIoU**	air	bag	cap	car	chair	ear	guitar	knife	lamp	laptap	motor	mug	pistol	rocket	bord	tabel
2017	Kd-Net[13]	82.3	77.4	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
	PointNet[4]	83.7	80.4	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
	PointNet++[14]	85.1	81.9	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
	SyncSpecCNN[38]	84.7	82.0	81.6	81.7	81.9	75.2	90.2	74.9	**93.0**	86.1	84.7	95.6	66.7	92.7	81.6	60.6	82.9	82.1
	Pd-Network[13]	85.5	82.7	83.3	82.4	87.0	77.9	90.9	76.3	91.3	87.3	84.0	95.4	68.7	94.0	82.9	63.0	76.4	83.2
2018	KCNet[39]	83.7	82.2	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.2	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
	SO-Net[40]	84.9	81.0	82.8	77.8	88.0	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	83.0
	RSNet[41]	84.9	81.4	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
	PCNN[42]	85.1	81.8	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
	SpiderCNN[43]	85.3	81.7	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
	SGPN[44]	85.8	82.8	80.4	78.6	78.8	71.5	88.6	78.0	90.9	83.0	78.8	95.8	77.8	93.8	87.4	60.1	**92.3**	**89.4**
	PointCNN[5]	86.1	84.6	84.1	**86.5**	86.0	80.8	90.6	79.7	92.3	88.4	85.3	96.1	77.2	95.3	84.2	64.2	80.0	83.0
2019	DGCNN[20]	84.9	81.4	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
	RS-CNN[45]	86.2	84.0	83.5	84.8	88.8	79.6	91.2	**81.1**	91.6	88.4	**86.0**	96.0	73.7	94.1	83.4	60.5	77.7	83.6
	KPConv rigid[6]	86.2	85.0	83.8	86.1	88.2	**81.6**	91.0	80.1	92.1	87.8	82.2	**96.2**	77.9	95.7	**86.8**	65.3	81.7	83.6
	KPConv deform[6]	**86.4**	**85.1**	**84.6**	86.3	87.2	81.1	91.1	77.8	92.6	88.4	82.7	**96.2**	**78.1**	**95.8**	85.4	**69.0**	82.0	83.6
2020	3D-GCN[46]	85.1	82.1	83.1	84.0	86.6	77.5	90.3	74.1	90.0	86.4	83.8	95.6	66.8	94.8	81.3	59.6	75.7	82.8
2021	PAConv[47]	86.1	84.6	84.3	85.0	**90.4**	79.7	90.6	80.8	92.0	**88.7**	82.2	95.9	73.9	94.7	84.7	65.9	81.4	84.0
	CRFConv (ours)	85.5	83.5	83.9	84.8	83.0	80.2	**91.8**	77.9	91.8	86.9	84.9	95.6	77.8	95.6	82.0	64.4	75.3	80.8

5.4. Scene-Level Evaluation

5.4.1. Data Preparation

Conventional Implementation. Scene-level point cloud data are more complex than object-level data. The former is spatially continuous and usually nonuniform in density and cannot be processed by the network directly. Following [4, 14, 5], we sample blocks from a scene point cloud in training and testing. For indoor scenes, the point cloud is first collected by rooms. Then, each block is randomly sampled from each room. For outdoor scenes, we directly sample blocks from each scene. To control the block size, we then uniformly sample N points in each block to feed the network. In training, the blocks are randomly sampled from the training set in each epoch. In the test, we sample multiple blocks in the test set until each point is evaluated at least once. After that, the final per-point labels can be obtained by voting. In this strategy, neighbor searching and cloud sampling are computed directly in the network. We adopt this strategy as a conventional implementation in scene-level evaluation experiments.

Advanced Implementation. In conventional implementations, the network will require considerable computation to organized the structure of the point cloud (i.e., neighbor searching and cloud sampling). Therefore, some methods[6, 48] adopt a more computationally efficient method to organize the input point cloud blocks. In this strategy, the multiscale point cloud and the point neighbors at each scale are efficiently precomputed on the CPU. This strategy allows them to handle ten times the input size than the conventional implementation under the same time complexity, which is helpful for training the network on large-scale point clouds. To compare with them, we also employ this strategy as an advanced implementation for the proposed method and conduct experiments on scene-level benchmarks. The advanced implementation is denoted as *CRFConv+* to be distinct from the conventional implementation, which is denoted as *CRFConv*.

5.4.2. Indoor Scene Segmentation

The proposed method is evaluated on the S3DIS dataset for scene-level segmentation tasks. We adopt the normalized coordinates in the block and the corresponding RGB color as the input feature. We set the input point size to 8,192 for the conventional implementation (CRFConv) and 40,960 for the advanced implementation (CRFConv+). We take the overall accuracy (OA), mean accuracy (mACC), mean IoU (mIoU), and per class IoU as metrics to evaluate the performance. Since previous methods usually give the results on Area 5, we also report the results on this area in Table 2. From the experimental results, our method can achieve state-of-the-art performance with the precomputed multiscale strategy. For conventional implementation, the proposed method can also outperform some previous methods that prepare data in a conventional style, such as PointNet and PointCNN.

Table 2: Performance of the proposed method on the Stanford S3DIS dataset (Area 5).

	Methods	**OA**	**mACC**	**mIoU**	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
2017	PointNet[4]	-	49.0	41.1	88.8	97.3	69.8	0.1	3.9	46.3	10.8	58.9	52.6	5.9	40.3	26.4	32.2
	SegCloud[49]	-	57.4	48.9	90.1	96.1	69.9	0.0	18.4	38.4	23.1	70.4	75.9	40.9	58.4	13.0	41.6
2018	Eff 3D Conv[50]	-	68.3	51.8	79.8	93.9	69.0	0.2	28.3	38.5	48.3	73.6	71.1	59.2	48.7	29.3	33.1
	TangentConv[51]	-	62.2	52.6	90.5	97.7	74.0	0.0	20.7	39.0	31.3	77.5	69.4	57.3	38.5	48.8	39.8
	RNN Fusion[52]	-	63.9	57.3	92.3	98.2	79.4	0.0	17.6	22.8	62.1	80.6	74.4	66.7	31.7	62.1	56.7
	PointCNN[5]	85.9	63.9	57.3	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
	SPGGraph[44]	86.4	66.5	58.0	89.4	96.9	78.1	0.0	**42.8**	48.9	61.6	**84.7**	75.4	69.8	52.6	2.1	52.2
	PCNN[42]	-	67.0	58.3	92.3	96.2	75.9	**0.3**	6.0	**69.5**	63.5	66.9	65.6	47.3	68.9	59.1	46.2
2019	GACNet[32]	87.8	-	62.9	92.3	98.3	81.9	0.0	20.4	59.1	40.9	78.5	85.8	61.7	70.8	**74.7**	52.8
	PointWeb[53]	87.0	66.6	60.3	92.0	98.5	79.4	0.0	21.1	59.7	34.8	76.3	88.3	46.9	69.3	64.9	52.5
	KPConv rigid[6]	-	70.9	65.4	92.6	97.3	81.4	0.0	16.5	54.5	69.5	80.2	90.1	66.4	74.6	63.7	58.1
	KPConv deform[6]	-	72.8	**67.1**	92.8	97.3	82.4	0.0	23.9	58.0	69.0	81.5	**91.0**	**75.4**	**75.3**	66.7	**58.9**
2020	FPConv[54]	-	-	62.8	**94.6**	98.5	80.9	0.0	19.1	60.1	48.9	80.6	88.0	53.2	68.4	68.2	54.9
	SegGCN	88.2	70.4	63.6	93.7	**98.6**	80.6	0.0	28.5	42.6	**74.5**	80.9	88.7	69.0	71.3	44.4	54.3
	PointASNL[55]	87.7	68.5	62.6	94.3	98.4	79.1	0.0	26.7	55.2	66.2	83.3	86.8	47.6	68.3	56.4	52.1
2021	PAConv[47]	-	72.3	65.6	93.1	98.4	**82.6**	0.0	22.6	61.3	63.3	78.5	88.0	64.5	73.5	70.1	57.3
	BCM+AFM[56]	88.9	73.1	65.4	92.9	97.9	82.3	0.0	23.1	65.5	64.9	78.5	87.5	61.4	70.7	68.7	57.2
	CRFConv (ours)	88.4	68.6	62.0	94.0	97.5	81.8	0.0	21.7	62.2	45.4	78.1	85.2	51.1	71.8	61.7	55.7
	CRFConv+ (ours)	**89.2**	**73.7**	66.2	93.3	96.3	82.2	0.0	23.7	60.3	68.2	82.4	86.0	63.4	73.8	72.4	**58.9**

5.4.3. Outdoor Scene Segmentation

We further evaluate the proposed method on the large-scale outdoor 3D point cloud segmentation benchmark Semantic3D. We first uniformly downsample the training set before sampling blocks to maintain a voxel resolution of $0.01m^3$, which is the as the reduced test set. The normalized position in the block and the corresponding RGB color are taken as the input features for the network. The input point cloud sizes are set to 8,192 and 65,536 for the conventional and advanced implementations, respectively. Because Semantic3D is an online benchmark, we submit the experimental results on the reduced-8 test set to the benchmark website.[3] The experimental results are also shown in Table 3. Note that only the published methods are listed in the table. From the table, the performance of RandLA-Net and SCF-Net is state-of-the-art overall. Next, the proposed method performs comparably with KPConv and BCM_AFM, and outperforms the other methods.

5.4.4. Discussion

According to the experimental results for scene-level evaluation, the proposed method can perform well in both indoor and outdoor scenes, which demonstrates the robustness of our method. Overall, the proposed method performs better in indoor scenes because the outdoor scenes are usually more complex and biased than indoor scenes. In Table 3, we find that our method performs better on large or regular objects such as terrain and cars but worse on small or irregular objects such as vegetation and landscape. The reason can be explained by the fact that the former will benefit from the proposed CRF model, but the latter may be erroneously viewed as noise and smoothed by the CRF model in feature extraction, especially when the training samples involving the case are very limited. Actually, this phenomenon occurs not only in this dataset but will be magnified in a biased situation. We will try to address this problem in future works. In addition, we find that the advanced implementations with a precomputed multiscale strategy can largely improve the performance on both indoor and outdoor scenes compared with the conventional implementations. This is because the block sampling strategy damages the completeness of the continuous point cloud, especially when the block size is small, which may harm the overall performance in some way. In contrast, the precomputed multiscale strategy allow us to handle a larger input block size, which is helpful for network learning on scene-level point clouds. More visualization results on the scene-level datasets can be found in the supplementary material.

[3] http://semantic3d.net/view_results.php?chl=2

Table 3: Performance of the proposed method on the Semantic3D dataset (reduced-8).

	Methods	**OA**	**mACC**	**mIoU**	man-made terrain	natural terrain	high vegetation	low vegetation	buildings	hard scape	scanning artefacts	cars
2017	SnapNet[57]	88.6	70.8	59.1	82.0	77.3	79.7	22.9	91.1	18.4	37.3	64.4
	SEGCloud[49]	88.1	73.1	61.3	83.9	66.0	86.0	40.5	91.1	30.9	27.5	64.3
2018	RF_MSSF[58]	90.3	71.6	62.7	87.6	80.3	81.8	36.4	92.2	24.1	42.6	56.6
	MSDVN[59]	88.4	77.2	65.3	83.0	67.2	83.8	36.7	92.4	31.3	50.0	78.2
	SPGraph[44]	94.0	79.6	73.2	97.4	92.6	87.9	44.0	93.2	31.0	63.5	76.2
2019	ShellNet[60]	93.2	79.7	69.3	96.3	90.4	83.9	41.0	94.2	34.7	43.9	70.2
	GACNet[61]	91.9	79.5	70.8	86.4	77.7	**88.5**	**60.6**	94.2	37.3	43.5	77.8
	KPConv[6]	92.9	81.4	74.6	90.9	82.2	84.2	47.9	94.9	40.0	**77.3**	79.7
2020	RandLA-Net[48]	**94.8**	84.1	77.4	95.6	91.4	86.6	51.5	**95.7**	**51.5**	69.8	76.8
	FGCN[62]	-	**89.3**	62.4	90.3	65.2	86.2	38.7	90.1	31.6	28.8	68.2
2021	BCM+AFM[56]	94.3	-	75.3	96.3	**93.7**	87.7	48.1	94.6	43.8	58.2	79.5
	SCF-Net[63]	94.7	85.2	**77.6**	97.1	91.8	86.3	51.2	95.3	50.5	67.9	80.7
	CRFConv (ours)	93.5	78.4	72.5	95.0	93.1	86.4	40.8	93.3	33.7	60.5	76.8
	CRFConv+ (ours)	94.2	84.1	74.9	**98.1**	91.1	81.8	44.8	95.2	40.8	59.4	**88.0**

5.5. Ablation Study

To further evaluate the performance of the continuous/discrete CRF convolution, we conduct an ablation study on the S3DIS dataset for the proposed method. First, we compare the performance of the proposed method with/without the continuous/discrete CRF convolution. Table 4 shows the experimental results on Area 5. Note that the results in this section are conventionally implemented without merging labels. We adopt the same encoder for all models. In the baseline model, we remove the continuous CRF convolution in the decoder and only perform linear interpolation for point cloud upsampling. Baseline+discrete CRF denotes that we add the discrete CRF at the end of the baseline model. Baseline+continuous CRF employs continuous CRF convolution in the decoder, which is the general version of the proposed method. For baseline+dual CRF, we add the discrete CRF convolution at the end of the general version to form the dual CRF version of the proposed method.

Table 4: Performance of the proposed method under different module configurations.

Models	**OA**	**mACC**	**mIoU**
Baseline	89.78	66.25	59.36
Baseline+Discrete CRF	90.01	66.51	59.88
Baseline+Continuous CRF (general version)	90.13	67.91	61.22
Baseline+Dual CRF (dual CRF version)	**90.31**	**68.12**	**61.54**

In Table 4, even the baseline model achieves promising performance, which can also prove the effectiveness of the encoder we designed. Compared with the baseline model, the model with continuous CRF has achieves a significant improvement. Although the discrete CRF also improves the performance in some way, the improvement effect is marginal. In traditional discrete CRF, the performance of the model depends largely on the unary term. In other words, the misclassified regions caused by the classifier are hard to correct by the discrete CRF. However, our continuous CRF model can fundamentally improve the presentation (location) ability of the point features, therefore avoiding such discrete CRF model limitations. Additionally, some visualization comparisons with/without the CRFConv are illustrated in Figure 5. We can see that the segmentation results of the CRFConv model look smoother and more complete than those of the baseline model in most cases, especially on some boundary areas (see *office_2* and *office_40*). Some misclassified regions produced by the baseline model can also be corrected by the proposed CRFConv (see *hallway_1*).

The proposed CRFConv layer is flexible and scalable and can be used as a plug-and-play module for integrating

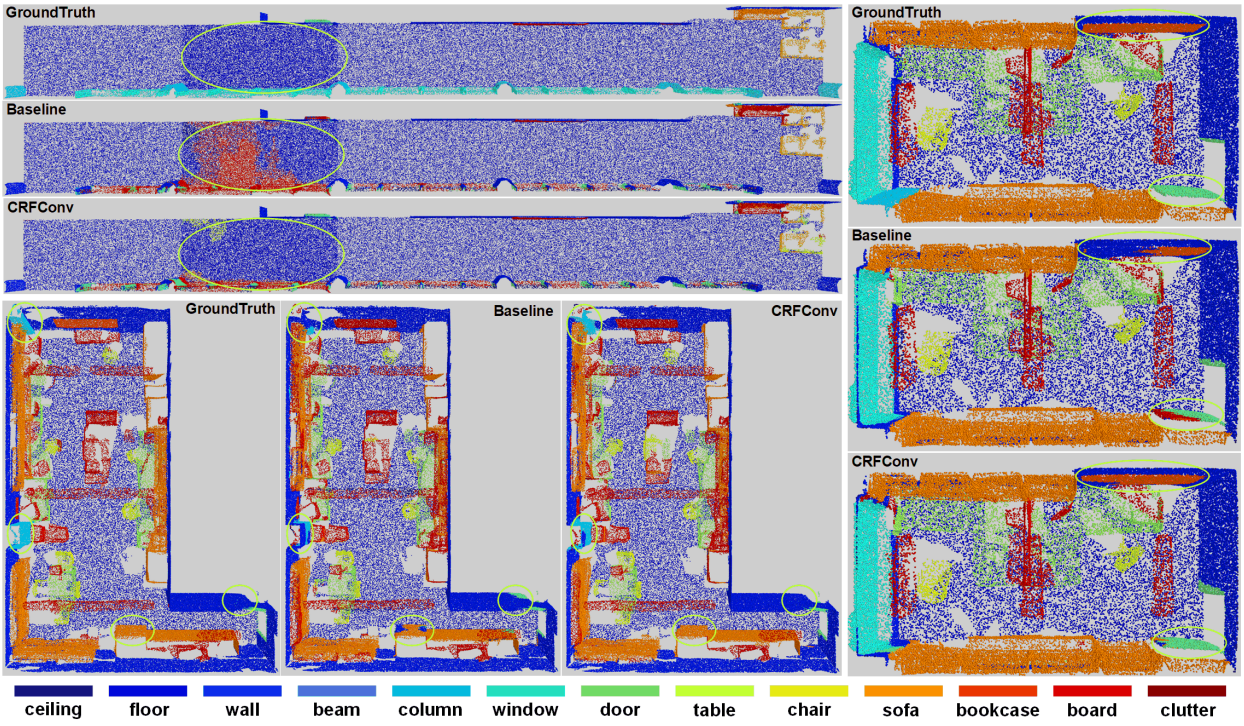


Figure 5: Visualization comparisons of the baseline model and the CRFConv model. We show three rooms in area 5 of the S3DIS dataset, that is, *hallway_1* on the top left, *office_2* on the right, and *office_40* on the bottom left. The results are visualized in the bird view from the upper axis. Note that the "ceilings" category is removed manually for better visualization. The regions marked by green circles show some significant local differences.

Table 5: Performance of RandLA-Net with/without CRFConv on S3DIS (Area 5).

Methods	**mIoU**	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
RandLA-Net[48]	63.0	91.5	97.0	81.1	**0.0**	22.5	60.9	44.8	**78.3**	87.7	**69.1**	71.1	65.2	50.0
RandLA-Net+CRFConv	**64.1**	**92.5**	**97.8**	**82.1**	**0.0**	**23.2**	**62.0**	**49.9**	78.1	**87.8**	56.9	**72.1**	**77.3**	**53.6**

into existing encoder-decoder networks to further improve their performance. In this section, we integrate CRFConv with RandLA-Net[48], which is one of the representative methods in the community, and conduct experiments on the S3DIS dataset. Specifically, we package the CRFConv as an upsampling layer and insert it into the decoding layers of the original RandLA-Net. All other settings follow the same schemes as the original paper. We report the results on Area5 in Table 5. In the table, our CRFConv can further improve its performance, which also proves the effectiveness of the proposed CRFConv. Through this experiment, we demonstrate that the proposed method has a certain generality that can improve the performance of the network independent of specific encoders.

5.6. Parameter Analysis

The mean-field step T of our CRFConv is a flexible parameter, which allows us to adopt different mean-field steps for the training and test phases. In training, we set the mean-field step as 1 for efficiency and to avoid vanishing/exploding gradients. Therefore, we evaluate the performance of various mean-field steps in the test phase. The performance of different mean-field steps on the validation set is illustrated in Figure 6. According to this figure, the continuous CRF performs best when the mean-field step is approximately 10. With the increase in the mean-field steps, the performance tends gradually become saturated. Owing to the fidelity ability of CRFConv, we do not observe obvious performance degradation caused by oversmoothness, which may occur in most current popular graph convolutions. However, we find that the model performs best when it is about to converge rather than converge. When the

model converges, the feature may become smoother, which may be harmful to some small or irregular objects. This can explain why the performance decreases slightly when the CRF reaches convergence. This experimental result is also coincident with our analysis of the outdoor scenes.

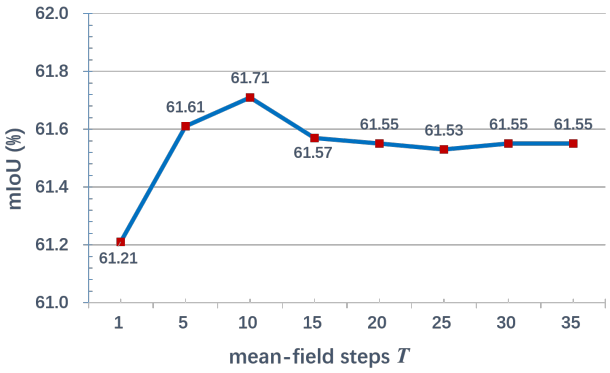


Figure 6: The test performance of the CRFConv on various mean-field steps.

5.7. Time complexity

In this section, we conduct a runtime analysis of the proposed method. First, the runtime of the proposed method with/without the precomputed multiscale strategy is given. Then, we additionally make a runtime comparison with some previous methods. Specifically, we choose the S3DIS benchmark for this experiment and run the previous methods on our experimental platform. The experimental setting follows the original paper. We give both the training and test runtime of each iteration including data preparation. To make a fair comparison, we make the input point cloud size of our method the same as previous methods. Note that, because KPConv adopts various input sizes, we report an approximate value here. The experimental results are shown in Table 6. From the table, we find that PointNet is the most efficient method because of its simple architecture. Although the overall runtime of the proposed method is higher than that of previous methods, the proposed CRFConv module only adds a small runtime cost to the backbone network in both training and testing. Moreover, due to the precomputed multiscale strategy, the efficiency of the network greatly improved. This strategy allows us to handle a larger batch size but with lower time complexity.

Table 6: Runtime analysis of the proposed method.

Methods	Batch Size	Precomputed Multiscale	Training Time (ms/iter)	Test Time (ms/iter)
PointNet[4]	16×4096	No	138	98
PointNet++(SSG)[14]	16×4096	No	532	495
PointNet++(MSG)[14]	16×4096	No	694	633
KPConv[6]	$\approx 8 \times 15000$	Yes	552	271
RandLA-Net[48]	8×40960	Yes	760	448
Baseline (ours)	8×8192	No	935	475
	8×40960	Yes	909	625
Baseline+CRFConv (ours)	8×8192	No	1141	653
	8×40960	Yes	990	667
RandLA-Net+CRFConv (ours)	8×40960	Yes	815	485

6. Conclusion

In this paper, we addressed the problem of point cloud segmentation. First, we proposed a continuous CRF graph convolution to capture the structure of the point features, by which the representation ability of the features improved.

Then, we designed an encoder-decoder point cloud segmentation network based on the proposed continuous CRF convolution. The continuous CRF convolution embedded in the decoder can restore the details of the high-level features produced by the encoder to enhance the localization ability of the network. Furthermore, we also showed how to formulate the discrete CRF as another message-passing graph convolution and combined it with the continuous CRF convolution, by which we modeled the data affinity not only in feature space but also in label space to further improve the segmentation results. Experiments on various datasets demonstrated the effectiveness of the proposed method.

However, compared with indoor scenes, we find that the proposed method performs worse in outdoor scenes that are usually more complex and biased than the former. In this case, the CRF model erroneously viewed some small or irregular objects as noise in feature extraction, especially when the number of training samples involving the case was very limited. This situation may be relieved by increasing the input point cloud size with the precomputed multiscale strategy. In the future, we will attempt to address this problem. In this work, we focused the decoding of the point cloud features and achieve promising results. However, the overall performance of the network depends on more than one aspect. For example, the feature encoder may play a decisive role. In future work, we will also attempt to investigate more feature encoding methods, such as convolution in the point cloud domain. Moreover, we will continue exploring the feasibility of integrating graph neural networks with probabilistic graphical models.

Acknowledgment

This work is partially supported by National Natural Science Foundation of China under Grant Nos 61872188, 61703209, U1713208, 61972204, 61672287, 61861136011, 61773215, and by the French Labex MS2T ANR-11-IDEX-0004-02 through the program Investments for the future.

References

[1] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239. doi:10.1109/34.969114. 1

[2] D. Koller, N. Friedman, Probabilistic graphical models: principles and techniques, MIT Press, 2009. 1

[3] A. Arnab, S. Zheng, S. Jayasumana, et al., Conditional random fields meet deep neural networks for semantic segmentation: combining probabilistic graphical models with deep learning for structured prediction, IEEE Signal Processing Magazine 35 (1) (2018) 37–52. doi:10.1109/MSP.2017.2762355. 1

[4] C. R. Qi, H. Su, K. Mo, et al., Pointnet: deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 652–660. doi:10.1109/cvpr.2017.16. 1, 2.1, 5.1, 5.3, 1, 5.4.1, 2, 6

[5] Y. Li, R. Bu, M. Sun, et al., Pointcnn: convolution on χ-transformed points, in: Advances in Neural Information Processing Systems, 2018, pp. 820–830. 1, 2.1, 4.1, 5.3, 1, 5.4.1, 2

[6] H. Thomas, C. R. Qi, J.-E. Deschaud, et al., Kpconv: flexible and deformable convolution for point clouds, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6411–6420. doi:10.1109/iccv.2019.00651. 1, 2.1, 4.1, 5.3, 1, 5.4.1, 2, 3, 6

[7] Y. Guo, H. Wang, Q. Hu, et al., Deep learning for 3d point clouds: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (Early Access)doi:10.1109/TPAMI.2020.3005434. 2.1

[8] F. Yang, H. Wang, Z. Jin, A fusion network for road detection via spatial propagation and spatial transformation, Pattern Recognition 100 (2020) 107141. doi:https://doi.org/10.1016/j.patcog.2019.107141. 2.1

[9] B. Wu, A. Wan, X. Yue, et al., Squeezeseg: convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud, in: Proceedings of the 2018 IEEE International Conference on Robotics and Automation, 2018, pp. 1887–1893. doi:10.1109/ICRA.2018.8462926. 2.1, 2.2

[10] E. Kalogerakis, M. Averkiou, S. Maji, et al., 3d shape segmentation with projective convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3779–3788. doi:10.1109/cvpr.2017.702. 2.1

[11] D. Maturana, S. Scherer, Voxnet: a 3d convolutional neural network for real-time object recognition, in: Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015, pp. 922–928. doi:10.1109/iros.2015.7353481. 2.1

[12] G. Riegler, A. Osman Ulusoy, A. Geiger, Octnet: learning deep 3d representations at high resolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3577–3586. doi:10.1109/cvpr.2017.701. 2.1

[13] R. Klokov, V. Lempitsky, Escape from cells: deep kd-networks for the recognition of 3d point cloud models, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 863–872. doi:10.1109/iccv.2017.99. 2.1, 1

[14] C. R. Qi, L. Yi, H. Su, et al., Pointnet++: deep hierarchical feature learning on point sets in a metric space, in: Advances in Neural Information Processing Systems, 2017, pp. 5099–5108. 2.1, 5.3, 1, 5.4.1, 6

[15] P. Hermosilla, T. Ritschel, P.-P. Vázquez, et al., Monte carlo convolution for learning on non-uniformly sampled point clouds, ACM Transactions on Graphics 37 (6) (2018) 1–12. doi:10.1145/3272127.3275110. 2.1, 4.1

[16] W. Wu, Z. Qi, L. Fuxin, Pointconv: deep convolutional networks on 3d point clouds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9621–9630. doi:10.1109/cvpr.2019.00985. 2.1, 4.1

[17] M. Feng, L. Zhang, X. Lin, et al., Point attention network for semantic segmentation of 3d point clouds, Pattern Recognition 107 (2020) 107446. doi:https://doi.org/10.1016/j.patcog.2020.107446. 2.1
[18] H. Zhou, Z. Fang, Y. Gao, et al., Feature fusion network based on attention mechanism for 3d semantic segmentation of point clouds, Pattern Recognition Letters 133 (2020) 327–333. doi:https://doi.org/10.1016/j.patrec.2020.03.021. 2.1
[19] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4558–4567. doi:10.1109/cvpr.2018.00479. 2.1
[20] Y. Wang, Y. Sun, Z. Liu, et al., Dynamic graph cnn for learning on point clouds, ACM Transactions on Graphics 38 (5) (2019) 1–12. doi:10.1145/3326362. 2.1, 1
[21] G. Li, M. Muller, A. Thabet, et al., Deepgcns: can gcns go as deep as cnns?, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9267–9276. doi:10.1109/iccv.2019.00936. 2.1
[22] M. M. Bronstein, J. Bruna, Y. LeCun, et al., Geometric deep learning: going beyond euclidean data, IEEE Signal Processing Magazine 34 (4) (2017) 18–42. doi:10.1109/msp.2017.2693418. 2.1
[23] P. Krähenbühl, V. Koltun, Efficient inference in fully connected crfs with gaussian edge potentials, in: Advances in Neural Information Processing Systems, 2011, pp. 109–117. 2.2, 4.3, 4.3, 4.3
[24] L.-C. Chen, G. Papandreou, I. Kokkinos, et al., Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (4) (2017) 834–848. doi:10.1109/tpami.2017.2699184. 2.2, 4.1
[25] S. Zheng, S. Jayasumana, B. Romera-Paredes, et al., Conditional random fields as recurrent neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1529–1537. doi:10.1109/iccv.2015.179. 2.2
[26] W. Qiu, X. Gao, B. Han, Saliency detection using a deep conditional random field network, Pattern Recognition 103 (2020) 107266. doi:https://doi.org/10.1016/j.patcog.2020.107266. 2.2
[27] R. Vemulapalli, O. Tuzel, M.-Y. Liu, et al., Gaussian conditional random field network for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3224–3233. doi:10.1109/cvpr.2016.351. 2.2
[28] J. Gilmer, S. S. Schoenholz, P. F. Riley, et al., Neural message passing for quantum chemistry, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 1263–1272. 3.2
[29] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the International Conference on Learning Representations, 2017. 3.2
[30] P. Veličković, G. Cucurull, A. Casanova, et al., Graph attention networks, in: Proceedings of the International Conference on Learning Representations, 2018. 3.2
[31] F. Monti, D. Boscaini, J. Masci, et al., Geometric deep learning on graphs and manifolds using mixture model cnns, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5115–5124. doi:10.1109/cvpr.2017.576. 3.2
[32] S. Wang, S. Suo, W.-C. Ma, et al., Deep parametric continuous convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2589–2597. doi:10.1109/cvpr.2018.00274. 4.1, 2
[33] A. Paszke, S. Gross, F. Massa, et al., Pytorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8026–8037. 5
[34] M. Fey, J. E. Lenssen, Fast graph representation learning with pytorch geometric, arXiv preprint arXiv:1903.02428. 5
[35] L. Yi, V. G. Kim, D. Ceylan, et al., A scalable active framework for region annotation in 3d shape collections, ACM Transactions on Graphics 35 (6) (2016) 1–12. doi:10.1145/2980179.2980238. 5.1
[36] I. Armeni, O. Sener, A. R. Zamir, et al., 3d semantic parsing of large-scale indoor spaces, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1534–1543. doi:10.1109/cvpr.2016.170. 5.1
[37] T. Hackel, N. Savinov, L. Ladicky, et al., Semantic3d.net: a new large-scale point cloud classification benchmark, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences IV-1/W1 (2017) 91–98. doi:10.5194/isprs-annals-iv-1-w1-91-2017. 5.1
[38] L. Yi, H. Su, X. Guo, et al., Syncspeccnn: synchronized spectral cnn for 3d shape segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2282–2290. doi:10.1109/cvpr.2017.697. 1
[39] Y. Shen, C. Feng, Y. Yang, et al., Mining point cloud local structures by kernel correlation and graph pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4548–4557. doi:10.1109/cvpr.2018.00478. 1
[40] J. Li, B. M. Chen, G. Hee Lee, So-net: self-organizing network for point cloud analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9397–9406. doi:10.1109/cvpr.2018.00979. 1
[41] Q. Huang, W. Wang, U. Neumann, Recurrent slice networks for 3d segmentation of point clouds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2626–2635. doi:10.1109/cvpr.2018.00278. 1
[42] M. Atzmon, H. Maron, Y. Lipman, Point convolutional neural networks by extension operators, ACM Transactions on Graphics 37 (4) (2018) 1–12. doi:10.1145/3197517.3201301. 1, 2
[43] Y. Xu, T. Fan, M. Xu, et al., Spidercnn: deep learning on point sets with parameterized convolutional filters, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 87–102. doi:10.1007/978-3-030-01237-3_6. 1
[44] W. Wang, R. Yu, Q. Huang, et al., Sgpn: similarity group proposal network for 3d point cloud instance segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2569–2578. doi:10.1109/cvpr.2018.00272. 1, 2, 3
[45] Y. Liu, B. Fan, S. Xiang, et al., Relation-shape convolutional neural network for point cloud analysis, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8895–8904. doi:10.1109/cvpr.2019.00910. 1
[46] Z.-H. Lin, S.-Y. Huang, Y.-C. F. Wang, Convolution in the cloud: learning deformable kernels in 3d graph convolution networks for point cloud analysis, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1800–1809. doi:10.1109/cvpr42600.2020.00187. 1
[47] M. Xu, R. Ding, H. Zhao, et al., Paconv: position adaptive convolution with dynamic kernel assembling on point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3173–3182. 1, 2
[48] Q. Hu, B. Yang, L. Xie, et al., Randla-net: efficient semantic segmentation of large-scale point clouds, in: Proceedings of the IEEE/CVF

Conference on Computer Vision and Pattern Recognition, 2020, pp. 11108–11117. doi:10.1109/cvpr42600.2020.01112. 5.4.1, 3, 5, 5.5, 6

[49] L. Tchapmi, C. Choy, I. Armeni, et al., Segcloud: semantic segmentation of 3d point clouds, in: Proceedings of the 2017 International Conference on 3D vision, 2017, pp. 537–547. doi:10.1109/3dv.2017.00067. 2, 3

[50] C. Zhang, W. Luo, R. Urtasun, Efficient convolutions for real-time semantic segmentation of 3d point clouds, in: Proceedings of the 2018 International Conference on 3D Vision, 2018, pp. 399–408. doi:10.1109/3dv.2018.00053. 2

[51] M. Tatarchenko, J. Park, V. Koltun, et al., Tangent convolutions for dense prediction in 3d, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3887–3896. doi:10.1109/cvpr.2018.00409. 2

[52] X. Ye, J. Li, H. Huang, et al., 3d recurrent neural networks with context fusion for point cloud semantic segmentation, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 415–430. doi:10.1007/978-3-030-01234-2_25. 2

[53] H. Zhao, L. Jiang, C.-W. Fu, et al., Pointweb: enhancing local neighborhood features for point cloud processing, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5565–5573. doi:10.1109/cvpr.2019.00571. 2

[54] Y. Lin, Z. Yan, H. Huang, et al., Fpconv: learning local flattening for point convolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4293–4302. doi:10.1109/cvpr42600.2020.00435. 2

[55] X. Yan, C. Zheng, Z. Li, et al., Pointasnl: robust point clouds processing using nonlocal neural networks with adaptive sampling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 5589–5598. doi:10.1109/cvpr42600.2020.00563. 2

[56] S. Qiu, S. Anwar, N. Barnes, Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1757–1767. 2, 3

[57] A. Boulch, B. L. Saux, N. Audebert, Unstructured point cloud semantic labeling using deep segmentation networks, in: Proceedings of the Eurographics Workshop on 3D Object Retrieval, 2017. doi:10.2312/3dor.20171047. 3

[58] H. Thomas, F. Goulette, J.-E. Deschaud, et al., Semantic classification of 3d point clouds with multiscale spherical neighborhoods, in: Proceedings of the 2018 International Conference on 3D Vision, 2018, pp. 390–398. doi:10.1109/3dv.2018.00052. 3

[59] X. Roynard, J.-E. Deschaud, F. Goulette, Classification of point cloud for road scene understanding with multiscale voxel deep network, in: Proceedings of the 10th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, 2018. 3

[60] Z. Zhang, B.-S. Hua, S.-K. Yeung, Shellnet: efficient point cloud convolutional neural networks using concentric shells statistics, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1607–1616. doi:10.1109/iccv.2019.00169. 3

[61] L. Wang, Y. Huang, Y. Hou, et al., Graph attention convolution for point cloud semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10296–10305. doi:10.1109/cvpr.2019.01054. 3

[62] Y. Ma, Y. Guo, H. Liu, et al., Global context reasoning for semantic segmentation of 3d point clouds, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 2931–2940. doi:10.1109/wacv45572.2020.9093411. 3

[63] S. Fan, Q. Dong, F. Zhu, et al., Scf-net: learning spatial contextual features for large-scale point cloud segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14504–14513. 3

Supplementary Material of Continuous Conditional Random Field Convolution for Point Cloud Segmentation

Fei Yang[a,b], Franck Davoine[c], Huan Wang[b], Zhong Jin[a,b,*]

[a]*Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing 210094, China*
[b]*School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[c]*Alliance Sorbonne Université, Université de technologie de Compiègne, CNRS, Heudiasyc Lab., CS 60319 - F-60203 Compiègne Cedex, France*

Abstract

This document provides some supplementary contents for the paper: Continuous Conditional Random Field Convolution for Point Cloud Segmentation. Section 1 of this document presents some details about the proof of Theorem 1. The model-level analysis of the relationship between the proposed CRFConv and previous diffusion-based methods is given in section 2. More experimental details and results can be found in section 3.

1. The proof of Theorem 1

Problem statement:

Given a continuous CRF defined on graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, its corresponding Gibbs distribution can be written as:

$$P(X) = \frac{1}{Z}\exp(-E(X)), \tag{1}$$

where the $X = \{\mathbf{x}_i \in \mathbb{R}^d | i \in \mathcal{V}\}$ denotes the set of all latent features, Z is a partition function to ensure a distribution. The energy function $E(X)$ is defined as a continuous quadratic energy model:

$$E(X) = \sum_{i\in\mathcal{V}}(\mathbf{x}_i - \mathbf{z}_i)^\top(\mathbf{x}_i - \mathbf{z}_i) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\mathbf{x}_i - \mathbf{x}_j)^\top\mathbf{w}_{ij}(\mathbf{x}_i - \mathbf{x}_j), \tag{2}$$

where $\mathbf{z}_i$ denotes the observed feature of node i. $\mathcal{N}(i)$ denotes the neighbor nodes set of node i in $\mathcal{G}$. $\mathbf{w}_{ij} \in \mathbb{R}^{d\times d} \succ \mathbf{0}$ is the weight matrix of two connected nodes in $\mathcal{G}$.

Let $Q(X) = \prod_i Q_i(\mathbf{x}_i)$ be the approximate distribution of $P(\mathbf{x})$, where $Q_i \sim N(\mu_i, \mathbf{\Sigma}_i)$. Our object is to obtain $\{(\mu_i^*, \mathbf{\Sigma}_i^*)|i \in \mathcal{V}\}$ by solving the following problem:

$$\{(\mu_i^*, \mathbf{\Sigma}_i^*)|i \in \mathcal{V}\} = \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin}\ KL(Q\|P), \tag{3}$$

where

$$KL(Q\|P) = \int Q(X)\log\frac{Q(X)}{P(X)}dX \tag{4}$$

is the KL-divergence between $Q(X)$ and $P(X)$.

*Corresponding author
Email addresses: yangfei92516@163.com (Fei Yang), franck.davoine@hds.utc.fr (Franck Davoine), wanghuanphd@njust.edu.cn (Huan Wang), zhongjin@njust.edu.cn (Zhong Jin)

Solution:

$$
\begin{aligned}
\{(\mu_i^*, \mathbf{\Sigma}_i^*)|i \in \mathcal{V}\} &= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin}\ KL(Q\|P) \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} \int Q(X)\log Q(X)dX - \int Q(X)\log P(X)dX \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} \sum_{i\in\mathcal{V}} \int Q_i(\mathbf{x}_i)\log Q_i(\mathbf{x}_i)d\mathbf{x}_i + \int Q(X)E(X)dX + \underbrace{\log Z}_{\text{constant}} \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\sum_{i\in\mathcal{V}} \mathbb{H}_{Q_i}[\mathbf{x}_i] + \mathbb{E}_{X\sim Q}[E(X)] \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\frac{1}{2}\sum_{i\in\mathcal{V}} \Big(\log|\mathbf{\Sigma_i}| + \underbrace{d(\log 2\pi + 1)}_{\text{constant}}\Big) + \mathbb{E}_{X\sim Q}[E(X)] \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\frac{1}{2}\sum_{i\in\mathcal{V}} \log|\mathbf{\Sigma_i}| + \sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathbf{x}_i^\top \mathbf{x}_i] - 2\sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathbf{z}_i^\top \mathbf{x}_i] + \underbrace{\sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathbf{z}_i^\top \mathbf{z}_i]}_{\text{constant}} \\
&+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathbf{x}_i^\top \mathbf{w}_{ij}\mathbf{x}_i] - 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathbf{x}_i^\top \mathbf{w}_{ij}\mathbf{x}_j] + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathbf{x}_j^\top \mathbf{w}_{ij}\mathbf{x}_j] \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\frac{1}{2}\sum_{i\in\mathcal{V}} \log|\mathbf{\Sigma_i}| + \sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_i^\top \mathbf{x}_i)] - 2\sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathbf{z}_i^\top \mathbf{x}_i] + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_i^\top \mathbf{w}_{ij}\mathbf{x}_i)] \\
&- 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_i^\top \mathbf{w}_{ij}\mathbf{x}_j)] + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_j^\top \mathbf{w}_{ij}\mathbf{x}_j)] \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\frac{1}{2}\sum_{i\in\mathcal{V}} \log|\mathbf{\Sigma_i}| + \sum_{i} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_i\mathbf{x}_i^\top)] - 2\sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathbf{z}_i^\top \mathbf{x}_i] + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_i\mathbf{x}_i^\top \mathbf{w}_{ij})] \\
&- 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_j\mathbf{x}_i^\top \mathbf{w}_{ij})] + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathbb{E}_{x\sim Q}[\mathrm{tr}(\mathbf{x}_j\mathbf{x}_j^\top \mathbf{w}_{ij})] \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\frac{1}{2}\sum_{i\in\mathcal{V}} \log|\mathbf{\Sigma_i}| + \sum_{i\in\mathcal{V}} \mathrm{tr}(\mathbb{E}_{x\sim Q}[\mathbf{x}_i\mathbf{x}_i^\top]) - 2\sum_{i\in\mathcal{V}} \mathbb{E}_{x\sim Q}[\mathbf{z}_i^\top \mathbf{x}_i] + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}(\mathbb{E}_{x\sim Q}[\mathbf{x}_i\mathbf{x}_i^\top]\mathbf{w}_{ij}) \\
&- 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}(\mathbb{E}_{x\sim Q}[\mathbf{x}_j\mathbf{x}_i^\top]\mathbf{w}_{ij}) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}(\mathbb{E}_{x\sim Q}[\mathbf{x}_j\mathbf{x}_j^\top]\mathbf{w}_{ij}) \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} -\frac{1}{2}\sum_{i\in\mathcal{V}} \log|\mathbf{\Sigma_i}| + \sum_{i\in\mathcal{V}} \mathrm{tr}(\mathbf{\Sigma}_i + \mu_i\mu_i^\top) - 2\sum_{i\in\mathcal{V}} \mathbf{z}_i^\top \mu_i + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}((\mathbf{\Sigma}_i + \mu_i\mu_i^\top)\mathbf{w}_{ij}) \\
&- 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}((\mu_j\mu_i^\top)\mathbf{w}_{ij}) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}((\mathbf{\Sigma}_j + \mu_j\mu_j^\top)\mathbf{w}_{ij}) \\
&= \underset{\{(\mu_i,\mathbf{\Sigma}_i)|i\in\mathcal{V}\}}{argmin} \underbrace{\Big(\sum_{i\in\mathcal{V}} \mu_i^\top\mu_i - 2\sum_{i\in\mathcal{V}} \mathbf{z}_i^\top\mu_i + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mu_i^\top \mathbf{w}_{ij}\mu_i - 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mu_i^\top \mathbf{w}_{ij}\mu_j + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mu_j^\top \mathbf{w}_{ij}\mu_j\Big)}_{\text{temrs consist of } \{\mu_i|i\in\mathcal{V}\}} \\
&+ \underbrace{\Big(-\frac{1}{2}\sum_{i\in\mathcal{V}} \log|\mathbf{\Sigma_i}| + \sum_{i\in\mathcal{V}} \mathrm{tr}(\mathbf{\Sigma}_i) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}(\mathbf{\Sigma}_i\mathbf{w}_{ij}) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)} \mathrm{tr}(\mathbf{\Sigma}_j\mathbf{w}_{ij})\Big)}_{\text{temrs consist of } \{\mathbf{\Sigma}_i|i\in\mathcal{V}\}}.
\end{aligned}
\tag{5}
$$

To minimize μ_i and $\mathbf{\Sigma}_i$, we only need to concern about the terms which contains μ_i and $\mathbf{\Sigma}_i$ in Eq. (5), respectively. For μ_i, we have

$$\begin{aligned}\mu_i^* &= \underset{\mu_i}{argmin}\, \mu_i^\top \mu_i - 2\mathbf{z}_i^\top \mu_i + \sum_{j\in\mathcal{N}(i)} \mu_i^\top \mathbf{w}_{ij}\mu_i - 2\sum_{j\in\mathcal{N}(i)} \mu_i^\top \mathbf{w}_{ij}\mu_j \\ &= \underset{\mu_i}{argmin}\, \mu_i^\top \Big(\mathbf{I} + \sum_{j\in\mathcal{N}(i)} \mathbf{w}_{ij}\Big)\mu_i - 2\mathbf{z}_i^\top \mu_i - 2\mu_i^\top \sum_{j\in\mathcal{N}(i)} \mathbf{w}_{ij}\mu_j\end{aligned} . \tag{6}$$

For $\mathbf{\Sigma}_i$, we have

$$\begin{aligned}\mathbf{\Sigma}_i^* &= \underset{\mathbf{\Sigma}_i}{argmin} -\frac{1}{2}\log|\mathbf{\Sigma_i}| + \mathrm{tr}(\mathbf{\Sigma}_i) + \sum_{j\in\mathcal{N}(i)} \mathrm{tr}(\mathbf{\Sigma}_i\mathbf{w}_{ij}) \\ &= \underset{\mathbf{\Sigma}_i}{argmin}\, \mathrm{tr}\Big(\mathbf{\Sigma}_i(\mathbf{I} + \sum_{j\in\mathcal{N}(i)} \mathbf{w}_{ij})\Big) - \frac{1}{2}\log|\mathbf{\Sigma_i}|\end{aligned} \tag{7}$$

Compute the gradients of Eq. (6) and Eq. (7) with respect to μ_i and $\mathbf{\Sigma}_i$ respectively and let the gradients to be zero, we can get the update equations of μ_i^* and $\mathbf{\Sigma}_i^*$:

$$\mu_i^* = \Big(\mathbf{I} + \sum_{j\in\mathcal{N}(i)} \mathbf{w}_{ij}\Big)^{-1}\Big(\mathbf{z}_i + \sum_{j\in\mathcal{N}(i)} \mathbf{w}_{ij}\mu_j\Big) \tag{8}$$

$$\mathbf{\Sigma}_i^* = \frac{1}{2}\Big(\mathbf{I} + \sum_{j\in\mathcal{N}(i)} \mathbf{w}_{ij}\Big)^{-1} \tag{9}$$

To maximize $Q(X)$, we only need to compute μ_i for all $Q_i(\mathbf{x}_i)$. It can be found that the update equation of μ_i in the mean-field approximation is the same as that of $\mathbf{x}_i$ in the coordinate descent algorithm, by which Theorem 1 can be proven.

2. The relationship with diffusion

Consider an anisotropic diffusion process on a heat field $\mathcal{H}$ defined on graph $\mathcal{G}_\mathcal{H}$. Denote $\mathbf{h}_t$ as the heat vector of $\mathcal{H}$ at time t on $\mathcal{G}_\mathcal{H}$. And let $\mathbf{L}_\mathcal{H} = \mathbf{I} - \mathbf{D}_\mathcal{H}^{-1}\mathbf{W}_\mathcal{H}$ be the normalized Laplacian matrix of $\mathcal{G}_\mathcal{H}$, where $\mathbf{D}_\mathcal{H}$ and $\mathbf{W}_\mathcal{H}$ are the degree and adjacent matrix, respectively. The diffusion process on $\mathcal{G}_\mathcal{H}$ can be described as the following PDE with a discrete form:

$$\mathbf{h}^{t+1} - \mathbf{h}^t = -c\mathbf{L}_\mathcal{H}\mathbf{h}^t, \tag{10}$$

where c is a diffusion coefficient to control the diffusion speed.

In the diffusion process, the heat field is a scalar field. By unfolding Eq. (10), we can obtain the temperature h_i^t in the heat field $\mathcal{H}$ for any position i and time t:

$$\begin{aligned}h_i^{t+1} &= h_i^t - c\sum_{j\in\mathcal{N}(i)} w_{ij}(h_i^t - h_j^t) \\ &= \frac{1}{2}(h_i^t + \sum_{j\in\mathcal{N}(i)} w_{ij}h_j^t) \qquad (\text{by setting } c = \frac{1}{2}),\end{aligned} \tag{11}$$

where w_{ij} is the weight of two adjacent nodes. If we set the compatibility transform matrix $\mathbf{C} = \mathbf{I}$ in our model, that is, each channel is independent of others, we can find that our CRFConv is equivalent to a diffusion model built on each channel individually at time $t = 0$. However, when $t > 0$, the difference between the diffusion process and our CRFConv is that the diffusion process adds the previous state at each iteration while our CRFConv always adds the initial state at each iteration.

Furthermore, when the diffusion process reaches the stable state, we have $\mathbf{L}_\mathcal{H}\mathbf{h} = \mathbf{0}$, which is the solution of

$$\mathbf{h}^* = \underset{\mathbf{h}}{argmin}\, \mathbf{h}^\top \mathbf{L}_\mathcal{H}\mathbf{h}, \tag{12}$$

where $\mathbf{h}^{\mathrm{T}}\mathbf{L}_{\mathcal{H}}\mathbf{h}$ is the Dirichlet energy. It can be found that the Dirichlet energy only acts as the smoothness term of our CRF model. That is to say, the diffusion-based methods only ensure the smoothness but ignore the uniqueness of features, which could cause over smoothness, while the proposed method does. In other words, our CRFConv can overcome the over smoothness phenomenon usually occurred in current popular graph convolution methods. Moreover, the diffusion-based methods suppose that each channel is independent of the others, while our CRFConv models the relations between each channel by a compatibility transformation matrix $\mathbf{C}$ which can be learned from the data automatically to improve the representation ability of the model.

3. More experimental details and results

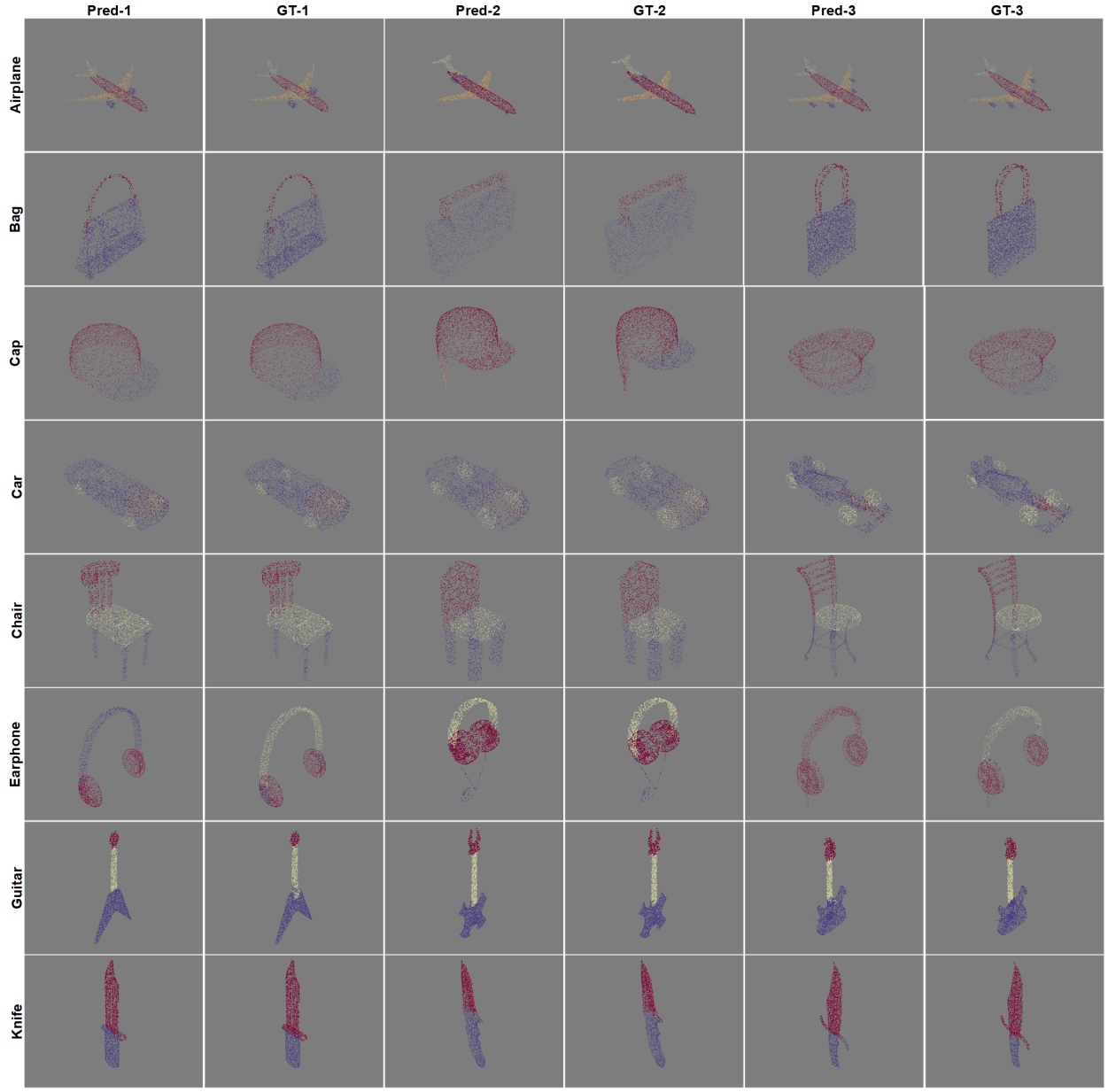
3.1. Implementation details

The details of the point cloud segmentation network can be found in Table 1. Note that the Batch Normalization with the momentum of 0.98 and the LeakyReLU with the negative slope of 0.1 are added after each convolution layer. For brevity, we did not show them in Table 1. We construct the point cloud graph based on kNN in both the encoder and the decoder. The dilated kNN is only available for the PointConv in the encoder. The momentum stochastic gradient descent (SGD) algorithm is used to minimize the cross entropy-loss, with a momentum of 0.95. We take an exponential decay strategy for the learning rate with initial rate of 0.01. The decay rate is set to ensure the learning rate is divided by 10 every 50 epochs. The network will reach convergence within 100 epochs. The batch sizes are set to 16 and 8 for the conventional and advanced (with precomputed multiscale strategy) implementations, respectively.

Table 1: Detailed architecture of the proposed point cloud segmentation network.

Modules		**Layers**
Encoder	**conv_1**	PointConv(in_dims=F, out_dims=64,kernel_size=32, dil=1, ratio=1.0) PointConv(in_dims=64, out_dims=64, kernel_size=32, dil=1, ratio=1.0)
	conv_2	PointConv(in_dims=64, out_dims=128, kernel_size=16, dil=2, ratio=0.25) PointConv(in_dims=128, out_dims=128, kernel_size=16, dil=2, ratio=1.0)
	conv_3	PointConv(in_dims=128, out_dims=256, kernel_size=16, dil=4, ratio=0.375) PointConv(in_dims=256, out_dims=256, kernel_size=16, dil=4, ratio=1.0)
	conv_4	PointConv(in_dims=256, out_dims=512, kernel_size=16, dil=4, ratio=0.375) PointConv(in_dims=512, out_dims=512, kernel_size=16, dil=4, ratio=1.0)
	conv_5	PointConv(in_dims=512, out_dims=1024, kernel_size=16, dil=2, ratio=0.375) PointConv(in_dims=1024, out_dims=1024, kernel_size=16, dil=2, ratio=1.0)
Decoder	**deconv_4**	CRFConv(unary_dims=1024, pairwise_dims=512, out=512, kernel_size=16)
	deconv_3	Linear(in_dims=512+512, out_dims=512) CRFConv(unary_dims=512, pairwise_dims=256, out_dims=256, kernel_size=16)
	deconv_2	Linear(in_dims=256+256, out_dims=256) CRFConv(unary_dims=256, pairwise_dims=128, out_dims=128, kernel_size=16)
	deconv_1	Linear(in_dims=128+128, out_dims=128) CRFConv(unary_dims=128, pairwise_dims=64, out_dims=64, kernel_size=16)
Classifier	**fc_1**	Linear(in_dims=64+64, out_dims=256) LeakyReLU(negative_slope=0.1)
	fc_2	Linear(in_dims=256, out_dims=L) Softmax(dim=-1)
Discrete CRF (optional)		DiscreteCRFConv(in_dims=L, out_dims=L, kernel_size=32)

3.2. *Some visualization results*



(a)

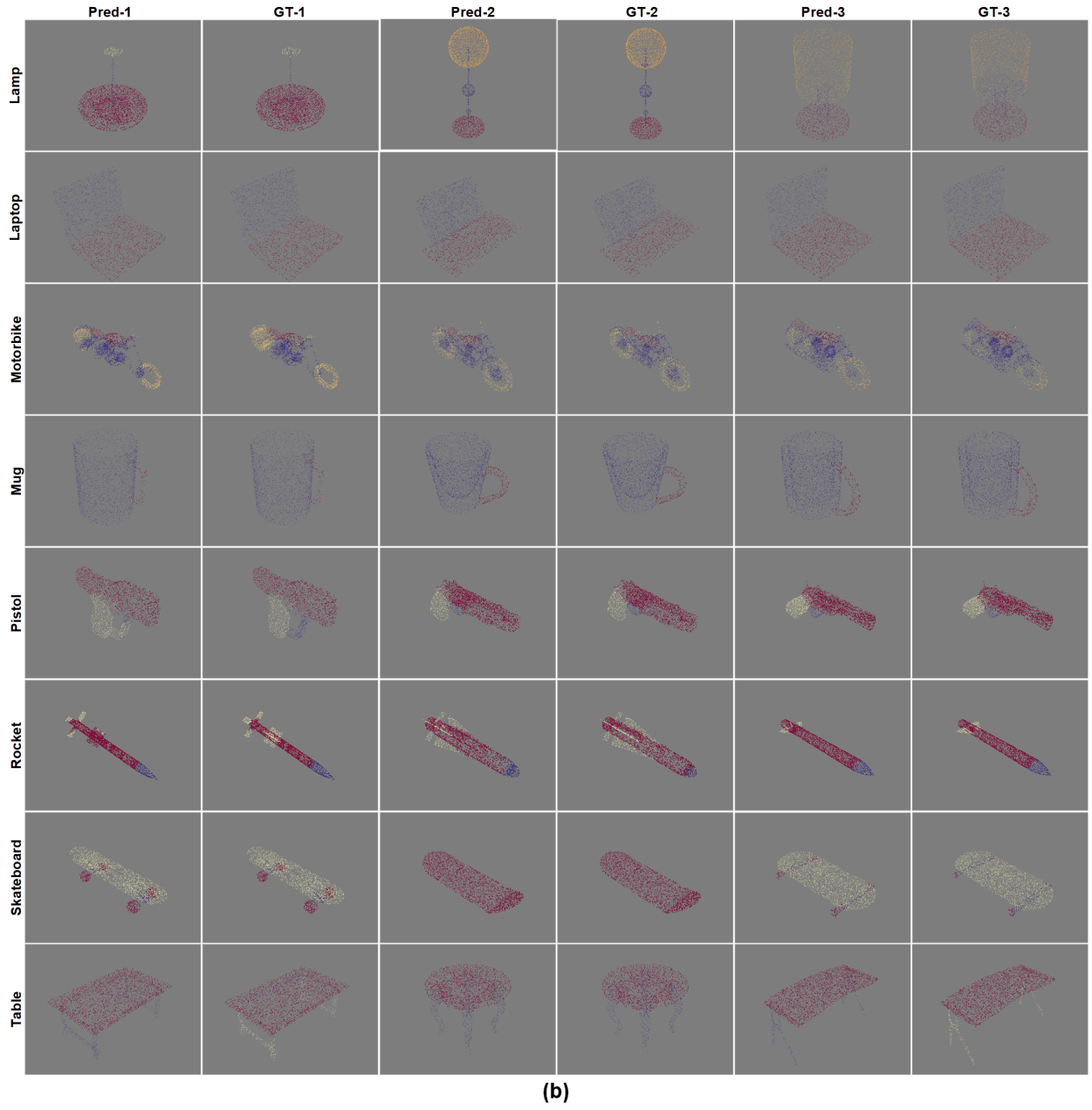


Figure 1: Visualization results compared with the groundtruth on the ShapeNet Part dataset. Three samples are shown for each object category, in which each part is visualized in a unique color.

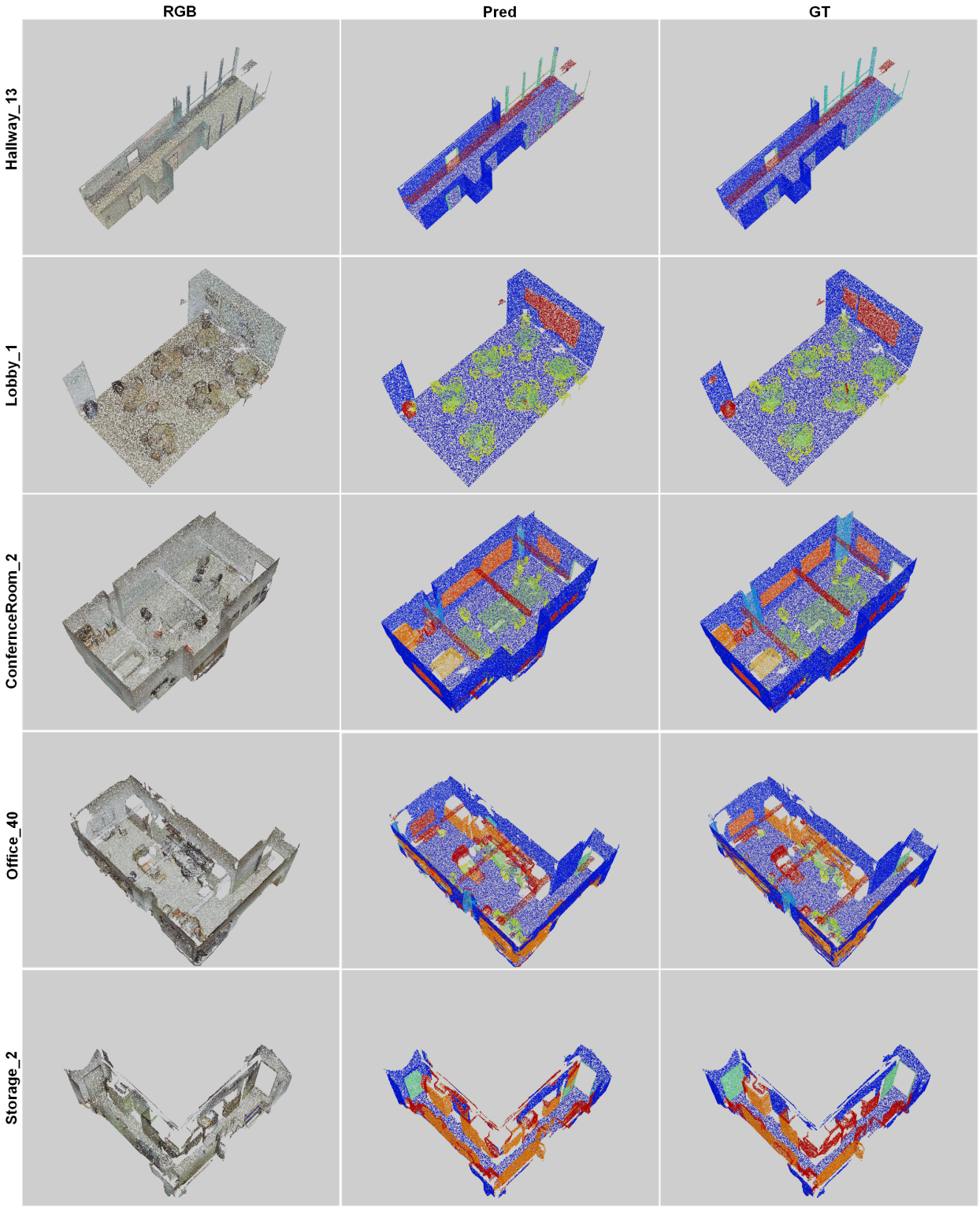


Figure 2: Visualization results compared with the groundtruth on Area 5 of the S3DIS dataset. Each class is visualized in a unique color. Note that the ”ceiling” category is removed manually for better visualization.

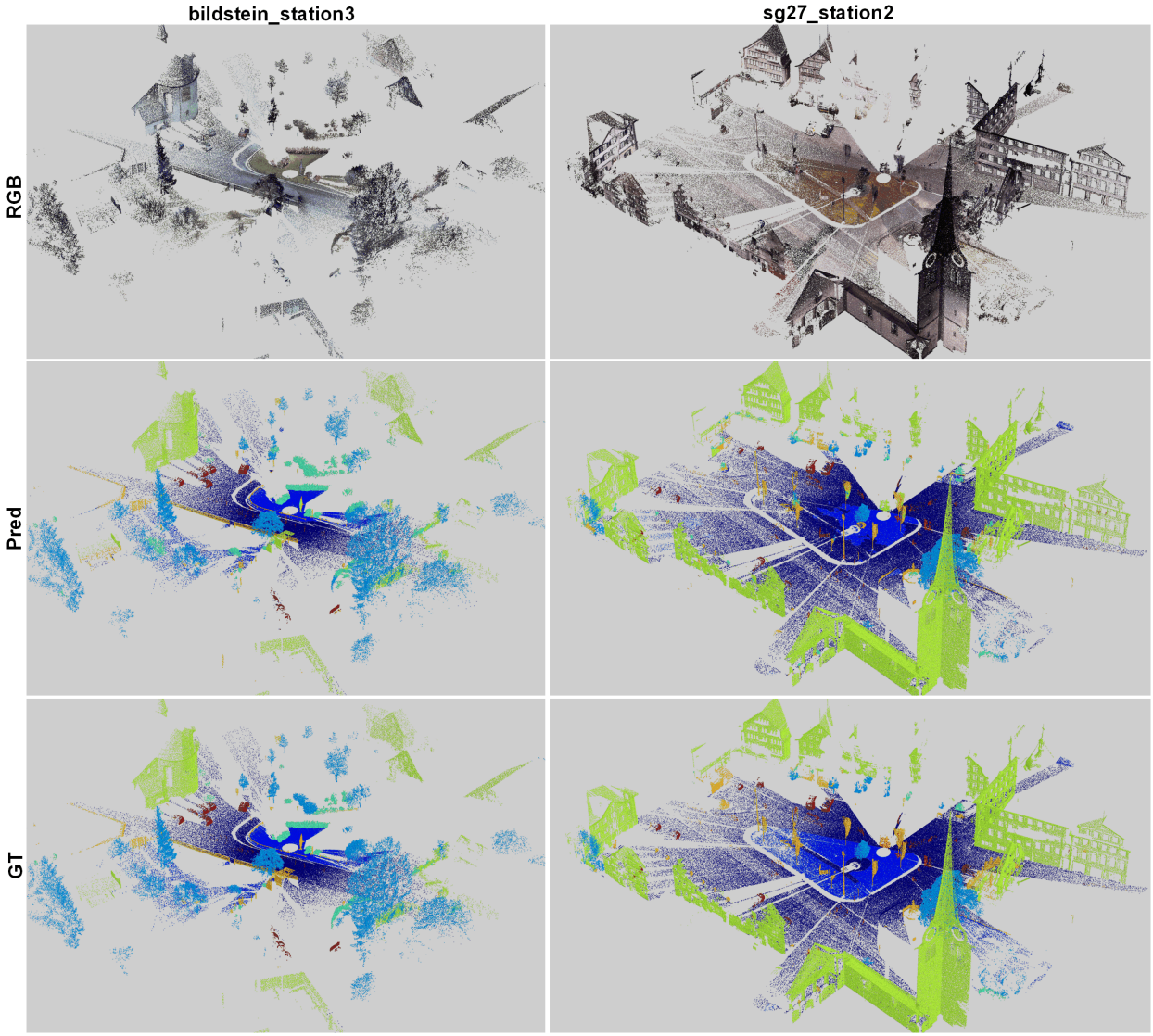


Figure 3: Visualization results compared with the groundtruth on the validation set of the Semantic3D dataset. Each class is visualized in a unique color. Note that the ”unlabeled” category is removed manually for better visualization.