UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Research on Point Cloud Segmentation and 3D Scene Understanding

by

Anan Du

Supervisor: Jian Zhang
Co-Supervisor: Qiang Wu

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

April, 2022

Certificate of Authorship/Originality

I, Anan Du declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering/Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 17/04/2022

Acknowledgements

During my doctoral study in UTS, I have received a lot of support from my supervisors, colleagues, and family. Here, I would like to express my sincere gratitude to them.

First and foremost, I would like to thank my Ph.D supervisor, Prof. Jian Zhang, for his guidance and support throughout my research and completion of this thesis. Prof. Zhang Jian has been an academic role model to me. He is rigorous and enthusiastic in work, has deep insights in academic, has rich skills on research writing and presentation and always keeps curiosity on new knowledge. All of these characters have guided me and will continue encouraging me to be a professional researcher. I also want to thank my co-supervisor, Associate Prof. Qiang Wu, for his constructive guidance regarding my research directions. Besides, his optimistic attitude to work and life has inspired me a lot.

I would like to thank my colleague, Dr. Xiaoshui Huang, who is also engaged in research on 3D point clouds. Dr. Xiaoshui has given me a lot of professional guidance during my entire Ph.D. study, from introducing me to the basic knowledge about 3D point cloud, to discussing my follow-up research works, to writing articles. He always works hard and efficiently. He also has a wide range of scientific research fields and often has novel ideas. It is an honor to work with him. I want to thank my co-author Dr. Shuchao Pang for supporting my study and life. In my Ph.D study, we often shared the articles we read, discussed our views, debugged the experiments, and polished each other's articles. In life, when I encounter setbacks, Dr. Shuchao provided me a lot of encouragement. I also want to thank my research collaborator Dr. Tianfei Zhou, for his help to my research. He has given me a lot of support in the method design, validation, and article writing. I also appreciate my colleague, Dr. Lingxiang Yao, for his help in polishing the articles and guidance in doing the

presentation with his excellent communication skills.

I want to thank all my colleagues supervised by Prof. Jian and Associate Prof. Qiang during my Ph.D study. Thanks to Dr. Jinsong Xu, Dr. Junjie Zhang, Dr. Muming Zhang, Dr. Lu Zhang, Dr. Yongshun Gong, Dr. Zhibin Li, Dr. Huaxi Huang, Dr. Litao Yu, Dr. Zongjian Zhang, Dr. Guofeng Mei, Dr. Wenbo Xu, Dr. Yan Huang, Dr. Peng Zhang, Dr. Qiang Li, Dr. Xunxiang Yao. I also thank all other colleagues I have studied with. They make my study and life in UTS beautiful.

At last, I want to thank my parents, brother and sister. They always show their patience, trust and love to me. I love them.

Anan Du

Sydney, Australia, 2022.

ABSTRACT

Recently, the amount of 3D data has been sharply increasing thanks to widely available 3D sensors like Lidar, Kinect and RealSense. Compared to 2D images, 3D data provides clear topology and geometric information, which is very important for many computer vision applications. Among many types of 3D data, the point cloud is widely used because of its easy availability and storage. This thesis summaries the works that have been conducted on understanding the high-level semantics and basic structure of 3D scenes based on point cloud data.

Firstly, a fully-supervised point cloud semantic segmentation framework is designed. Contextual information can help resolve ambiguity and improve the robustness of a recognition system. To capture long-range context, a long-short-term feature bank based framework is introduced to exploit the patch-wise relationship for point cloud semantic segmentation. This approach can capture context in an arbitrary range by costing a little extra computation than a standard segmentation model. Experiments demonstrate that the proposed approach outperforms other point cloud semantic segmentation approaches exploring long-range context.

Manually labeling point cloud datasets, especially point-wise annotations, for fully-supervised methods is expensive. To avoid manually annotating 3D keypoints, a weakly-supervised 3D keypoint extraction method for point cloud registration, called KPSNet, is proposed, which only uses the relative transformation matrices between input pairs of point clouds as weak labels to establish point-to-point correspondences on-the-fly for training. Moreover, KPSNet can simultaneously detect 3D keypoints and learn the representations. Experimental results reveal that our proposed method performs better in point cloud registration than other methods without keypoint annotations.

To cut the need for manual point-wise annotations, a weakly-supervised point

cloud semantic segmentation approach is proposed, which uses coarse labels only. The proposed approach reformulates this problem as a semi-supervised point cloud semantic segmentation problem with noisy pseudo labels. Then a three-branch network is proposed, which can reduce the impact of noises in pseudo labels and leverage the inner structure of point clouds to improve the segmentation performance. By evaluating on benchmark datasets, this method surpasses other weakly-supervised methods and fully-supervised method PointNet++, and narrows the gap with other outstanding fully-supervised approaches. Subsequently, another weakly-supervised point cloud semantic segmentation framework is introduced, which uses incomplete point-level labels. It explores contrastive learning with cross-sample contrast and low-level contrast and a pseudo label refinement module to mine more helpful supervision information from pseudo labels. This method gains state-of-the-art performance on benchmark datasets with several weakly-supervised annotation settings.

Keywords: Point cloud, semantic segmentation, 3D keypoint, weakly-supervised, 3D scene understanding

Contents

List of Figures

List of Tables

List of Publications

Conference Papers

1. **Anan Du**, Shuchao Pang, Xiaoshui Huang, Jian Zhang, Qiang Wu, "Exploring Long-Short-Term Context For Point Cloud Semantic Segmentation", In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 2755-2759. IEEE, 2020.
2. **Anan Du**, Xiaoshui Huang, Jian Zhang, Lingxiang Yao, Qiang Wu, "KPSNet: Keypoint Detection and Feature Extraction for Point Cloud Registration", In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2755-2759. IEEE, 2019.

Journal Papers

1. **Anan Du**, Shuchao Pang, Xiaoshui Huang, Jian Zhang, Qiang Wu, "Weakly-supervised Point Cloud Semantic Segmentation with Coarse-grained Labels." In *IEEE Transactions on Multimedia*. (Submitted)
2. **Anan Du**, Tianfei Zhou, Shuchao Pang, Jian Zhang, Qiang Wu, "Point Contrast and Labelling for Weakly-supervised Point Cloud Semantic Segmentation." In *IEEE Transactions on Image Processing*. (Under submission)

Chapter 1

Introduction

1.1 Background

Recently, the amount of 3D data has been increasing rapidly thanks to widely available 3D sensors like Lidar, Kinect and RealSense. Compared with 2D images, 3D data provides clear topology and geometric information, which is very important for many computer vision applications. Among many types of 3D data, the point cloud is widely used because of its easy availability and storage. As a fundamental task of 3D scene understanding, point cloud based semantic segmentation occupies a vital position in understanding 3D environments, which derives various applications, such as autonomous vehicles and robotics. It pursues predicting the semantic class of every point in a given point cloud, which presents numerous challenges. The most obvious one is that point clouds are inherently sparse in 3D space distributed only on object surfaces, making most training methods inefficient. Another challenge is that applying many end-to-end learning methods on point clouds is difficult owing to the unstructured characteristic of point clouds. Moreover, point clouds captured by Lidar are usually large-scale with various densities. These obstacles prevent Convolutional Neural Networks (CNNs) applied on point clouds from achieving the impressive performances attained for speech processing or image understanding. In addition, it is hard to discover, represent and explore the relationships between points, which also makes the point cloud segmentation task more difficult.

There are a few solutions to use deep learning techniques to 3D point cloud data by reviewing existing research. The first solution is to convert irregular point cloud

data to regular grid data such as 2D images and 3D voxels in order to directly borrow the successful CNNs operating on regular data. For instance, [92] uses 3D CNNs on voxelized point cloud data. [5] projects point clouds into collections of 2D images to use 2D image segmentation networks. The 2D segmentation results are finally mapped to the original 3D data. However, the processing of data format transformations is usually accompanied by quantization artifacts, leading to information loss, thereby limiting the representative ability of the models. More specifically, transforming point clouds into regular voxels tends to discard small details. This also adds a lot of redundancy which is inefficient, because point clouds only record information about objects' surfaces. Transforming point clouds into sets of images results in 3D structure information being inevitably lost and the inverse projection problem from 2D to 3D being introduced, which is another tough challenge. Moreover, using the standard 3D CNNs [92, 66] and 2D CNNs [73, 87] often requires a lot of computational and memory resources because of the huge amount of redundancy existing in the transformed voxels or images. The second solution is to use the raw point clouds directly without any format transformations. PointNet [72] is a seminal work in this line of methods. It learns a feature representation for each point of an input point cloud and then aggregates all point-wise feature representations into a global feature representation through max-pooling. Since PointNet processes all points independently, it cannot capture local structures. In order to exploit local geometric contexts, many improved works based on PointNet have been proposed to exploit the local structure in neighborhoods of points [74, 83]. However, there are limited prior works that consider capturing long-range spatial context. [24, 123] try to incorporate the larger spatial context, but they are limited by the computational resource and ignore the point-wise relationship.

Although recent deep learning methods have shown to be very valuable and achieved remarkable success on 3D computer vision tasks, these problems are still

not fully solved. The main reason is that deep neural networks require substantial annotated data, which is hard to obtain. In the case of dense prediction tasks like point cloud semantic segmentation and 3D keypoint extraction, annotations should be at the point level, which is highly laborious and time-consuming. Currently, only limited 3D point cloud datasets with annotation at the point level are publicly available. These challenges hinder fully-supervised point cloud understanding methods from being extended to more application scenarios. More recently, self-supervised learning and weakly-supervised learning for 3D computer vision tasks have attracted dramatically growing attention, but they are still in their infancy.

1.2 Research Challenges and Objectives

The goal of this thesis is to apply deep learning to point clouds, understanding the high-level semantics and basic structure of 3D scenes based on point cloud data. Specifically, this thesis focuses on point cloud registration and semantic segmentation tasks. Although many efforts have been made, 3D computer vision tasks are still not fully solved. The main challenges are as discussed below.

- The unordered and unstructured attributes of point cloud increase the difficulty of extracting contextual information, which eventually limits the performance of point cloud semantic segmentation approaches.
- Point clouds captured by Lidar are usually large scale with various densities, which increases the difficulty of capturing long-range context for point loud segmentation methods.
- Data-hungry fully-supervised methods increase the difficulty of extending dense prediction methods to more application scenarios.

Three research objectives are set forth in this thesis to address these challenges:

- The first objective is to exploit long-range contextual information under limited computational resources to improve the efficiency of point cloud semantic segmentation. We achieve this objective in Chapter 3.

- The second objective is to explore a weakly-supervised method to understand the basic structure of 3D scenes. Particularly, we aim to develop a weakly-supervised 3D keypoint extraction method for the point cloud registration task without using manually annotated keypoint labels. Chapter 4 achieves this goal.

- The third objective is to design weakly-supervised methods for point cloud semantic segmentation, reducing the dense annotation costs. Particularly, we aim to employ coarse cloud-level weak labels and incomplete point-level weak labels for point cloud semantic segmentation separately. This objective is achieved in Chapter 5 and Chapter 6.

1.3 Research Contributions

As displayed in Figure 1.1, we develop new approaches to tackle two fundamental tasks — point cloud semantic segmentation and 3D keypoint extraction for point cloud registration in the field of 3D scene understanding. Each chapter in this thesis has the following major contributions:

- In Chapter 3, we introduce a novel Long-Short-Term Context framework (LSTC), which adopts a long-short-term feature bank to exploit both the local context within each block and the long-range context beyond the current task block. The proposed framework is flexible and easy to be combined with existing models, thereby enabling existing models to capture spatial context within an arbitrary range. Extensive experiments demonstrate that the proposed model

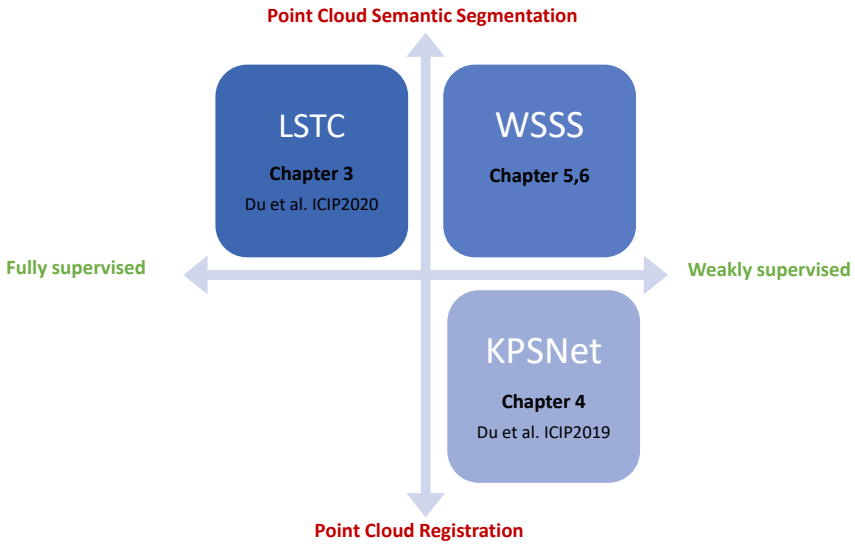


Figure 1.1 : **Thesis contributions**: supervised and weakly-supervised methods for 3D scene understanding.

achieves improved segmentation performance, and augmenting existing models with a long-short-term feature bank consistently increases the performance. This chapter is supported by one published conference paper in "*2020 IEEE International Conference on Image Processing (ICIP)*".

- In Chapter 4, we propose a KeyPoint Siamese Network (KPSNet) to learn the task-specific keypoint detector simultaneously and feature extractor for point cloud registration. Since the proposed approach does not require manually annotating keypoints and local patches pairwise, it can be easily extended to new scenes. Experiments on the open-source benchmark dataset show the validity and competitive performance of our approach. This chapter is supported by one published conference paper in "*2019 IEEE International Conference on Image Processing (ICIP)*".

- In Chapter 5, we present a weakly-supervised point cloud semantic segmentation approach using coarse labels. We reformulate the weakly-supervised

point cloud semantic segmentation with coarse labels as a problem of semi-supervised point cloud semantic segmentation with noisy pseudo labels, and devise a three-branch framework to learn a semantic segmentation model from the noisy pseudo labels. The proposed framework can reduce the impact of noises in pseudo labels and leverage the inner structure of point clouds to improve segmentation performance. On benchmark datasets, our weakly-supervised method for point cloud semantic segmentation outperforms other weakly-supervised methods and fully-supervised method PointNet++, and narrows the gap with other outstanding fully-supervised approaches. This chapter is based on one journal paper submitted to "*IEEE Transactions on Multimedia*".

- In Chapter 6, we present PCL, a label-efficient point cloud semantic segmentation method using incomplete point-level labels. We explore cross-sample contrast to learn category-discriminative representations and develop a dynamical label refinement strategy to mine supervision information from pseudo labels while alleviating the impact of noisy pseudo labels. Experiments on benchmark datasets show the proposed approach gains outstanding performance on benchmark datasets with several weakly-supervised settings. This chapter is based on one journal article that is being polished and submitted to "*IEEE Transactions on Image Processing*".

1.4 Structure of Thesis

This thesis is organized as follows:

- *Chapter 2*: this chapter provides a complete overview of the literature on deep learning on point clouds, point cloud semantic segmentation, 3D keypoint detection, weakly-supervised learning on point clouds, and contrastive learning for dense prediction.

- *Chapter 3*: this chapter introduces our LSTC approach, a supervised point cloud semantic segmentation framework, which exploits long-range spatial context with the help of the proposed long-short-term feature bank, therefore improving the segmentation performance.

- *Chapter 4*: this chapter introduces our KPSNet, a weakly-supervised 3D keypoint extraction method for point cloud registration, which does not require manually annotated 3D keypoint labels.

- *Chapter 5*: this chapter presents our weakly-supervised framework for point cloud semantic segmentation, employing only coarse weak labels as supervision.

- *Chapter 6*: this chapter presents our PCL, a label efficient point cloud semantic segmentation approach that employs incomplete labels as supervision, and highly improves the performance of weakly-supervised point cloud semantic segmentation via cross-sample contrast and dynamical label refinement.

- *Chapter 7*: the last chapter concludes the entire thesis and provides insights on future directions.

Chapter 2

Literature Review

In this chapter, we briefly review works related to 3D point cloud scene understanding. In particular, we start from deep learning technology on point cloud data in Section 2.1, followed by existing efforts on point cloud semantic segmentation in Section 2.2 and 3D keypoint detection in Section 2.3. Since the methods presented in Chapters 4, 5 and 6 are weakly-supervised, we review weakly-supervised approaches applied on 3D point clouds in Section 2.4. Contrastive learning is a popular approach for representation learning in 2D and 3D vision tasks, which is also applied to our works about point cloud semantic segmentation. Therefore, at last Section 2.5, we introduce related works of contrastive learning for dense prediction.

2.1 Deep Learning on Point Clouds

There are many efforts to explore deep learning techniques on point cloud data. According to the types of point cloud representations fed into the deep neural networks, these methods are categorized into four groups: multi-view based, voxel based, raw points based, and other methods.

2.1.1 Multi-view based Methods

Multi-view based methods firstly project, from multiple virtual views, point clouds into a set of 2D images according to the projection strategies. Then they use the well-designed 2D Convolutional Neural Networks (CNNs) to extract features from each projected image. Finally, these view-wise features are fused to various 3D understanding tasks, including 3D shape classification [87, 73], segmentation [51]

and detection[12, 71, 65].

The basic idea of projection strategies is to choose the positions for virtual cameras to take pictures of the point clouds. [54] chooses to rotate a virtual camera around a fixed vertical axis and generates four virtual views every 12° by changing the pitch angles and translations, producing three images (*i.e.* color, depth and surface normal) at each virtual view. [6] proposes to take photos of the re-produced 3D mesh of a given point cloud, instead of directly rending images from the point clouds, in order to prevent points behind the observed surface from revealing due to the sparsity problem of point clouds. It uses random virtual views to generates a set of images by randomly choosing the center coordinates of the virtual camera within the given 3D scene space, as well as the altitudes in $10 \sim 30$ meters and the view directions in a 45° cone facing the ground. The projection strategies are very important for multi-view based methods, because their performance is sensitive to viewpoint selection. However, it is difficult to choose appropriate and enough viewpoints to gain descriptive feature representations of point clouds.

How to aggregate a series of view-wise features to form a discriminative fused representation of the given 3D scene is another crucial problem. [87] generates the global feature representations by max-pooling multiple view-wise features. As a result of the max-pooling operation, much information is lost because it only makes use of the maximum elements of the multi-view features. [122] proposes to leverage the inter-relationships between multi-view images to generate a discriminative global representation. It designs a relation network, which uses a reinforcing block to explore the region-to-region relationship between different view-wise feature maps to enhance the features of each individual image and use an integrating block to exploit the view-to-view relationships across a set of view-wise features to fuse them to a global representation for 3D object recognition.

Besides, information loss is unavoidable when projecting a point cloud. Therefore, this line of methods cannot fully exploit the geometric and structural information of given point clouds. With well-engineered 2D CNNs and well-designed projection strategies, multi-view based approaches can achieve leading performance on the task of shape classification. However, using this type of representation always fails when referring to other tasks of 3D scene understanding, like point cloud semantic segmentation and shape completion.

2.1.2 Voxel based Methods

This series of approaches firstly generate regular 3D voxel grids by voxelizing given unstructured point clouds, and then leverage 3D CNNs for 3D computer vision tasks. [73, 113, 66] are the pioneers that adopt 3D CNNs to the volumetric representations of point clouds for 3D object recognition. There are also a lot of works using 3D CNNs on voxelized point cloud data for semantic segmentation [17, 92]. These approaches have yielded encouraging outcomes, but they are still hampered by several obvious limitations. Firstly, the memory capacity and computational complexity of 3D convolution grow by a third power with respect to the resolution of point cloud data, which impedes these approaches from being applied to large-scale dense point clouds. Secondly, this issue is further exacerbated by the data sparsity of point clouds, because point cloud data only records information on the surface of objects. The type of voxelized representation of point cloud naturally adds a lot of redundancy, and almost 90 percent of the voxelized representation is empty [131]. Thirdly, this line of methods are sensitive to the voxel size due to information loss caused by voxelization, which also results in the problem of balancing between accuracy and efficiency.

To reduce the information loss during voxelization and preserve the details within voxels, [55] proposes PointGrid to combine the point and voxel representation by

representing the feature of a voxel as the fused feature of a fixed number points within the voxel. To reduce the memory and computational costs, [75, 103] propose to leverage octree structure to focus memory allocation and computation on relevant dense voxels. Specifically, [75] proposes to partition point clouds into 3D voxels in a hierarchical fashion based on the octree structure, until the voxels either hold a single active site or are empty. The feature representation stored in each leaf node is obtained by pooling all feature activations of the voxel it comprises. By operating on the octrees, [75] can focus computational and memory resources depending on the 3D geometric structure of input data, leading to efficient sparse convolutional operation. However, empty space still suffers from computational and memory burdens. Recently, [15, 28] develop sparse convolution that does not perform computations on empty voxels. With the developed sparse convolution, they build large-scale 3D networks on voxelized point clouds and obtain advanced performance in many tasks of 3D scene understanding, including classification, detection and segmentation.

2.1.3 Raw Point based Methods

In place of converting point clouds to images, voxels or other representations, raw point based approaches apply deep learning to the raw points. [72] proposes a pioneering method, which is called PointNet, that directly presents 3D point clouds. Its main idea is to learn a convolutional neural network to embed every single point of the given point cloud to a feature vector and then combine all individual point feature vectors to a fused feature representation. Then [72] utilizes the fused feature representation for object classification tasks and part/semantic segmentation tasks within a unified neural network framework. PointNet is quite simple yet highly efficient and effective. However, PointNet only captures global features of point clouds and does not exploit the useful local structure. It is proved that local structure occupies a crucial position in the success of convolutional neural networks. To

exploit local structure, [74] introduces a hierarchical architecture, PoineNet++, from neighboring regions progressively. To be specific, [74] proposes the set abstract layer that divides an input point set into a number of overlapping local regions depending on a distance metric. Then the features of points within each local region are extracted and aggregated to form a local feature of the local region, feeding to the higher-level as a unit feature. Repeating the set abstract layer several times, the network generates the global representation of the input point cloud that contains fine geometric structures from local neighborhoods. Since PointNet and PointNet++, more efforts have been made to operate on raw point clouds by developing new kinds of deep learning networks. DGCNN [108] aims to capture local structure by dynamically searching neighbors of points in feature spaces and aggregating their features to enhance the point features. SPG [53] embeds PointNet in edge graph CNNs for large-scale semantic segmentation. It employs a PointNet-like network to produce local node features of a constructed superpoint graph and then apply edge graph convolution on graphs for point cloud semantic segmentation. VoxelNet [131, 60] combines PointNet with 3D CNNs for 3D object detection. [36, 57, 59, 50] take regular grids as convolutional kernels. When operating on point clouds, as the neighbors of a point scatter in various locations locally, they modify the kernel weights for neighboring points according to their offsets to the point. [62, 112, 129, 95, 120, 46] consider the convolutional operation on point clouds as convolution on continuous signals and apply continuous functions to create kernel weights for points' neighbors. By considering different factors that affect the weights operating on neighboring points, various kernel functions are designed, resulting in a number of novel deep networks for point clouds.

As raw point based approaches operate directly on point clouds, they can avoid information loss during format transformation. In the past five years, this line of methods has attracted much attention and achieved outstanding progress in 3D

scene understanding based on point clouds. However, most of this line of approaches only receive small-scale point clouds, resulting in inconvenience to capture long-range spatial context and inefficiency when dealing with large-scale point clouds in tasks of 3D scene understanding.

2.1.4 Other Methods

Several researchers also explore other representations for point cloud understanding.

Spherical Representation: [110] proposes SqueezeNet that firstly projects a 3D point cloud by spherical projection, forming a 2D LiDAR image. It treats projected LiDAR images as RGB images to be processed with standard convolutional networks. As this spherical representation does not require viewpoint selection and view-wise feature aggregation, as well as leveraging many powerful 2D CNNs, it attracts much interest quickly. [111, 117, 68] follow this spherical representation to further improve the performance. [111] leverages an unsupervised domain adaption pipeline to address domain shift. [117] proposes Spatially-Adaptive Convolution (SAC) for efficient computation. [68] proposes RangeNet++ to address the problems of discretization errors and blurry inference outputs by an efficient post-processing step. However, similar to multi-view based methods, information loss is also unavoidable for spherical projection based methods because of the point occlusion problem during projection process. *Graph based Methods:* Recently, some researchers have attempted to extend deep learning techniques to unstructured graph data, deriving the graph convolution networks. Since 3D point clouds lie in the irregular and non-Euclidean domain, they can be naturally structured as graphs and then be processed by graph operations. [86] is a pioneer that employs graph convolutions to address the point cloud classification task. The authors proposed a convolution operation on graph data constructed based on spatial distances. The filter weights

are defined to condition on edge labels and dynamically generated, therefore called Edge-Conditioned Convolution (ECC). In the following work of [53], a superpoint graph based architecture is introduced to process large-scale point clouds, capturing spatial context to improve segmentation performance. This architecture uses PointNet for superpoint embedding and introduces a more efficient version of ECC [86]. [93] proposed RGCNN for point cloud segmentation. Each layer of RGCNN is regularized by adding graph-signal smoothness prior to the loss function. Indeed, these methods recast point clouds into graphs to mine the structure information of point clouds and leverage graph convolution networks to capture the interactions between points to enhance feature representations.

2.2 Point Cloud Semantic Segmentation

Deep CNNs have already been broadly applied in 2D image understanding tasks. Several outstanding deep architectures, like ResNet [33], appear as a general feature extractor network (called backbone). But in the area of 3D scene understanding based on point cloud data, such deep architectures still do not exist due to the challenge of processing unordered and unstructured (*i.e.* irregular or non-raster) point clouds. Based on the deep learning techniques introduced in Section 2.1, various deep architectures have been introduced to 3D semantic segmentation. Similarly, depending on the formats of the input data of deep architectures, we group these architectures into multi-view based architectures, voxel based architectures, point based architectures and other architectures.

2.2.1 Multi-view based Deep Architectures

Multi-view based deep architectures [54, 5] convert an irregular point cloud to multiple images, and then usually apply 2D CNNs that are developed from 2D image semantic segmentation models, such as ResNet [33], Deeplab [9], FCN[64],

and U-Net [76] to predict per-pixel score maps for all projected images. The final segmentation result of an input point cloud is obtained by aggregating all predicted score maps over different views and reprojecting to the point cloud. [54] generates several views for an input point cloud, each view consisting of 2D color, depth and surface normal images and processed by a three-stream architecture. All predicted score maps over different views are fused and re-projected via the score fusion module to generate the final prediction. To deal with point clouds with low density, [5] firstly generates a mesh of a point cloud, and then generates two types (*i.e.* RGB and depth composite) of multi-view images from the mesh. Finally, the predicted segmentation results of all views are first reprojected to mesh, and then to the original point cloud. Instead of predicting score maps directly from multiple views, [43] proposes Multi-View PointNet (MVPNet) that uses multi-view based 2D U-type networks to generate 2D feature maps. After that, the authors reversely project and aggregate the 2D feature maps to the original point cloud to augment the point-wise features. Generally, the performance of multi-view based approaches is easily influenced by the projection strategies and aggregation strategies. Under the presumption that data points in small areas are distributed nearly on Euclidean surfaces, [91] proposes tangent convolution and builds a U-type network on the tangent plane around every point of a point cloud, called tangent images. Obviously, [91] can produce per-point feature representations without reprojection operations by taking the feature representation of each tangent image as that of each point. As a result of precomputing the tangent images based on the local surface geometric information, [91] can process large-scale datasets for efficient point cloud semantic segmentation.

2.2.2 Voxel based Deep Architecture

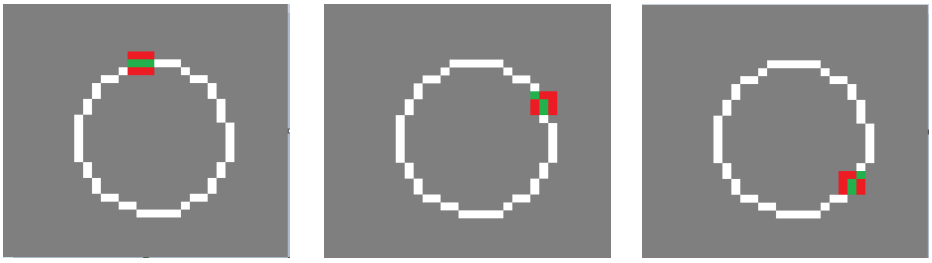
Voxel based deep architectures [70, 131, 15, 28] usually take the regular 3D voxels, transformed from the original point clouds, as input and implement 3D convolutional neural networks. [38, 17] convert point clouds to sets of occupancy voxel-grids. After that, they utilize a 3D Fully Convolutional Neural Network (3D FCNN) to produce segmentation prediction of each voxel. The final point cloud segmentation results are obtained by assigning the predicted labels of a voxel into all points within the voxel. Obviously, the performance of these approaches is subject to the boundary artifacts caused by the voxelization and the voxel size. If reducing the voxel size (*i.e.* increasing resolution) for better performance, it will be difficult to train the segmentation networks due to the huge memory and computational cost. Many efforts have been done to solve the above problems. To refine the final point cloud segmentation results, [92] proposes SEGCloud that firstly maps the poor voxel-wise predictions produces by 3D FCNN to the original point cloud by the deterministic trilinear interpolation operation, and then applies Fully Connected Condition Random Field (FC-CRF) to achieve global consistent segmentation results on the basis of the assumption of spatial consistency in the per-point labels. To capture local details within voxels and reduce the information loss caused by coarse voxels, [67] proposes the Voxel VAE network (VV-Net) that embeds the local details within each voxel by a kernel-based interpolated variational autoencoder (VAE) architecture. Therefore, each voxel is represented by an information-rich feature vector defined by the latent space representation using a pre-trained VAE, replacing the traditional voxel representation of a 0-1 occupancy signal. Moreover, to achieve robust segmentation results, instead of using a standard convolution operation, [67] defines a group convolution operator to extend group equivariant CNN to 3D. Equipped with the group convolution, the segmentation network can be invariant to transformations of translation, rotation and symmetry, and enhance the expressive capacity without in-

creasing parameters. However, all of the above voxel based approaches consider the voxelized data as densely populated grids, therefore they usually suffer from the limitations of memory capacity and computational complexity growing exponentially with increasing resolution.

As point clouds are sparse in 3D space and generally distributed on surfaces of objects, exploiting the data sparsity is crucial for efficient voxel based semantic segmentation approaches. [27] implements a sparse convolutional operator for 3D data that only stores and calculates the *active* sites of each input of each layer to reduce memory requirements and speed up operations. Specifically, for a spatial location (site) in the input layer, [27] defines it as active if the site's vector is not the zero vector; for a site in a hidden layer, [27] defines it as active if any of its input sites is active. In other words, a site in a hidden layer takes in some sites from its lower layer. If any of these sites is active, then the site in the higher layer is set as active. However, the degree of data sparsity will rapidly disappear with convolutional layers increasing, called submanifold dilation [28]. For intuitive understanding, we show an example of submanifold dilation in Figure 2.1(a). To maintain data sparsity across the entire network and use the sparse convolutional operators, [23] uses rectified linear units (ReLU) to keep data sparsity in the intermediate feature representations, and introduces an $\mathcal{L}_1$ regularizer to encourage the network to discard uninformative feature representations and increase sparsity. However, the submanifold dilation of sparse data caused by the convolutional operator is still a bottleneck of the sparse convolutional networks with respect to the memory and computational cost.

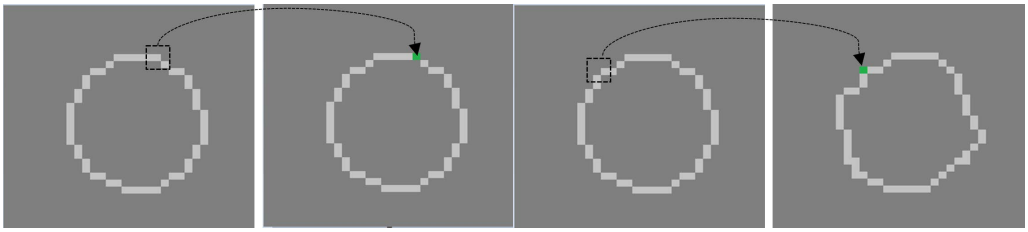
Very recently, [28] proposes the submanifold sparse convolution (SSC) that processes spatially-sparse data more efficiently. Different from prior convolutional operators, SSC keeps the pattern of active sites, as shown in Figure 2.1(b). Taking SSC as the basic block and together with other derived operators such as strided sparse convolution and batch normalization, [28] develops submanifold sparse convo-

(a) Sparse Convolution (SC)



(b) Submanifold Sparse Convolution (SSC)

Figure 2.1 : Illustrations of sparse convolution (SC) and submanifold sparse convolution (SSC) [28]. Dark grey color denotes *non-active*. (a) shows an example of SC used in [27]. An input sparse data (left) passes through a convolutional layer with a 3×3 kernel filled with $1/9$, obtaining a more dense result (middle) with more active sites. Feeding the middle result to the same convolutional layer further increases the number of active sites, obtaining the right result. This phenomenon is called submanifold dilation. (b) illustrates the calculating procedure of SSC proposed by [28]. A output site is determined to be active if and only if the central site of its receptive field in the input is active. Therefore, the output are forced to have the same number of active sites as the input. When applying a SSC on a sparse input, if an output site is active, SSC will compute its output feature vector; otherwise, no computation performs. From left to right: receptive fields of a 3×3 SSC centered at different active sites. Green sites denote active sites in the receptive field, and red sites are non-active sites that are ignored by SSC.



(a) Submanifold Sparse Convolution (b) Generalized Sparse Convolution

Figure 2.2 : Difference between submanifold sparse convolution [28] and generalized sparse convolution [15]. White sites denote active sites in the input. Dark grey color denotes *non-active*. Light grey sites denote active sites in the output. Dashed lines mark the receptive field of a 3×3 convolutional kernel and grey sites denotes their output sites. (a) Submanifold sparse convolution does not change the pattern of active sites. (b) Generalized sparse convolution maintains the degree of data sparsity, but does not enforce the pattern of active sites to be unchanged.

lutional networks (SSCNs) for efficient point cloud semantic segmentation, including sparse variants of the popular FCNs, U-Nets and ResNets. [15] further improves the SSC operation and proposes the generalized sparse convolution that maintains the degree of sparsity of active sites but does not restrict the pattern of active sites in the output, as shown in Figure 2.2. Moreover, [15] develops an open-source library, Minkowski, to implement the generalized sparse convolution, called MinkConv. Using Minkowski, [15] builds large-scale U-type Minkowski convolutional neural networks, MinkUNets. The authors apply MinkUNets to the point cloud segmentation task, and gain top performance on various 3D benchmarks. More importantly, SSCNs and MinkUNets can deal with a complete large-scale point cloud at a high resolution instead of sub-clouds with the help of the proposed SSC and MinkConv, therefore capturing long-range spatial context conveniently. Subsequently, SSCNs and MinkUNets are widely used as backbone for point cloud semantic segmenta-

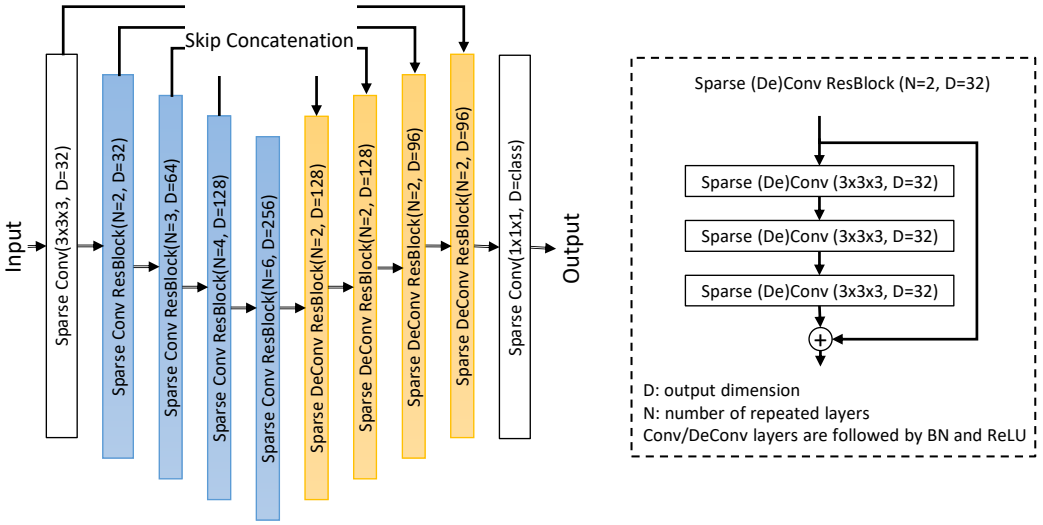
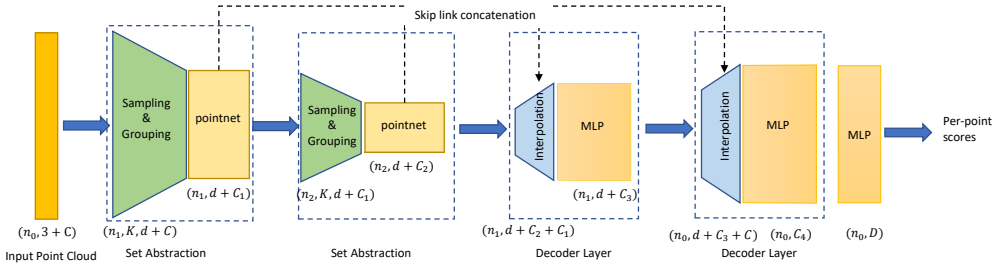


Figure 2.3 : Architecture of a 3D-UNet implemented with Minkowski. The architecture takes voxels as input. It firstly embeds the input into higher dimensional ($D = 32$) representations using a sparse convolutional layer with kernel size of $3 \times 3 \times 3$, denoted as SparseConv($3 \times 3 \times 3, D = 32$), and then uses the encoder and decoder to generate a 96-dimensional feature representation for each active voxel. The encoder is consist of four sparse residual blocks (SparseConv ResBlock), while the decoder is consist of four sparse residual deconvolutional blocks (SparseDeconv ResBlock). The intermediate representations of each block of the encoder are combined with the corresponding ones of the decoder for feature enhancement. Finally, a classifier (SparseConv($1 \times 1 \times 1, D = class$)) receives the final 96-dimensional feature representations as input and produces the segmentation predictions.

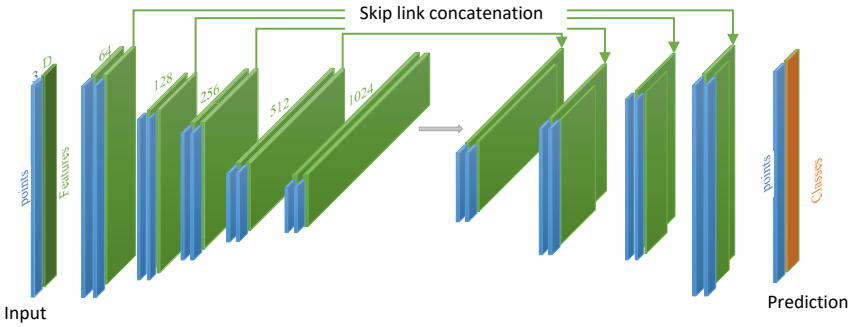
tion [77], point cloud instance segmentation [47], self-supervised 3D representation learning [115, 34] and weakly-supervised point cloud segmentation [63, 45, 89]. Considering the high efficiency, performance and applicability of MinkUNets, we adopt the 3D-UNet architecture [115] implemented with Minkowski as one of the backbone architectures in this thesis. Figure 2.3 displays the detailed architecture of the backbone network.

2.2.3 Point based Deep Architecture

Point-based point cloud semantic segmentation methods [72, 74, 108, 22, 129, 112, 8, 95, 35] directly applies deep architectures on raw point clouds to learn point-wise features. As a pioneer, PointNet [72] embeds each point individually through shared multi-layer perceptrons (MLPs), generating per-point feature. Every per-point feature is augmented by concatenating the aggregated global feature. Finally, the augmented point features are used to predict the point cloud segmentation results. As PointNet cannot capture local structure, the following works develop many techniques to explore the local structure. As shown in Figure 2.4(a), PointNet++ [74] proposes a symmetrical encoder-decoder architecture to capture local context, which adopts a hierarchical structure composed by several set abstraction layers as encoder and another hierarchical structure composed by several propagation layers as decoder. Specifically, the set abstraction layer uses max-pooling layers to aggregate point features within each local region. DGCNN [108] proposes the EdgeConv operation to captures local geometric structure and integrates EdgeConv into PointNet to form a segmentation network. PointCNN [59] proposes $\mathcal{X}$-Conv layer to explore spatially-local correlation. The proposed $\mathcal{X}$-Conv operator works as follows. It firstly learns an $\mathcal{X}$-transformation conditioned on its input points. Then it performs the learned $\mathcal{X}$-transformation on its input points, associated with their features, in order to permute input points to a specific order. Subsequently, a typical convolutional operation is carried out on the permuted points to produce new embeddings. PointWeb [129] attempts to exhaust the contextual information in a local region by considering the relationship of all pairs of points in the local region. PointConv [112] defines the convolution operation on point clouds as a continuous convolution and designs continuous functions to generate the convolutional kernels. As the kernel functions take the densities of points' neighborhoods as parameters, PointConv is robust to point clouds with a variety of densities. To build a segmen-



(a) Architecture of PointNet++ [74] for point cloud segmentation



(b) Architecture of KPConv [95] for point cloud segmentation

Figure 2.4 : Illustrations of popular point based deep architectures.

tation network in an encoder-decoder style, the authors also present a deconvolution version. [95, 120] propose to define explicit convolution kernels for points and then build networks based on their developed convolution kernels. Specifically, [95] proposes Kernel Point Convolution (KPConv) that is efficient and robust to varying densities. [95] also proposes a deformable version of KPConv, which can learn to modify kernel points according to local geometry. With the proposed KPConv and its deformable version, the authors present two network architectures to address point cloud classification and segmentation, respectively. Figure 2.4(b) shows the architecture of KPConv for point cloud segmentation, KP-FCNN. The segmentation network KP-FCNN is an encoder-decoder-style architecture. Its encoder contains a five-layers hierarchical architecture. The points are downsampled progressively after each layer by the derived strided KPConv. Its decoder is composed of several nearest

upsampling layers correspondingly, each together with a MLP, to progressively get the final point-wise feature representation. The intermediate representations from the encoder are concatenated to the corresponding upsampled ones from the decoder through Skip links. Among the above-mentioned point based architectures for point cloud segmentation, KP-FCNN [95] achieves the top performance. Moreover, KP-FCNN can process relatively large point clouds with varying densities efficiently. Therefore, we use KP-FCNN as one of our backbone in this thesis.

Since the above-mentioned point-based methods usually take a subdivided sub-cloud as input due to the high computational memory requirement, they cannot exploit long-range spatial contextual information. To deal with large-scale point cloud semantic segmentation, [53] proposes to apply an edge convolutional network on the introduced superpoint graphs to model long-range context, whose nodes are set of geometrically simple points called superpoints. However, generating superpoints requires extra pre-processing, and the quality of superpoints has a significant impact on its performance. RandLA-Net [35] is based on KPConv, but has the capability of processing large-scale point clouds. However, RandLA-Net still does not consider capturing long-range spatial context. Therefore, developing a framework, which enables existing point based semantic segmentation approaches to capture long-range spatial context, is one of our objectives in this thesis.

2.2.4 Other Architecture

Spherical projection based deep architectures. Several works projects a point cloud onto a spherical surface to generate a 2D range image, and apply 2D CNNs to generate pixel-wise range image representations. In the end, these approaches map the representations of the range images to the corresponding original point clouds, thereby producing the point-wise segmentation results of the given point clouds. Compared with multi-view based methods, spherical projection based methods sig-

nificantly reduce the complexities of projection based deep architectures because they do not need to consider how to select the viewpoints to generate multi-view images and how to aggregate the predictions across all views. SqueezeSeg [110] proposes an end-to-end pipeline for real-time road-object segmentation from point clouds. It takes range images transformed from point clouds as input and generates the segmentation results based on SqueezeNet [42] and CRF. The following work SqueezeSegV2 [111] improves the structure of SqueezeSeg by introducing a Context Aggregation Module (CAM) to increase robustness to noise, as well as improvement in loss function. SqueezeSegV3 [117] proposes the Spatially-Adaptive Convolution (SAC), an efficient spatially-adaptive convolution, to adopt different filters for different image locations. RangeNet++ [68] proposes a GPU-accelerated post-processing post-processing algorithm to mitigate problems caused by the spherical projection, such as discretization errors and blurry CNN outputs. Considering the real-time requirements in applications like autonomous driving, PointSeg [107] proposes a lightweight network based on SqueezeNet, which introduces an enlargement layer and an squeezing reweighing layer to make a balance on efficiency and accuracy.

Graph based architectures. A few researchers also exploit graphs to represent point clouds, and develops graph based architectures for point cloud semantic segmentation [53, 102, 108]. As they use a graph to describe a point cloud, graph based architectures seek to generate the feature representation of each node and predict its class. The construction of graphs and the graph convolutional operation are two of the most important components of graph based architectures. [53] firstly over-segments a point cloud, generating a set of simple segments (called superpoint) based on geometric features. Then it builds a directed graph (SPG) for each point cloud, where each node denotes a superpoint, and its feature vector is represented by the output of a tiny point based network operating on the superpoint. The edges of SPG denote the adjacency relationship between superpoints and whether two su-

perpoints are adjacent or not depends on the distance and similarity of point pairs respectively from the two superpoints. The input edge features are represented by a 13-dimensional vector describing the geometric attributes of the adjacent superpoints. As for the graph convolution, [53] combines GRU (Gated Recurrent Unit) and ECC [86] based on the assumption that the feature representation of a superpoint depends on itself and its local neighbors within the SPG, in order to capture contextual information. [102] builds the graph of a point cloud by taking each point as a node and adding edges between neighboring nodes depending on the spatial distances.

Transformer based architectures. Transformer based architectures, simply called transformers, have achieved great success in the image domain. Very recently, transformers in 3D vision has been attracting more and more interest. Many transformers are proposed to deal with object part segmentation [116, 128, 61, 37], which can only process very small scale point clouds one time due to the limitations of computational and memory resource. To apply transformers on point clouds semantic segmentation in large scale complete scenes, [69] proposes to apply voxel hashing proposed in [15] for fast neighborhood selection and to speed up local self-attention modules. [96] proposes to generate segment-wise features from graph segmentation methods and apples a stack of attention encoder blocks to fuse these segment-wise features. [118] proposes a cross-scale attention module to aggregate the generated sparse voxel features to capture the long-range context for large objects.

The methods mentioned above are fully-supervised that require vast amount of annotated data with dense labels (point-level or voxel-level) for training. Obviously, designing such fully annotated datasets is labor-intensive and time-consuming in terms of data cleansing and manual labeling.

2.3 3D Keypoint Detection

3D keypoint detection pursues a repeatable and distinctive 3D local representation from 3D data, e.g., point cloud and 3D mesh, which can be used to establish correspondences between 3D surfaces. A 3D keypoint is represented by its position coordinates and feature representation. It plays a key role in pairwise 3D registration, 3D shape retrieval and 3D objection recognition and pose estimation. A completed 3D keypoint detection process generally contains a 3D keypoint detector and a 3D keypoint descriptor.

2.3.1 3D Keypoint Detector

The objective of 3D keypoint detectors is to identify the positions of 3D keypoints. This is a challenging task and only a few works investigate 3D keypoint detectors. The main reason is that it is difficult to define how a 3D keypoint should be like. Therefore, the common way is to use a hand-crafted saliency function by combining the domain knowledge, or uniformly select some local patches and taking their centers as keypoint [79, 78, 48, 125]. However, this type of 3D keypoint detectors may not yield the best performance when combined with the given 3D descriptors for downstream tasks. Recently, [98] proposes a descriptor-specific keypoint detector, which casts 3D keypoint detection as a classification problem in terms of whether the points can be matches by a pre-defined 3D descriptor. The performance of this method depends on the pre-defined 3D descriptor. However, learning a good 3D descriptor needs a large amount of annotated keypoints, and thus causes a chicken-egg problem. Very recently, a few works [88, 44, 21, 3] attempt to extract task-specific 3D keypoints via an end-to-end optimization to improve the overall performance in a particular task. [88] learns an ordered set of consistent 3D keypoints across multiple views and object instances for a specific downstream task, such as relative pose estimation. However, it requires to train one model for each

category, and is inflexible to be extended to new scenario. 3DFeat-Net [44] learns both 3D keypoint detector and descriptor for point cloud registration via a three-branch Siamese architecture, which uses a training tuple of an anchor, positive and negative point cloud to compute a triplet loss. Its 3D keypoint detector outputs the orientation and attention predictions, and learns to give higher attentions to keypoints that are useful for point cloud registration implicitly. In this thesis, we also study to simultaneously learn 3D keypoint detector and descriptor in a unified framework [21]. Different with [44], our framework is a two-branch Siamese architecture and learn the 3D keypoint detector explicitly via a classification task to identify whether a keypoint candidate is useful for point cloud registration. Subsequently, D3Feat [3] leverages a fully convolutional network for joint dense feature detection and description, where a detector loss is proposed to guide the network to predict highly repeatable keypoints.

2.3.2 3D Keypoint Descriptor

The objective of 3D keypoint descriptor is to learn the representations of 3D keypoints or 3D local patch. Most existing works related to 3D keypoint focus on 3D descriptors. Early works [48, 78, 97] are unsupervised and use **handcrafted 3D features** (*e.g.* histograms) to estimate the similarity of keypoints by calculating the point number in each spatial bin or considering the surface normal property. These approaches are developed for specific applications. Therefore, it's usually hard to extend them to new scenes. [125] is a pioneer that applies deep learning to **learn 3D descriptors**. It uses a Siamese 3D CNN that takes voxel grids as input to match 3D local patches and has made significant progress. Its major principle is to train the network to ensure that the features of matched pairs are similar and the features of un-matched pairs are diverse. Therefore, [125] requires the dense annotated data, containing a number of matched 3D local patches for

training. Subsequently, many learning-based approaches follow [125] to take uniformly distributed local patches as input and design various networks to learn 3D descriptors [20, 19, 26, 1]. For example, PPFNet [20] leverages PointNet [72] as its encoder and takes raw points augmented with point-pair features (PPF) as inputs. It learns a globally informed 3D descriptor by a N-tuple loss based on the permutation invariant property. PPF-FoldNet [19] leverages PPF and FoldingNet [121], which is an auto-encoder architecture, to develop an unsupervised method to learn rotation-invariant 3D descriptors. SpinNet [1] transforms 3D local patches into 3D cylindrical volumes and leverages 3D cylindrical convolution to learn features. It uses the local reference frame for patch alignment to learn rotation-invariant 3D local descriptors, and achieves advanced generalization ability to unseen datasets. However, such patch-based methods usually suffer from low computational efficiency because of the repeated calculations on the overlapping regions of adjacent patches, which limits their usage in many real-world applications. To deal with this limitation, [16] makes the first attempt to apply a fully convolutional architecture on the whole point cloud in a single pass to learn dense 3D descriptors. It uses 3D sparse convolution proposed in [15] and achieves comparable performance to patch-based methods with 600× speed-up in computation. Subsequently, the encoder-decoder architecture are widely used to learn dense 3D descriptors via end-to-end optimization for point cloud registration [84, 124].

2.4 Weakly Supervised Learning for Point Cloud Understanding

Various approaches are proposed to reduce the demand for huge amounts of annotated 3D data. We broadly divide these techniques into two groups according to the types of their supervision information: incomplete supervision and inexact supervision. Figure 2.5 illustrates an example of point cloud semantic segmentation

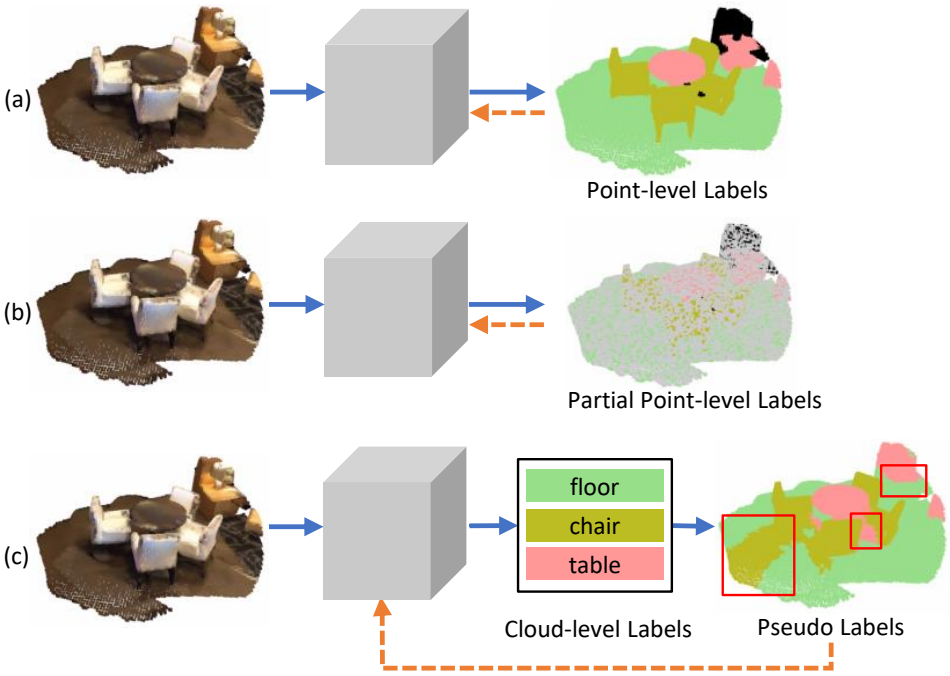


Figure 2.5 : Illustrations of differences between training semantic segmentation models under (a) full supervision with point-level labels, (b) incomplete supervision with partial point-level labels (grey color marks unlabeled points), and (c) inexact supervision with cloud-level labels.

trained with different types of annotations for reference.

2.4.1 Incomplete Supervision

Along this line of works, most approaches use self-supervised learning to pre-train a feature embedding network. After that, they employ a small portion of annotations to train a classifier or fine-tune a semantic segmentation network initialized with the pre-trained model [58, 32, 80, 10, 81, 19, 115, 34]. These approaches focus on learning general feature representations, which benefit to downstream tasks. Although achieving impressive performance, but there are still limitations due to the gap of optimized objective between the downstream task and the self-supervised pretext task. Most recently, [119, 127, 63] develop methods specifically for point

cloud semantic segmentation with partial annotations, e.g., 10%, 1% and 1pt (one labeled point per category in each training sample) of points are labeled. [119] is a pioneer that designs a weakly-supervised point cloud segmentation network using partial point labels. In order to learn better feature representations, it introduces four losses: a standard segmentation loss constraining labeled points only, a multiple-classification loss constraining the global features of an input, a Siamese loss constraining all points under the consistency assumption of the predictions of a point cloud and its transformed variant, and a smooth loss constraining all points based on the locally smooth assumption. [119] achieves impressive results with 10× fewer labeled points. However, the proposed network takes small point cloud blocks, containing several thousand points, as input, and can not deal with large-scale point clouds, because the smooth loss requires huge computational resources. [127] designs a self-supervised pretext task for pre-training, and then fine-tunes the segmentation network with the encoder initialized by pre-trained weights. The authors also present a label propagation method to expand the given sparse labels. [127] evaluates the proposed approach with different annotation settings of {1pt, 0.02%, 1%, 10%}, and shows outstanding results in the task of weakly-supervised point cloud semantic segmentation. [63] proposes a new weakly-supervised annotation setting of 1t1c, which means annotating one point for each object in a point cloud. With this setting, [63] propagates the sparse labels based on pre-computed over-segments and trains the segmentation network in a self-training manner by updating the labels iteratively. [63] obtains outstanding performance on indoor datasets, which also presents an inspiration to design an excellent annotating strategy within a fixed budget. [45] focuses on a different weakly-supervised setting of providing a fraction of fully annotated point clouds and evaluates the proposed approach on several datasets with {5%, 10%, 20%, 30%, 40%} of the training point clouds fully annotated. It applies semi-supervised learning following the consistency regularization strategy [90]

to constrain unlabeled data, which promotes the features of matched point pairs to be close in the feature space after using separate transformations.

2.4.2 Inexact Supervision

Instead of point-level annotations, [109] proposes MPRM that only uses cloud/subcloud-level annotations for point cloud semantic segmentation. [109], a pioneer that uses cloud-level weak labels for point cloud semantic segmentation, follows the standard pipeline of weakly-supervised semantic segmentation (WSSS) in 2D computer vision. Specifically, given the cloud-level weak labels, [109] firstly trains a classification network, and uses it to generate the point-level pseudo labels. After then, with the produced pseudo labels, it re-trains a standard point cloud semantic segmentation network, same as supervised approaches. Intuitively, the quality of pseudo labels act as a critical factor for the ultimate performance of the method. Therefore, [109] focused on improving the classification network to generate better pseudo labels and designed the multi-path region mining module. However, it does not take into account how to make better use of the noisy pseudo labels. However, the inevitable noises (*i.e.* incorrect labels) in the generated point-level pseudo labels will have impact on the learning of the semantic segmentation network in the retraining stage. Designing a semantic segmentation network with cloud-level annotations can reduce the annotation cost greatly, having promising commercial value and wide applications. However, this research field has not been well studied. The performance of this kind of method is still inferior to the current advanced fully-supervised point cloud segmentation approaches.

2.5 Contrastive Learning for Dense Prediction

Recently, contrastive learning methods have drawn much interest as a result of their success in representation learning and other applications [13, 11, 115, 34, 16,

39]. The core idea of these methods is to encourage the feature representations of the matched positive pairs to be close and the ones of un-matched negative pairs to be away from each other in the feature space. This thesis only reviews the most related works that explore contrastive learning to address dense prediction tasks. Most methods in 3D tasks use a contrastive self-supervised pre-training step before the downstream dense prediction task. PointContrast [115] proposed to learn point-level feature representation by a self-supervised pretext task, using the proposed PointInfoNCE loss. It builds the matched positive pairs from two overlapped views of a 3D scene, while the negative point pairs are formed by unmatched points from these matched pairs. The following work [34] improves PointContrast by proposing a ShapeContext-like spatial partition and removing unmatched point pairs that located in the same spatial partition from the original negative point pairs, resulting in the location-aware contrastive learning. By providing more divergent negative point pairs for the contrastive learning, [34] can learn more discriminative point-level feature representations. Very recently, [45] extends contrastive learning for semi-supervised point cloud segmentation, following the consistency regularization strategy [90]. It deals with the task of learning a point cloud semantic segmentation model with a small fraction of the training data being fully-annotated. [45] applies the supervised contrastive learning by constructing the positive pairs as one-by-one matched points from two homologous point clouds under different transformations and constructing the negative pairs as un-matched point pairs with different classes.

In this thesis, we also explore contrastive learning to help learn more discriminative representations, increasing the performance of weakly-supervised point cloud semantic segmentation. We pursue to promote the learned class-specific feature representation to be discriminative in an end-to-end framework. We also want to mine useful information unlabeled data with pseudo labels at the same time.

Chapter 3

Exploring Long-Short-Term Context for Point Cloud Semantic Segmentation

3.1 Introduction

This chapter aims to explore contextual information for supervised point cloud semantic segmentation. Point cloud semantic segmentation aims to assign the semantic class to every point of the given point cloud. The characteristics of point clouds make point cloud semantic segmentation a tough problem. Firstly, the points are always unstructured and unordered, which makes it hard to build the relationship between the points. Secondly, the points collected by 3D sensors are generally sparse with various density in 3D space, which makes it difficult to designing an effective and efficient deep architecture for dense prediction tasks like point cloud semantic segmentation. Thirdly, scale variation of the point clouds increases the difficulties. Fourthly, the number of points is huge, which will cost large GPU memory to capture rich information by considering all the points. Consequently, all the above characters make the point cloud semantic segmentation difficult to incorporate long-range spatial dependencies in a point cloud processing architecture.

As mentioned in Chapter 2, there are a few efforts to solve the above task. One type is to transform the irregular point clouds into collections of 2D images [5, 29] or regular 3D voxels [92, 75, 28]. But such a transformation process suffers from inevitably losing information during quantization.

Another way is to operate on 3D point clouds directly. PointNet [72] first subdivides a large-scale point cloud into small volumetric blocks. After that, it takes

each block as an input and processes each block independently without any connection. Specifically, it learns a high-dimensional feature representation of each point by MLPs and fuses all point-wise features by a symmetric pooling function, forming the global representation of an input block.

This paper belongs to the latter category. However, PointNet-based methods have two limitations: (1) *PointNet does not consider the local structures*, (2) *PointNet cannot capture long-range spatial context beyond the small blocks*. Therefore, the overall accuracy of PointNet is limited in complicated scenes.

There are a large number of works attempting to deal with the first limitation. PointNet++ [74] introduces a hierarchical network to progressively extract local features from neighboring regions. Firstly, depending on a specific distance metric, the input point sets are separated into a number of overlapping local regions. Each local region is associated by a local feature aggregating from the features of points within the local region, considered as a unit of the higher-level layer. With this process repeated, the network can capture the local structure and generate feature representations that contain fine geometric structures from local neighborhoods. Subsequently, based on the pipeline of this hierarchical architecture, [112, 129, 106] concentrate on learning better local feature representations. Moreover, [61] proposes to enhance the feature of the local region by leveraging the correlation between different scale regions. [108] explores the interactions between neighboring points by EdgeConv.

However, there are limited prior works that focus on the second limitation. [24] proposes two mechanisms built upon PointNet to incorporate a larger spatial context for the point cloud semantic segmentation task, which only considers the global feature of each block. But in some challenging scenarios, a coarse holistic global context might not be enough for the classification of ambiguous objects in the scene.

[123] proposes to discover long-range spatial dependencies by a RNN model. The authors design a bidirectional hierarchical RNN model and apply it to an input sequence of spatially adjacent point clouds. Thus, the output of each point cloud contains contextual information among all point clouds of the input sequence. However, the RNN layers perform in sequence-to-sequence mode, therefore, it needs to feed all point clouds of an input sequence before generating the output of the first point cloud. This serial operation is inefficient and inflexible. In addition, neither [24] nor [123] leverages the relationship between points while building local features. Besides, [53] proposes a novel superpoint graph method that can capture long-range context. But the segmentation performance heavily depends on the build of superpoint, which is not optimized during the following training process.

This chapter will present a novel framework to address the second limitation. To exploit long-range spatial context, we introduce a long-term feature bank that stores a rich, position-indexed representation of the point cloud around the current small block within a specified arbitrary range. Intuitively, there are several inherent contextual connections within large objects or between neighboring objects. Exploiting these contexts will be beneficial for solving the ambiguity. For instance, doors are generally inside the walls and connect to the floor, and chairs are often near the table. We learn the feature representation of the information about surrounding scenes and object, and store them in the proposed long-short-term feature bank. As this information provides a supportive context that allows a point cloud segmentation model to infer better the class of a point in the current task block/sample, we expect the long-term feature bank to improve current semantic segmentation models based only on information from a small point cloud block, typically 1-1.5 m by 1-1.5 m along the horizontal direction [72], which we call short-term context. By the way, the reason of taking the small blocks as input is to fit in GPU memory. When integrating the long-term feature bank with local features, we use an attention mech-

anism modified from the non-local operation [105]. Experimental results show that augmenting existing models feeding in small blocks with a long-term feature bank yield consistently increasing performance in the point cloud semantic segmentation task.

3.2 Methods

Our motivation for exploring contextual information for point cloud segmentation results from the observation that natural scenes usually have a reasonable and coherent composition of objects. The presence of one object, even in a spatially disjoint region, can be compelling evidence of the existence of the other. The ability to relate the current status to the context that is distant in space can improve the robustness of the recognition system and help resolve the ambiguity of object labels against noises. With this motivation in mind, we propose a novel Long-Short-Term Context (LSTC) framework by using the proposed long-short-term feature bank to explicitly enable these interactions, therefore enabling existing methods to have the ability to capture long-range spatial dependencies conveniently.

Figure 3.1 depicts our proposed framework. As show in Figure 3.1, the proposed LSTC framework consists of five modules: (a) Encoder, (b) Long-Short-Term Feature Bank Establishment, (c) Feature Bank Fusion Module, (d) Decoder, (e) Predictor. In the following five subsections, a detailed description of each module will be given.

3.2.1 The Encoder Module

With a given input point set, the encoder aims to obtain the local feature representations of all the initial points [72, 108] or points in the abstracted point set [74, 112]. In this work, we build our encoder according to the hierarchical encoder structure proposed by PointNet++ [74], the architecture of which is illustrated in

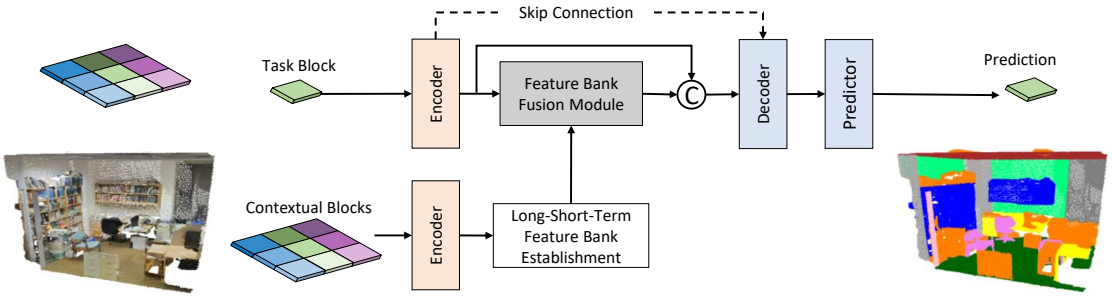


Figure 3.1 : Architecture of our Long-Short-Term Context framework (LSTC). LSTC takes point cloud blocks as input. The task block is used for the point cloud semantic segmentation tasks, and participates in the whole training processing together with its labels. The contextual blocks are used to extract context on-the-fly and only execute the feedforward process. 'Ⓒ' denotes the channel-wise concatenation operation.

Figure 2.4(a) of chapter 2.

The encoder is composed by several *set abstraction* (SA) levels. The SA level processes and abstracts its input point set to a new set with fewer elements by *sampling, grouping* and *pointnet* three layers. For the sake of clarity, we denote N_{l-1} as the number of input points, N_l as the number of output points of SA level l, where $N_l \leq N_{l-1}$. We denotes D_{l-1} and D_l as the dimension of input point features and output point features of SA level l, respectively. The SA level takes an $N_{l-1} \times (3 + D_{l-1})$ matrix as input and output an $N_l \times (3 + D_l)$ matrix, where N_l denotes the number of subsampled points, $(3 + D_l)$ denotes the D_l-dimensional output feature and its 3-dimensional coordinates. More details can be found in [74].

3.2.2 The Long-Short-Term Feature Bank Establishment

Like most point cloud semantic segmentation methods, we also split a large-scale point cloud into several small blocks of $1m$-by-$1m$ along horizontal directions. The long-short-term feature bank, L, is built to provide relevant contextual infor-

mation to aid semantic segmentation of the current point cloud block. The steps are as follows: Firstly, we run an encoder to generate a set of local features for each block. As the blocks are split along the horizontal direction, we take a two-tuple (x, y) as the index of each block. We call $L_{x,y} \in \mathbb{R}^{N_{x,y} \times D}$ the matrix of $N_{x,y}$ D-dimensional local features of block (x, y). Then we let the feature bank $L = [L_{0,0}, ..., L_{0,\mathbf{y}-1}, ..., L_{\mathbf{x}-1,0}, ..., L_{\mathbf{x}-1,\mathbf{y}-1}]$ be a position-indexed list of features, where $\mathbf{x}$ and $\mathbf{y}$ denote the number of divided blocks in two directions. Intuitively, L provides contextual information about the whole point cloud, not only contextual information within each block. We call *short-term* spatial context as the spatial context within one input block, and call *long-term* spatial context as the spatial context in a larger range beyond an input block. Therefore, L is called the *long-short-term feature bank*.

3.2.3 The Feature Bank Fusion Module

For making full use of the long-short-term feature bank to capture the long-short-range spatial context, we introduce a *feature bank fusion module* FBF, which is an attention operator. As shown in Figure 3.2, the fusion module $\mathrm{FBF}(T_{x,y}, \tilde{L}_{x,y})$ takes $T_{x,y}$ and $\tilde{L}_{x,y}$ as inputs, where $T_{x,y} \in \mathbb{R}^{N_{x,y} \times D}$ is the matrix of features generated from the current task block (x, y) and $\tilde{L}_{x,y}$ is $[L_{x-\lfloor \frac{w}{2} \rfloor, y-\lfloor \frac{w}{2} \rfloor}, ..., L_{x-\lfloor \frac{w}{2} \rfloor+(w-1), y-\lfloor \frac{w}{2} \rfloor+(w-1)}]$, a subset of L covering the current task block (x, y) with window size $w \times w$, stacked into a matrix $\tilde{L}_{x,y} \in \mathbb{R}^{N \times D}$, where $N = \sum_{x'=x-\lfloor \frac{w}{2} \rfloor}^{x-\lfloor \frac{w}{2} \rfloor+(w-1)} \sum_{y'=y-\lfloor \frac{w}{2} \rfloor}^{y-\lfloor \frac{w}{2} \rfloor+(w-1)} N_{x',y'}$. We use $T_{x,y}$ to attend to features in $\tilde{L}_{x,y}$, and add the attended information back to $T_{x,y}$ via a shortcut connection. Specifically, we replace the self-attention of the standard non-local block with attention between the task block features $T_{x,y}$ and the long-short-term feature bank subset $\tilde{L}_{x,y}$.

Intuitively, the feature bank fusion module computes an updated version $T'_{x,y}$ of the task block features $T_{x,y}$ by relating them to both the short-term context

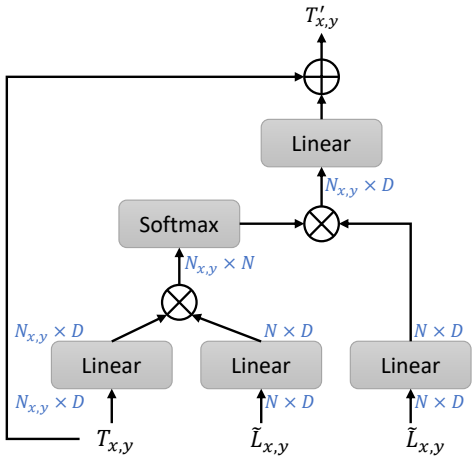


Figure 3.2 : Feature Bank Fusion Module. $T_{x,y}$ denotes the feature representations of the task block indexed as (x, y). $\tilde{L}_{x,y}$ denotes a subset of the feature bank, containing the contextual feature vectors of the task block (x, y) and its surrounding blocks. D is defined as the dimension of the feature vectors. $N_{x,y}$ and N denote the number of feature vectors of $T_{x,y}$ and $\tilde{L}_{x,y}$, respectively. '$\otimes$' denotes matrix multiplication and '$\oplus$' demotes element-wise sum.

features within the current task block and the long-term context features in a larger range surrounding the current task block. In other words, the feature bank fusion module adaptively captures the long-short-term spatial context from the proposed long-short-term feature bank. The output $T'_{x,y}$ is then channel-wise concatenated with $T_{x,y}$ and used as input into the following decoder.

3.2.4 Decoder and Predictor Modules

The decoder is used to propagate the features of the abstracted points to the original points. This work applies a hierarchical decoder structure [74], which contains a number of *feature propagation* (FP) levels. In a FP level, we first interpolate the feature representations of the N_l points at the positions of the N_{l-1} points. Then we concatenate the interpolated features on N_{l-1} points with the point features

from the corresponding SA level through skip connection, followed by a multi-layer perceptron.

Finally, we achieve the predictions by applying several shared fully connected layers, together with the following softmax layer, to the propagated features of the original point set.

3.2.5 Implementation Details

In our implementation, we use four SA levels for the encoder and four FP levels for the decoder. The numbers of points in the SA layers are 1024, 256, 64, 16, respectively. We use the same encoder architecture to compute the local features T from the current task block and the long-short-term features L from its contextual blocks. Parameters are not shared between these two encoders. The encoders are both initialized by a pre-trained baseline model, which does not use the long-short-term feature bank, but a PointNet++ model in actuality. Considering the computational and memory complexity of back-propagating through the long-short-term feature bank, we treat the encoder used to compute L as a fixed component that is trained offline and not updated subsequently. Hence, L can be efficiently computed in a single pass over the whole point cloud by the encoder.

3.3 Experiments

3.3.1 Datasets and Evaluation Metrics

Datasets. We carry out experiments and evaluate our approaches on the Stanford Large-Scale 3D Indoor Spaces (S3DIS) dataset. **S3DIS** [2] was collected from six indoor areas that are primarily used for education and offices. These areas, marked as Area-$i, i = \{1, ..., 6\}$, are distributed in three different buildings, with Area-1, Area-3 and Area-6 coming from the first building, Area-2 and Area-4 originating from the second building, and Area-5 coming from the third building. There

are several different types of rooms in each area, such as lobbies, hallways, offices, conference rooms and so on. Each room is represented by a single point cloud and in total, there are 271 point clouds in S3DIS. All point clouds are automatically generated using a Matterport scanner without any manual intervention, providing their XYZ coordinates and RGB values. For semantic segmentation, this dataset provides full per-point annotations with 13 classes (structural elements: ceiling, floor, walls, beams, columns, Windows, doors, and movable elements of other elements: tables, chairs, sofas, bookshelves, boards, and clutter). In our experiments, we use point clouds of Area-5 for testing and of the rest five areas for training, following [92]. Because the building where Area-5 comes from does not contain other areas, which is more fair to evaluate the generalizability performance of approaches.

Evaluation Metrics. We evaluate quantitative and qualitative performance on S3DIS testing set. We use the following evaluation metrics in our experiments: the intersection over union (IoU) of every semantic class and the mean of IoUs over all classes (mIoU). For each semantic class c, its IoU is computed as follows:

$$IoU(c) = \frac{TP(c)}{T(c) + P(c) - TP(c)} \tag{3.1}$$

where $T(c)$ denotes the number of positives in the ground truth, *i.e.* points with given labels being c. $P(c)$ denotes the number of positives in the predictions, *i.e.* points with predictions being c. $TP(c)$ indicates the number of true positives, *i.e.* points with both ground-truth labels and predictions being c.

3.3.2 Experimental Settings

Data Preparation. For fair comparison, we follow the settings described in [24]. To get more representative blocks, we pre-process all training point clouds by uniformly downsampling them with a grid size of $0.03m$. As introduced in Section 3.2, our proposed LSTC framework takes small point cloud blocks as input. Therefore, during training, we divide each point cloud into several point cloud blocks of

size $1m \times 1m$ along with two horizontal directions separately. As a result, all point cloud blocks extend over the entire height of a point cloud. There is an overlap of $0.5m$ between two adjacent blocks. For each block, we randomly sample 4096 points and hold the coordinates (x, y, z) , color (r, g, b) and normalized coordinates (x', y', z') of each point. During testing, we use the same data pre-processings as in training except for splitting each point cloud into non-overlapping blocks. Each block is evaluated only once and then uses knn interpolation to get the predicted results of the original point clouds.

Training Configuration. For all experiments, we use the 9-dimensional vector $[x, y, z, r, g, b, x', y', z']$ as input point features. We use the Adam optimizer to optimize our model and set 0.001 as the initial learning rate. Before feeding into the feature bank fusion module, the dimensionality of features is reduced to 128 by a linear layer. We also add one dropout layer after each linear layer of our feature bank fusion module and set 0.2 as the dropout rate.

3.3.3 Performance on S3DIS

As mentioned in [92], if the training set and testing set cover point clouds from the same building, the generated segmentation results will be better than usual, resulting in unfair evaluation for the generalizability performance. For fair comparisons, we only compare the segmentation results that evaluate on Area 5 of S3DIS, *i.e.* , training models on the first two buildings and testing on the third building.

First, we compare the proposed long-short-term context model LSTC-w (w denotes the context window size) with the baseline model PointNet++. We use the official code to train the PointNet++ model on S3DIS and produce the segmentation results following the testing setting as defined in Section 3.3.2. The LSTC-1 model is equal to a 'degraded' version of our long-short-term context model, which uses a short-term feature bank that only contains contextual features within the

Table 3.1 : **Comparison results on S3DIS.** The results are tested on our testing data of S3DIS Area-5. Bold fonts indicate the best results on the metrics of mIoU(%) and per-class IoU(%).

Method	mIoU	Ceiling	Floor	Wall	Beam	Column	Window	Door	Table	Chair	Sofa	Bookcase	Board	Clutter
PointNet++ [74]	50.21	90.38	96.94	75.07	0.00	14.40	47.79	15.66	66.48	72.95	19.71	61.04	48.68	43.58
G+RCU [24]	45.06	91.37	98.06	71.90	0.11	6.26	51.58	10.82	66.07	60.04	5.56	49.63	35.02	39.32
3DRNN [123]	53.40	**95.20**	**98.60**	**77.40**	**0.80**	9.83	52.70	**27.90**	**78.30**	76.80	27.40	58.60	39.10	**51.00**
LSTC-1(ours)	51.42	90.54	96.93	75.00	0.00	**15.17**	49.29	18.02	67.88	75.89	25.66	60.96	49.90	43.19
LSTC-2(ours)	**53.46**	92.10	97.45	75.34	0.00	14.60	**54.31**	17.64	70.43	**79.49**	**31.11**	**62.53**	**53.01**	46.92

task block. As shown in Table 3.1, our LSTC-1 model performs better than PointNet++ by increasing 1.21%mIoU. Especially for the categories that are various in shape, such as table, chair, sofa, using the short-term feature bank leads to 1.4%, 2.94%, 5.95% improvement over the baseline model respectively. This demonstrates the baseline model can get benefit from the feature bank, which provides useful contextual information.

To validate the benefit of incorporating long-term contextual information, we increase the contextual window size to 2. Comparing LSTC-1 and LSTC-2, we can observe the LSTC-2 model further improve the results: 2.04%mIoU, 2.55% for table, 3.6% for chair, 5.45% for sofa. And compared to the backbone PointNet++, our LSTC-2 model outperforms PointNet++ by 3.25%mIoU, 3.95% for table, 6.54% for chair, 11.4% for sofa, which is about doubles the performance of LSTC-1. Besides, for the categories with large size or ambiguous structure, the LSCT-2 model outperforms PointNet++ in large margins, e.g. increasing by 6.52% for window , 4.33% for board.

We also compare the proposed LSTC model with methods that also exploit long-range contextual information and report results on S3DIS Area 5. The G+RCU [24] can be considered as a particular instance of the proposed framework, which equals

to build the feature bank with the global features of each block and the context window size is 2. As the global features is too coarse for segmentation task, using our more representative feature bank leads the LSTC-2 model to outperform G+RCU by a large margin (8.4%mIoU). 3DRNN [123] can also be regarded as an instance of our framework, which unrolls the point-wise features of each block to form the feature bank and the context window size is 3. Although 3DRNN using more detailed and larger-range context, our LSTC-2 model still makes a marginal improvement in overall performance (+0.06%mIoU) and is superior in several difficult categories like column and bookcase.

Besides, since our feature bank isn't optimized in the training stage for the segmentation task, it is convenient for the LSTC model to capture the context in an arbitrary long-range with only a little extra computational cost in the fusion module. Specifically, the number of parameters of the LSTC models increases by 18.7% over PointNet++ (from 968,942 to 1,150,126). This advantage is remarkable in terms of limited computational resources.

3.3.4 Qualitative Results and Discussions

The qualitative experimental results of our models applied to S3DIS are shown in Figure 3.3. To clearly demonstrate the quality of the segmentation results, we provide both the segmentation results and difference maps compared with the ground truths.

Robust to abnormal features. We can note from Figure 3.3 that the segmentation results of our baseline model PointNet++ [74] contains many unsmooth predictions on object parts with abnormal features or ambiguous structure. For example, PointNet++ misclassifies many points of the wall part with blue marks in the first sample, the wall part with stain in the third sample and the windows in the last sample. Since PointNet++ does not consider the interaction of local

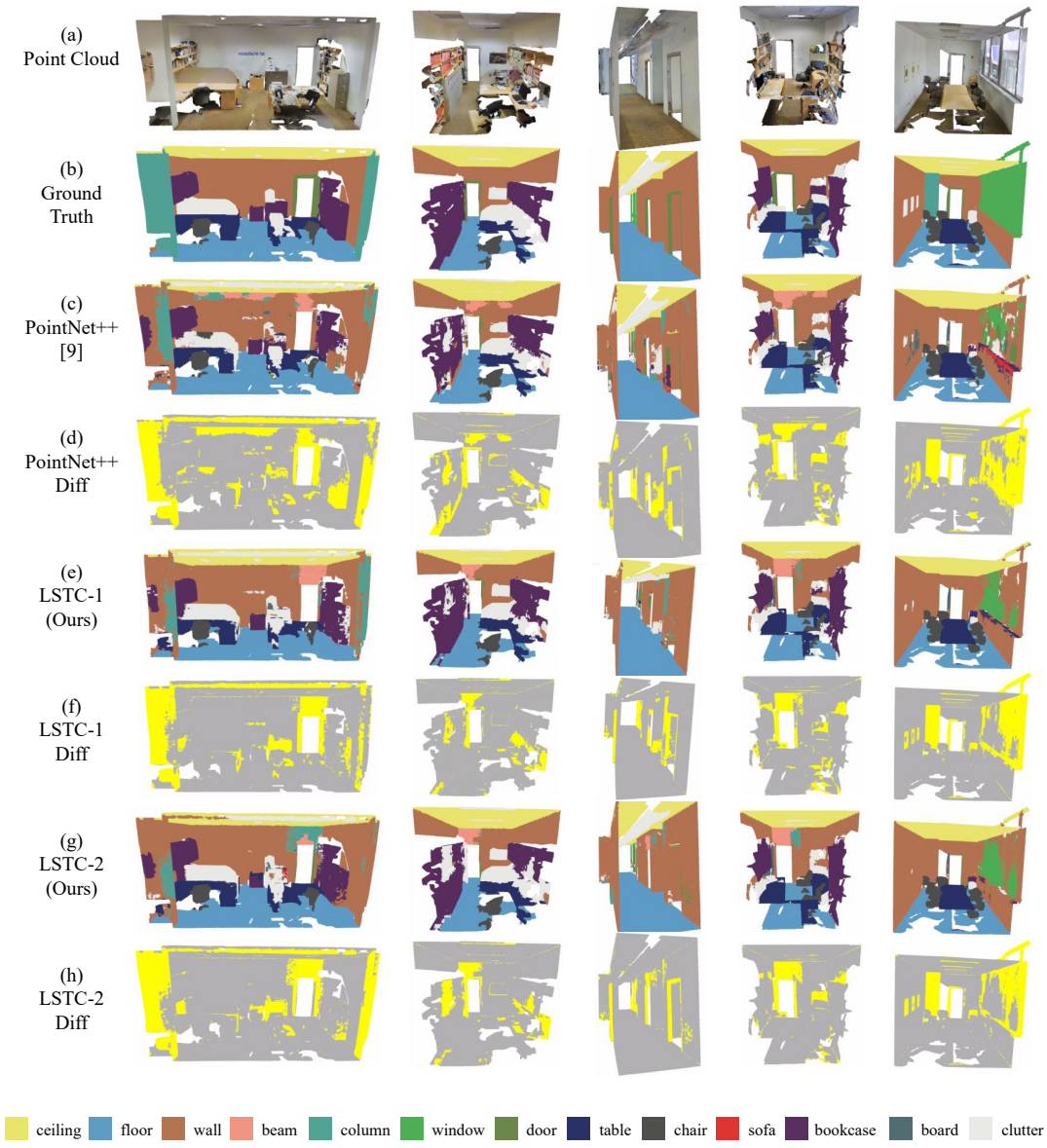


Figure 3.3 : Visualization of segmentation results on S3DIS. Top to bottom: (a) point cloud, (b) ground truth, (c) prediction of PointNet++ [74], (d) differences between ground truth and prediction of PointNet++, (e) prediction of our LSTC-1 model, (f) differences between ground truth and prediction of our LSTC-1 model, (g) prediction of our LSTC-2 model, (h) differences between ground truth and prediction of our LSTC-2 model. In rows (d), (f) and (h), yellow points indicate incorrect predictions.

features within a spatial distance, it is easy for PointNet++ to make mistakes on these abnormal parts of objects. Our LSTC models can slash these errors by exploiting contextual information and leveraging the interaction of local features. We can observe that our LSTC-1 model reduces the errors on objects parts with abnormal features or ambiguous structure, and generates more smooth segmentation results than PointNet++ by exploiting only the short-term context within the task block. When increasing the context window size to exploit the long-short-term context, our LSTC-2 model can further reduce these errors and obtain the best performance on most categories.

Refinement on boundary regions. Besides, we also observe that our models can improve the predictions of the boundary regions of objects. We can see from Figure 3.3 that the baseline model PointNet++ misclassifies many points in the boundary regions between adjacent objects, such as the boundary regions between walls and ceilings, the boundary regions between walls and doors, the boundary regions between walls and bookcases, and the boundary regions between walls and windows. Our LSTC-1 model and LSTC-2 model improve the segmentation predictions of the boundary regions in most of the shown point cloud samples. This demonstrates that our method can learn more discriminative feature representations by fusing contextual information and enhancing the local features with related contextual features.

3.4 Conclusions

This chapter presents LSTC, a novel 3D point cloud semantic segmentation framework to enable existing segmentation methods to capture long-range spatial dependencies. We show that using the proposed long-short-term feature bank leads the baseline model to a significant performance gain, with only spends a little extra computational cost. It is reasonable to suppose that a better feature bank can result

in better performance.

Chapter 4

Weakly-supervised 3D Keypoint Detection for Point Cloud Registration

4.1 Introduction

We describe a fully-supervised framework for 3D point cloud semantic segmentation in the previous chapter, which requires point-wise fully annotated datasets for training. However, designing such point-wise fully annotated 3D datasets is extremely expensive, much more than the cost of 2D due to the large scale and annotating difficulty of 3D data. Therefore, it is highly necessary to develop weakly-supervised methods that do not require fully annotated datasets for 3D vision tasks.

In this chapter, we aim to develop a weakly-supervised 3D keypoint detection method applied to the point cloud registration task. 3D keypoint detection is an essential part of various 3D scene understanding tasks, including 3D reconstruction, 3D object recognition, 3D shape retrieval. The goal of 3D keypoint detection is to pursue a repeatable and distinctive 3D local representation from 3D data, e.g., point cloud and 3D mesh, which can be used to establish correspondences between 3D surfaces. Although more and more researchers have been working on 3D keypoint detection in recent years, it's still an open research topic. Since a point cloud is composed of a group of discrete points having nonuniform densities, it is challenging to define what keypoints are in a 3D point cloud. In addition, it's impossible for people to manually label the position of a keypoint in a 3D point cloud. Lack of annotated datasets limits researchers from leveraging powerful supervision learning methods on 3D keypoint detection.

Lots of relevant research focuses on 3D local descriptors but does not address the detection of keypoint positions. So far, many 3D descriptors have been proposed. [48, 78, 97] use histograms to estimate the similarity of keypoints by figuring out the number of points falling in every spatial bin or by taking into account the surface normal property. These methods are designed for specific applications, and it's hard to extend to new scenes. [125, 82] learn 3D local features by making use of a Siamese 3D CNN and has made significant progress. They both take pairs of 3D local patches as input. Therefore, dense keypoint annotations are required, labeling a collection of matched 3D local patches. During training, they create pairs of matching local patches as positives and pairs of non-matching ones as negatives, fed to their loss functions to optimize the networks. Eventually, they learn the 3D ConvNet-based descriptors.

For keypoint detector, it's common to use a hand-crafted saliency function by combining the domain knowledge, or uniformly selecting some local patches and taking their centers as keypoint. Differently, [98] proposes a descriptor-specific keypoint detector, which casts 3D keypoint detection as a classification problem in terms of whether the points can be matched by a pre-defined 3D descriptor. The performance of this method depends on the pre-defined 3D descriptor. However, learning an excellent 3D descriptor needs a large amount of annotated keypoints, and thus causes a chicken-egg problem.

In this work, we explore whether one can simultaneously learn a 3D keypoint detector and feature extractor in a unified framework. Inspired by [25], which detects keypoints and learns keypoint features from depth images by an end-to-end framework, we present a KeyPoint Siamese Network based framework (KPSNet) to simultaneously detect 3D keypoints and learn their feature representations directly from 3D point clouds. The KPSNet receives as input pairs of point clouds and their relative rigid transformation matrix. Each branch of the KPSNet contains a

keypoint detection sub-network to predict keypoints and a feature extraction sub-network to learn 3D features. These two branches are the same. We design an alignment module to establish correspondence between the two branches in order to jointly optimize the whole network. In other words, the alignment module generates pairs of matching and non-matching samples and labels them on-the-fly. Then we train the network to minimize the distance between matched pairs in feature space and maximize the distance between unmatched pairs.

In the following, we introduce in detail the proposed framework in Section 4.2, including the architecture, training, and joint optimization. Then we present the implementation results in Section 4.3. At last, we summarize the advantages and limitations of the proposed weakly-supervised method in Section 4.4.

4.2 Methods

4.2.1 Architecture of KPSNet

As shown in Figure 4.1, we demonstrate the whole architecture of our proposed KPSNet. Each branch takes an entire point cloud, P^m, where $m \in \{1, 2\}$, as input and contains a keypoint detector and a feature extractor. It firstly generates N point clusters from the point cloud similar to [74, 44]. We consider the centroids of these point clusters as candidates of the keypoints (*the red points*) and define the corresponding point clusters as their feature support regions (*the yellow balls*). These candidates, along with their feature support regions , are fed into the keypoint detector and the feature extractor, respectively. The keypoint detector contains two multi-layer perceptions (MLPs) modules [72] and a max pooling layer. For each input point cloud, the keypoint detector outputs a vector s^m, signifying the probability of each candidate being a keypoint. The feature extractor contains three MLP modules with skip link feature concatenation. It outputs feature representations $\mathbf{f}^m$ of the candidates $\mathbf{x}^m$. The keypoint detector and the feature extractor share the

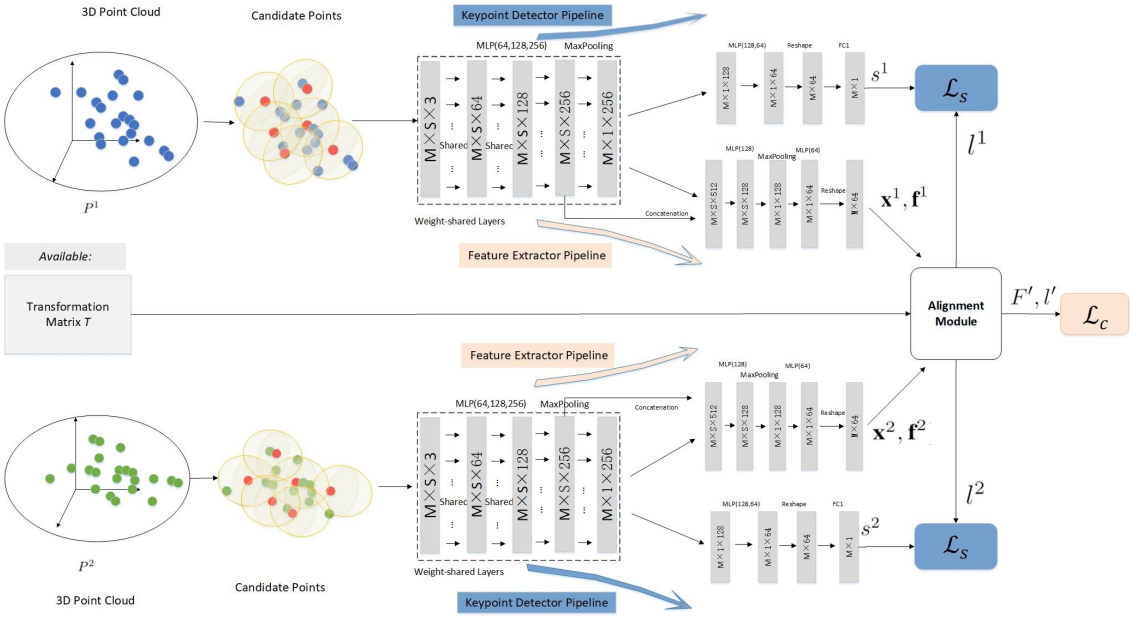


Figure 4.1 : Architecture of our KPSNet. Our KPSNet takes pairs of point clouds $\{P^1, P^2\}$, associated with the weak labels T (*i.e.* their relative rigid transformation matrices), as input and generates N candidates from each input point cloud. These candidates (*the red points*), along with their support regions (*the yellow balls*), are fed into a MLP module (*MLP(64,128,256) denotes there are three convolution layers in this MLP module and the number of filters are 64, 128 and 256 respectively*), followed by a max pooling layer. The keypoint detector branch generates predictions s^m of each candidate, while the keypoint feature extractor branch generates features of each candidate. For each input point cloud, we pass the predictions s^m and labels l^m generated from the alignment module to the score loss $\mathcal{L}_s$. The features from both two input point clouds $\mathbf{f}^1$ and $\mathbf{f}^2$ are organized into pairs $F' = \{F'_n | n = n = 1, ..., N\}$, along with the generated labels l' of all pairs F', are used to compute the contrastive loss $\mathcal{L}_c$. It is worth noting that the weights from the two branches of our KPSNet are shared. More details on the notations are introduced in Section 4.2.2.

first MLP module $MLP(64, 128, 256)$.

These two branches of our KPSNet share all structures and weights and are linked

by an alignment module. The alignment module establishes the correspondence between the two sets of candidates using the known rigid transformation matrix T, thereby generating positive and negative pairs of samples $\{F', l'\}$. These pairs of samples are transferred to a contrastive loss $\mathcal{L}_c$ to optimize the network, so that the positive pairs are close to each other and the negative pairs are far from each other in the feature space. In addition, the alignment module also generates a label for each candidate, forming a label vector l^m, where $m \in 1, 2$ for all candidates from each input. The score loss $\mathcal{L}_s$ uses these labels, l^1 and l^2, to lead the keypoint detector to find as many keypoints as possible. Apparently, our proposed method does not require separately annotated keypoints or local patches pairwise, which is cost-saving and easily extended to new scenes.

4.2.2 Model Training

The proposed KPSNet takes pairs of point clouds $\{P^1, P^2\}$ and their weak labels T, their relative rigid transformation matrices, as input. Each pair of point clouds must have some overlap. From each point cloud, KPSNet generates a set of candidates, $K^m = \{(\mathbf{x}_n^m, s_n^m, \mathbf{f}_n^m) | n = 1, ..., N\}$, where $m \in \{1, 2\}$ corresponds to the pair of point clouds, N is the number of candidates, $\mathbf{x}_n^m = (x_n^m, y_n^m, z_n^m)$ are the 3D coordinates of the points, s_n^m is the output of the keypoint detector which signifies the probability of the candidate being a keypoint, and $\mathbf{f}_n^m$ is the corresponding feature vector.

The Alignment Module Considering the point cloud registration task, we desire that a keypoint should be repeatable and distinctive. To be specific, if a keypoint is detected somewhere in one point cloud, then we desire that it can always be detected no matter what transformations are performed on the coordinate system of the point cloud. In addition, for two different point clouds with different densities and coordinate systems, keypoints are expected to be found in their overlapping

region as many as possible. Under this scenario, we design the alignment module to establish the correspondences between the two sets of candidates and generate labels on-the-fly for training the whole model.

The alignment module receives the coordinates and feature vectors of the two sets of candidates,$\{\mathbf{x}^1, \mathbf{f}^1\}$ and $\{\mathbf{x}^2, \mathbf{f}^2\}$. It firstly transforms $\mathbf{x}^1$ and $\mathbf{x}^2$ to the same coordinate system using the given rigid transformation matrix T and calculates the Euclidean distance between any two points from $\mathbf{x}^1, \mathbf{x}^2$ respectively. Then for each candidate $\mathbf{x}_i^1 (i = 1, ..., N)$, we find the closest candidate $\mathbf{x}_j^2 (j = 1, ..., N)$ on the basis of the Euclidean distance and generate the n^{th} pair of features $F_n' = (\mathbf{f}_i^1, \mathbf{f}_j^2)$. The pair of features F_n' is treated as a positive pair, *i.e.* $l_n' = 1$, in case the distance between $\mathbf{x}_i^1$ and $\mathbf{x}_j^2$ is under a small threshold τ, denoted as $d(\mathbf{x}_i^1, \mathbf{x}_j^2) < \tau$. At the same time, we take $\mathbf{x}_i^1$ and $\mathbf{x}_j^2$ as the positive samples of P^1 and P^2 respectively for training the keypoint detection sub-network and denote as $l_i^1 = l_j^2 = 1$. Otherwise, we label F_n', $\mathbf{x}_i^1$ and $\mathbf{x}_j^2$ as negative, denote as $l_n' = l_i^1 = l_j^2 = 0$. By doing so, we get the required labels for training the keypoint detector and the feature extractor.

Joint Optimization We aim to simultaneously learn a keypoint detector and a feature extractor for point cloud registration. As mentioned above, the desirable keypoints should have properties like repeatability, view-invariant and distinctive. To achieve this objective, we present the following multitask loss similar to [25]:

$$\mathcal{L}\left(\{K^1, K^2\}\right) = \alpha \mathcal{L}_c\left(F', l'\right) + \beta \mathcal{L}_s^1\left(s^1, l^1\right) + \beta \mathcal{L}_s^2\left(s^2, l^2\right), \tag{4.1}$$

where $F' = \{F_n' | n = 1, ..., N\}$ is the set of feature pairs generated by the alignment module, $l' = \{l_n' | n = 1, ..., N\}$ is the set of labels of F', $s^1 = \{s_n^1 | n = 1, ..., N\}$, $s^2 = \{s_n^2 | n = 1, ..., N\}$ are outputs of the keypoint detector and l^1, l^2 are labels of s^1, s^2, respectively. α, β are weighted factors. L_c is the improved contrastive loss that optimizes over the feature representation of pairs of the keypoints. Its basic

principle is to minimize the feature distance of positive pairs and to maximize the feature distance of negative pairs. We define the contrastive loss as follows:

$$\mathcal{L}_c\left(F', l'\right) = \frac{\sum_{n=1}^{N} l'_n \left\|\mathbf{f}_n^1 - \mathbf{f}_n^2\right\|^2}{2N_{pos}} + \frac{\sum_{n=1}^{N} \left(1 - l'_n\right) \max\left(0, \delta - \left\|\mathbf{f}_n^1 - \mathbf{f}_n^2\right\|\right)^2}{2N_{neg}} \quad , \quad (4.2)$$

where, N_{pos} and N_{neg} are the number of positive and negative pairs, respectively. N is the total number of pairs of samples, *i.e.* $N = N_{pos} + N_{neg}$. δ is a margin to control the lower bound on the distance between negative pairs. Here, we conveniently denote $\mathbf{f}_n^1, \mathbf{f}_n^2$ as the n^{th} feature pair. Considering the class sizes of positive and negative pairs are imbalanced, we normalize the contribution of each class to the loss by their number proportion. We should notice that N is also equal to the number of keypoint candidates that generated from each input point cloud. In general, N is set to much less than the number of points in each input point cloud for effective and efficient computation. A larger N will result in more similar keypoint candidates with their support regions highly overlapped, which helps little to the model learning but increases much computational cost.

$\mathcal{L}_s^m$ is the score loss of the prediction from the keypoint detection sub-network. As it's hard to define what a keypoint should be like, in this paper, we take task-specific keypoints as the objective. In other words, we train the keypoint detection sub-network to select points for which we can find positive correspondence between two point clouds taken from different positions. So we only consider the positive points and want to detect as many as positive keypoints as we can. Thus, we define the score loss as follows:

$$\mathcal{L}_s^m\left(s^m, l^m\right) = -\frac{\gamma \sum_{n=1}^{N} l_n^m \log s_n^m + 1}{N_{pos} + 1}, \quad (4.3)$$

where $m \in \{1, 2\}$ corresponds to the pair of input point clouds, γ is a regularization parameter.

4.3 Experimental Setup and Results

4.3.1 Datasets and Experimental Settings

Datasets. We use several open-source RGB-D reconstruction datasets collected by [125] to design our training samples with weak labels for training. These RGB-D datasets are SUN3D [114, 31], 7-Scenes [85], RGB-D Scenes v2 [52], BundleFusion [18] and Analysis by Synthesis [99]. In total, these datasets contain 62 scenes, each scene containing one or more RGB-D video sequences. Same as [125], we split these scenes into 54 scenes for training and 8 scenes for testing. In order to train our model, we need to construct the point cloud dataset with weak labels. We firstly create a series of 3D point clouds from the RGB-D video sequences of all training scenes. Each point cloud is reconstructed and integrated from 50 depth frames. Then we prepare our training dataset by creating pairs of overlapped point clouds, along with their rigid transformation matrices, following the method of constructing the 3DMatch geometric registration benchmark dataset. To obtain relatively balanced positives and negatives for the proposed framework, we filter pairs of point clouds whose overlapping regions are too small (less than 30% of both two point clouds) or too large (more than 70% of either of two point clouds). In total, we collect 22.8K pairs of point clouds to form our training dataset, and the weak label of each pair is defined as the corresponding rigid transformation matrix. We use the geometric registration benchmark dataset provided by [125] as our testing dataset, which is constructed from the 8 testing scenes listed in Table 4.1.

Experimental Settings. In our experiment, each input point cloud is pre-processed to contain 10000 points. The number of generated candidates in each point cloud N is set to 512 for training, and 1024 for evaluation unless otherwise specified. The support region of the candidate is set to a ball with a radius of 0.2m. The threshold τ used in the alignment module is set to 0.05m .The weighted factors

α is set to 0.5, and β is set to 0.2. The margin δ and the regularization parameter γ are both set to 0.5.

Evaluation Metrics. In order to validate our method, we apply the 3D keypoints generated by the well-trained KPSNet to the point cloud registration task, and evaluate on the geometric registration benchmark dataset. More specifically, given a 3D point cloud from the testing dataset, we use the well-trained KPSNet to predict the coordinates and feature representations of a set of 3D keypoints. Then a nearest neighbor matching is performed on them and finally RANSAC method is used on these obtained nearest neighbor matches for estimating a usable rigid transformation between every two point clouds implemented by 3dmatch-toolbox (`https://github.com/andyzeng/3dmatch-toolbox`). We should note that the number of RANSAC iterations is limited to 5000 and we do not perform any subsequent refinement like using Iterative Closest Point (ICP).

We use two evaluation metrics, recall and precision, same as 3DMatch [125]. The evaluation metrics are computed as follows: For a pair of non-consecutive testing point clouds (P^i, P^j), where P^i is smaller than P^j, this pair is valid only if their overlap in alignment covers at least 30% of P^i. Then we can get the ground-truth rigid transformation matrix T_{ij} of that pairs and the set of point-to-point ground-truth correspondences $\mathcal{K}_{ij}^*$. The estimated rigid transformation matrix $\hat{T}_{ij}$ associated with (P^i, P^j) is positive if at least 30% of $\hat{T}_{ij}P^i$ overlaps with P^j. If a positive $\hat{T}_{ij}$ can align the ground-truth correspondence pairs, then it is a true positive, formulated as Eq.(4.4),

$$\frac{1}{\left|\mathcal{K}_{ij}^*\right|} \sum_{(q^i, q^j) \in \mathcal{K}_{ij}^*} \left\| \mathbf{T}_{ij} q^i - q^j \right\|^2 < \mathcal{E}^2 \tag{4.4}$$

where $\mathcal{K}_{ij}^*$ denotes the set of point-to-point ground-truth correspondences of point cloud pair (P^i, P^j) computed using the ground-truth rigid transformation matrix T_{ij}, q^i and q^j denotes the matched two points respectively. $\mathcal{E}$ is a threshold and set

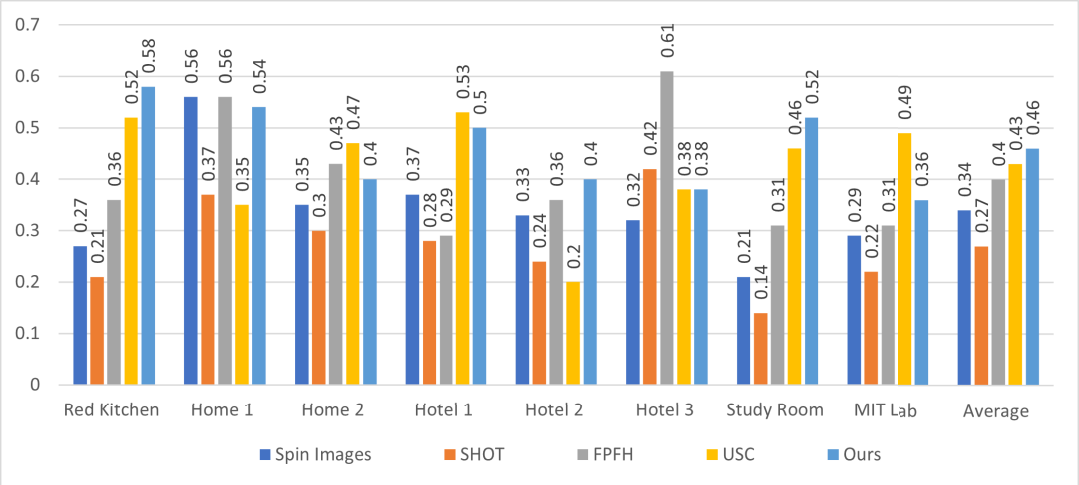
Table 4.1 : Overall performance on 3D-match benchmark. Recall and precision are computed across all scenes. Bold fonts mark the top performance.

	Spin Images [48]	SHOT [79]	FPFH [78]	USC [97]	Ours
Recall	0.34	0.27	0.40	0.43	**0.46**
Precision	0.18	0.17	0.22	0.27	**0.29**

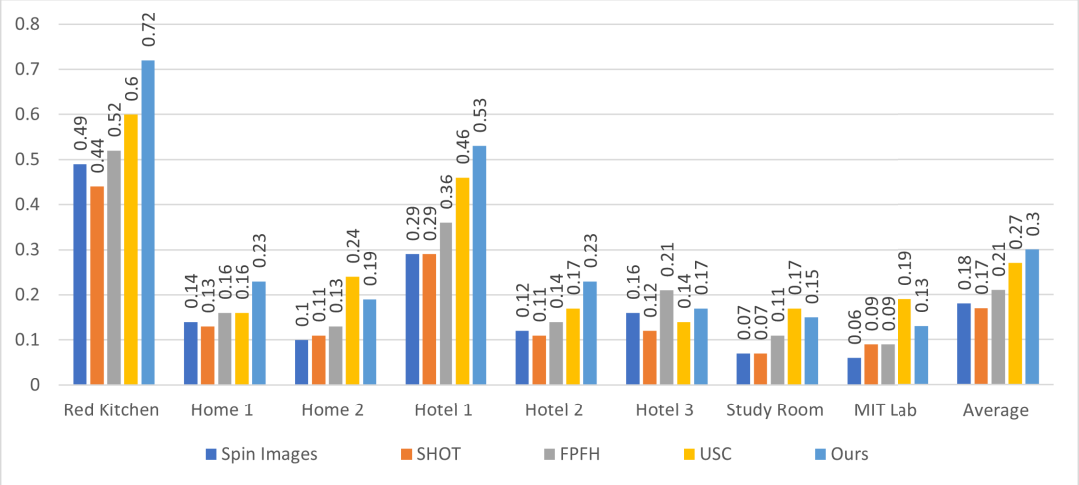
to $\mathcal{E} = 0.2$ meters in all experiments. Finally, the metrics of recall and precision are computed on each scene of the benchmark dataset for evaluation, respectively.

4.3.2 Performance on 3DMatch Benchmark

We combine the 3D keypoints generated by our KPSNet with RANSAC to form our KPSNet-based point cloud registration approach and run pairwise point cloud registration on every pair of point clouds from the 3DMatch benchmark dataset. As our KPSNet are trained without using point-wise 3D keypoints annotations, we compare the registration results using the presented KPSNet against the hand-crafted baselines that plug the hand-crafted descriptors of Spin Images [48], SHOT [85], FPFH [78], USC [97] into the RANSAC based point cloud registration pipeline. These baselines are all based on hand-crafted 3D local feature descriptors, therefore, do not require 3D keypoints annotations. The 3DMatch benchmark dataset consists of point clouds from eight scenes, one from 7-scenes [85], and the others from SUN3D [114] datasets. We report the overall recall and overall precision across all scenes in Table 4.1. As observed from Table 4.1, our proposed KPSNet outperforms all the hand-crafted methods in overall recall and overall precision. We also provide the recall and precision on every scene in Figure 4.2. We can see, from Figure 4.2, our KPSNet obtain the first place of recalls on three scenes of Red Kitchen, hotel and Study Room. We also observe the recall of our KPSNet on the scene of Hotel3 is less



(a) Per-scene Recall



(b) Per-scene Precision.

Figure 4.2 : Per-scene performance on 3DMatch benchmark. There are eight scenes in the testing dataset. *Red Kitchen* comes from [85], and the rest seven scenes come from [114].

than the highest recall achieved by FPFH [78] by a large margin of 0.23. The main reasons may be that: 1) point clouds in Hotel3 are not as compact as other scenes, which results in that only a small number of keypoints generated by our method are distributed in the overlapping area of the two point clouds to be registered, thereby more susceptible to incorrect keypoints; 2) point clouds in Hotel3 contain

many similar or repetitive structures, such as bed and wall, making our keypionts less distinctive. These two factors cause our method to perform suboptimally in Hotel3. However, our KPSNet performs best on average recall across all scenes, which demonstrates the robustness of our method. As for the precision, our KPSNet obtains the first place on four scenes and the second place on the rest scenes. Overall, our method obtains a competitive performance on all scenes. We should also notice that we only use 1024 candidate keypoints to obtain these results. Generally, the precision increases with the number of keypoints. Therefore, it is for sure that with more keypoints and more iterations, our proposed KPSNet can achieve more remarkable performance.

4.3.3 Qualitative Results and Discussions

Point Cloud Registration Results. To validate our KPSNet, we combine our KPSNet and RANSAC to process the point cloud registration task. Figure 4.3 visualizes the point cloud registration results in eight scenes of 3DMatch benchmark testing datasets. i) *Robust to overlapping ratios.* As exposed in Figure 4.3, the given two point clouds are overlapped in various ratios. In general, small overlapping ratios will increase the difficulty of point cloud registration using hand-craft keypoints, because smaller overlapping ratios result in more keypoint candidates located in non-overlapped regions, harmful for the estimation of transformation matrix. Using our KPSNet can achieve excellent point cloud registration results for point clouds with various overlapping ratios, even though very small overlapping ratio, such as the third group of the right sub-figure of Figure 4.3. This is due to our leaned keypoint detector, which predicts the probability of a keypoint candidate being in the overlapped regions, and helps to remove keypoints with low confidence. This demonstrates our KPSNet is robust to overlapping ratios. ii) *Robust to geometric structure.* Registering two point clouds that have simple geometric structure with

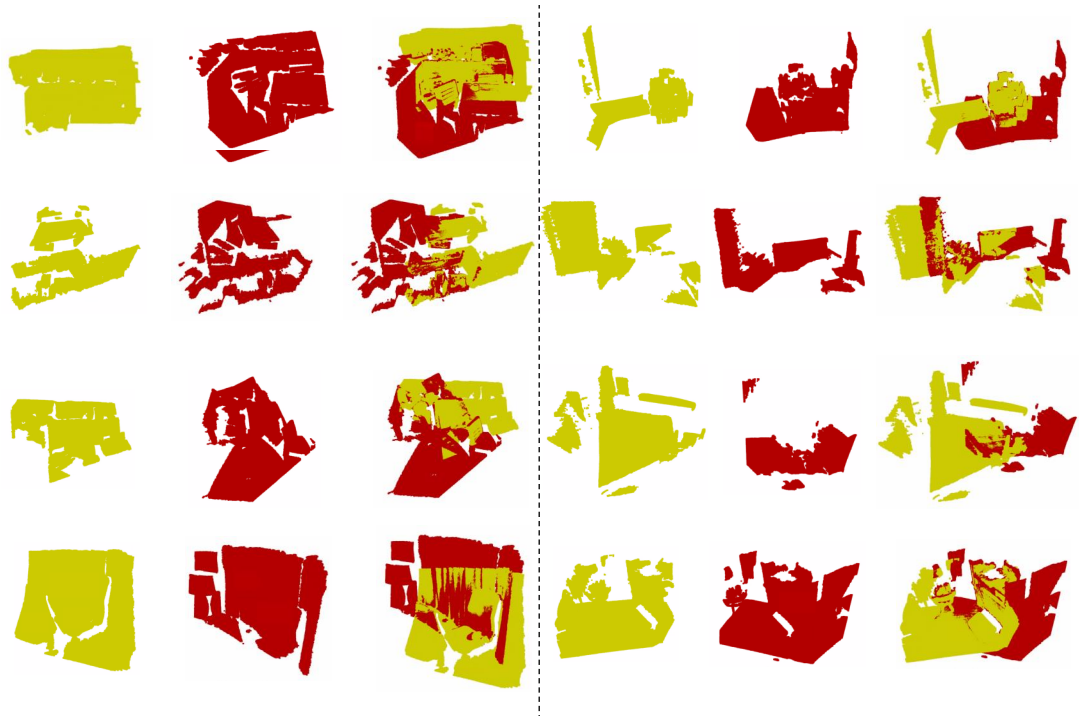


Figure 4.3 : Visualization of our KPSNet applied for point cloud registration. The visualized results are shown in groups of three plots. The first two of each group are two different point clouds to be registered. For clarity, two separate point clouds shown in this figure have been rotated to registered orientations. The third plot shows the registration result.

few variations is challenging. The reason is that keypoints located in regions with complex geometric structure are usually more discriminative than the ones located in regions with less variations in geometry. For example, two different points on the same plane usually have similar features when appearance information is not used, which often leads to the failure of point cloud registration. Our KPSNet can address this challenging scene, such as the forth group of the left sub-figure of Figure 4.3. Of cause, for more easier scenes that have distinct geometric structure, such as the first group in the left sub-figure of Figure 4.3, our proposed KPSNet can achieve perfect point cloud registration results. This observation demonstrates our KPSNet

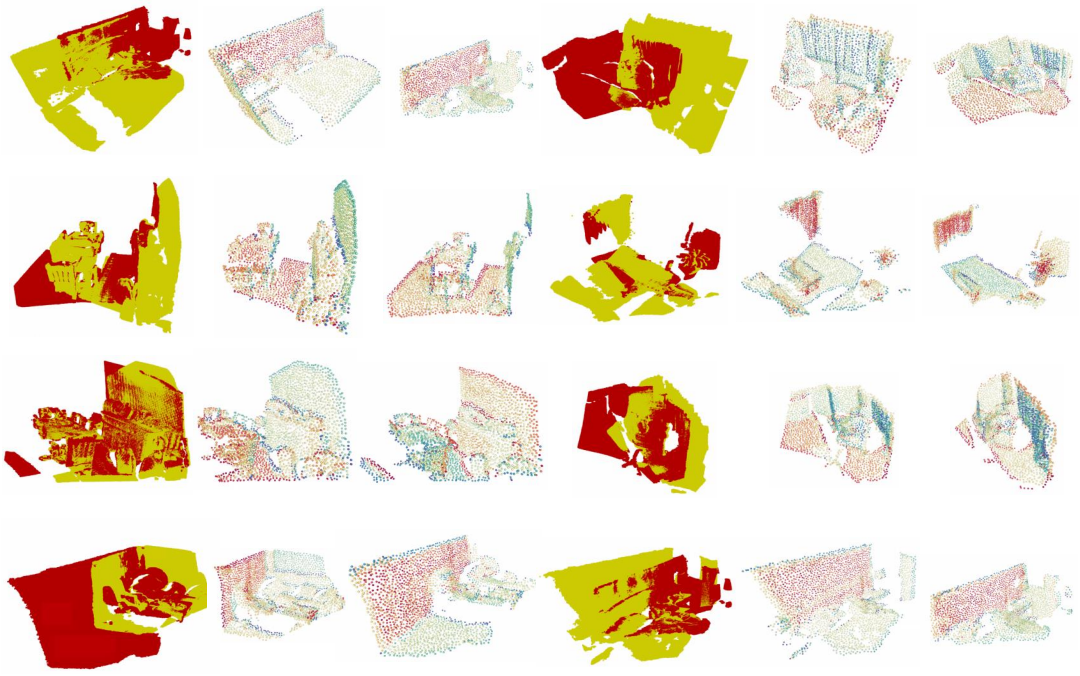


Figure 4.4 : Visualization of keypoint features extracted by our proposed KPSNet. The visualized results are shown in groups of three plots. The first of each group shows two registered point clouds, denoted by yellow and red, respectively, and the last two plots of the group display their candidate keypoints generated by our proposed KPSNet, each point is colored by. The first two columns show two diffident point clouds to be registered. The last columns show the results after registration.

is robust to geometric structure and can generate discriminative keypoints. Overall, our KPSNet can work on challenging scenarios with various overlapping ratios and various variations in geometry.

Visualization of Keypoint Features. To intuitively understand and evaluate our KPSNet, we visualize the keypoint features extracted by our KPSNet in Figure 4.4. Specifically, for each pair of point clouds, we use our KPSNet to generate their keypoint coordinates and keypoint features. And then we make use of t-SNE [100] to project the extracted 64-dimensional keypoint features into one-dimensional

vectors, and encode the one-dimensional vectors to generate keypoints' colors for visualization, same as [16]. Therefore, keypoints rendered with similar colors have similar features. Here, we generate 3000 keypoint candidates for each point cloud and visualize all these keypoints without using our keypoint detector to filter them, in order to show the keypoint locations in the original point clouds more clearly and identify their matching keypoints. Please note that in other experiments, we only generate 1024 keypoint candidates. As shown in Figure 4.4, most of the matched pairs of keypoints have the same or very similar features, and unmatched pairs are different in the feature space. Figure 4.4 shows several challenging scenes. For example, the give point cloud pairs are with various overlapping ratios, and some overlapping regions only contains partial objects or are in very simple geometric structures. This demonstrates the effectiveness of our KPSNet on learning discriminative keypoint features.

Failure Analysis. In Figure 4.5, we show cases with poor point cloud registration results applying our proposed KPSNet. In this part, we delve deeper into these poor or failed cases for the improvement of the follow-up works. Looking at the first two columns of Figure 4.5, for these two cases, the given two pairs of point clouds are almost registered, but their estimated transformation matrices are not accurate enough. We can observe there are offsets between the walls and the floors in the first case and there is a small angle deviation in the second case. For the first case, the corresponding colored plots of their keypoint features show that most matched keypoints are located in planes with similar features, such as the walls, and rare matched keypoints are located in the corner or boundary regions, which are usually with distinctive features. As a result, it is easy to estimate poor transformation matrix by RANSAC. The same problem exists in the second case. In addition, some distinctive keypoints located in the corner are wrongly matched, which results in the errors in rotation angle estimation. The last two columns display two failed

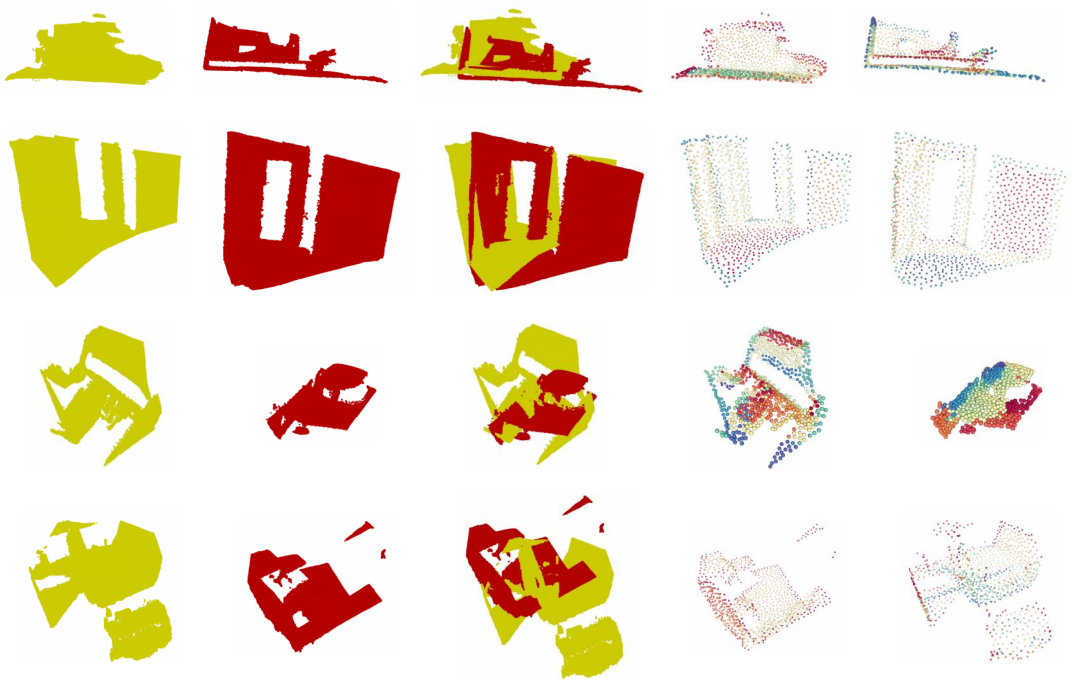


Figure 4.5 : Cases with poor point cloud registration results applying our proposed KPSNet. The first two columns show two different point clouds to be registered. The third column show our point cloud registration results. The last two columns visualize the keypoint features of the two given point clouds, respectively. Keypoint features are colored same as Figure 4.4.

registration cases. From their corresponding plots of keypoint features, we observe many keypoints are wrongly matched, resulting in wrong transformation matrix by RANSAC. Looking at the two failed cases, we observe at least one of the give point clouds have several similar corners within a small spatial range. Only depend on the local regions to extract keypoint features are not enough to identify them. Therefore, the potential direction to improve our method is to explore longer-range spatial contextual information to learn more discriminative keypoint features.

4.4 Conclusions

This chapter introduces a novel and unified KPSNet framework to simultaneously learn a task-specific keypoint detector and feature extractor for point cloud registration. We designed a novel strategy to generate required labels during training by using the indirect ground truth of the point cloud registration task, which avoids the costly manual keypoint annotation. To train our proposed network, we introduced a multitask loss function and jointly optimized the keypoint detector and feature extractor. Experimental results on public datasets show that our approach is rather competitive than other 3D keypoint feature extraction methods for the 3D point cloud registration task. Next step, we further improve the novel methodology by adding more context information and adapting it to outdoor scenes.

Chapter 5

Weakly-supervised Point Cloud Semantic Segmentation with Coarse-grained Labels

5.1 Introduction

Supervised deep learning has achieved impressive results in 3D point cloud segmentation, but it requires a large amount of annotated data [72, 74, 59, 91, 112, 95]. However, designing such fully annotated datasets involves a significant effort in terms of data cleansing and manually labeling, especially the point-level annotations. To reduce the data-labeling cost, researchers have made various efforts recently. Most researchers develop self-supervised learning methods to learn general feature representations, and then use fewer labels to fine-tune the networks of the downstream segmentation task [58, 32, 80, 115, 94]. Although these methods make great successes in object part segmentation (ShapeNet [7]), their effectiveness for real-world semantic segmentation with fewer labels has not been demonstrated. More recently, a few works focus on real-world semantic segmentation with partial labeled points (see Figure 2.5(b)), e.g. 10%, 1% and 1pt (one labeled point per category in each input sample) [119, 127], and achieve impressive results in different scenarios including indoor and outdoor. The above efforts make considerable reductions in data-labeling costs, but the point-level annotating is still inevitable.

To further slash the cost and difficulty of manually annotating, one solution is to develop weakly-supervised methods, which only use easy-to-get coarse labels, e.g. cloud-level labels denoting existing semantic categories, as shown in Figure 2.5(c). The feasibility of applying weakly-supervised methods has been practically proved

in 2D image semantic segmentation field [49, 126, 130], but it has not been well studied in the area of point cloud semantic segmentation. To the best of our knowledge, [109] is the only work to address point cloud semantic segmentation with cloud-level weak labels. It follows the general two-stage pipeline of weakly-supervised methods in 2D image segmentation. Specifically, it first trains a classification network with cloud-level weak labels, from which the Point Class Activation Maps (PCAMs) are extracted for generating point-level pseudo labels. And then, it follows the training style of fully-supervised learning to retrain a more complex semantic segmentation network using the produced pseudo labels. After training, only the semantic segmentation network is applied to predict the final segmentation results.

As a pioneer of applying only cloud-level weak labels for point cloud semantic segmentation, [109] presents a successful attempt. However, it focuses on generating pseudo-labels from cloud-level weak labels, and may not further consider how to make better use of the generated pseudo-labels. The reasons are: **1) noises cannot be eliminated in the generated pseudo labels.** Intuitively, it is impossible to generate perfect pseudo labels automatically in the first stage; otherwise, the second stage is unnecessary. Practically, classification networks trained with cloud-level weak labels can only produce coarse and inaccurate discriminative object localization, which results in noisy pseudo labels, especially in the boundary regions (e.g., regions in red rectangles in Figure 2.5(c)). **2) noisy labels will result in the segmentation network being trained suboptimally and, in turn, degrade the segmentation performance.** For fully-supervised learning methods, the more noises existing in point-level pseudo labels, the worse the model and the poorer the performance will be.

Therefore, in this work, we address this problem from two aspects: 1) How to effectively reduce noises and preserve enough useful information from the generated pseudo labels? 2) How to alleviate the impact of noisy labels during training

semantic segmentation networks?

We present a novel weakly-supervised point cloud semantic segmentation method using coarse cloud-level labels only. In Phase-1, we first train a classification network using the weak labels, generating the point-level pseudo labels. Then, in order to remove the noises in the generated pseudo labels, we add a pseudo label selection module to select the pseudo labels with high confidence for Phase-2. In Phase-2, instead of a simple fully-supervised fashion, we train the semantic segmentation network via a three-branch framework to alleviate the impact of noisy labels. To be specific, a segmentation branch using only the selected pseudo labels alleviates the impact of noisy labels to a certain degree; a Siamese branch exploits the inherent characteristics of point clouds and does not use any labels, which further reduces the impact of noises and improve the robustness and expressiveness of the semantic segmentation network; a multi-label classification branch uses the cloud-level labels to implicitly enhance the learning of the semantic segmentation network.

The proposed weakly-supervised point cloud semantic segmentation method has some appealing characteristics. On the one hand, it learns 3D semantic segmentation by making use of only cloud-level weak supervision. This makes data annotation faster and simpler than strong and partial point-level supervision that requires careful annotation of points on the point cloud. This weakly supervised 3D semantic segmentation approach not only offers an opportunity to reduce the intense need for oversight in the field, but also offers potential commercial benefits. On the other hand, it presents an effective and efficient method to deal with the low-quality pseudo labels, which is compatible with any pseudo-label generating methods and improves the performance of weakly-supervised point cloud semantic segmentation approaches.

Our major achievements of this work are summarized as follows:

- We propose a novel weakly-supervised point cloud semantic segmentation method using coarse cloud-level annotations, which highly reduces the annotating cost.
- We introduce a pseudo label selection module and a three-branch semantic segmentation framework, which can lessen the impact of noises in pseudo labels, as well as improving the robustness and expressiveness of the semantic segmentation network.
- Experimental results on two benchmark datasets indicate that the proposed approach is superior to the advanced weakly-supervised point cloud semantic segmentation approaches. Moreover, our method shows comparable results with some fully-supervised methods, which encourages future research on point cloud segmentation with only coarse cloud-level annotations.

In the rest, we present the overview of the proposed pipeline in Section 5.2, as well as giving the details of Phase-1 and Phase-2. We present our implementation details and experimental results in Section 5.3. At last, we give the conclusions in Section 5.4.

5.2 Pipeline for Weakly-supervised Point Cloud Semantic Segmentation

Our weakly-supervised point cloud semantic segmentation method aims to train a point cloud semantic segmentation model using coarse cloud-level labels. As shown in Figure 5.1, our framework comprises two phases: pseudo label generation with cloud-level weak annotations (Phase-1) and semantic segmentation network learning with pseudo labels (Phase-2). In Phase-1, we firstly generate point-level pseudo labels with the given coarse cloud-level labels, as shown in Figure 5.1 (a). We notice that the pseudo labels involve many unavoidable errors, which can be considered

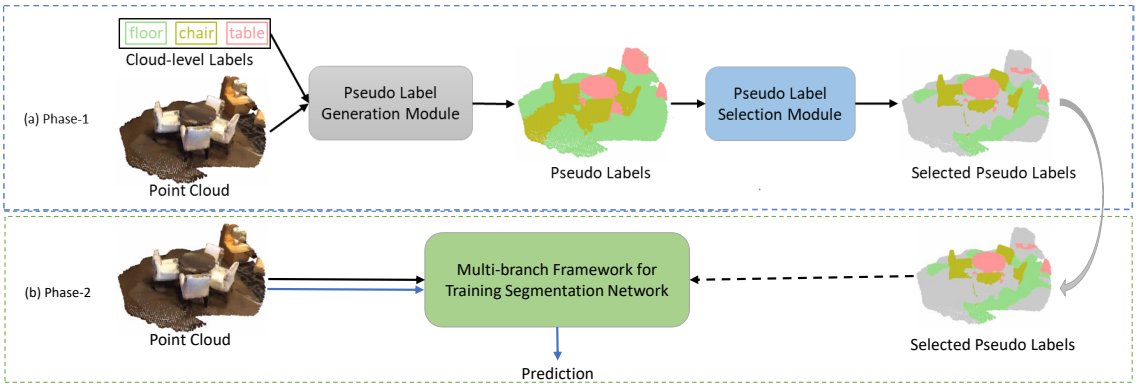


Figure 5.1 : Overview of our weakly-supervised point cloud semantic segmentation method. The black arrows indicate the training flow and the blue arrows indicate the inference flow. During training, (a) Phase-1 generates point-level pseudo labels with cloud-level weak labels by a pseudo label generation module, and then uses the pseudo label selection module to get selected pseudo labels. (Grey color in the Selected Pseudo Labels marks unlabeled points.) (b) Phase-2 retrains a semantic segmentation network via a three-branch framework with the selected pseudo labels. When inference, only the segmentation model learned in Phase-2 is used to produce predictions.

as noisy labels. In order to lessen the impact of the noises existing in the pseudo labels, we introduce the pseudo label selection module, which selects pseudo labels with high confidence. In Phase-2, we develop a three-branch framework to train a semantic segmentation network using the selected pseudo labels, shown in Figure 5.1 (b).

Worthy of mention, the two-stage processing is only applied during training. During inference, given a point cloud, only the semantic segmentation network trained in Phase-2 is utilized, so the inference is efficient. In the next subsections, we will describe the two phases in detail.

5.2.1 Phase-1: Pseudo Label Generation With Cloud-level Weak Annotations

Given coarse cloud-level labels, the goal of Phase-1 is to generate reliable point-level pseudo labels. The used coarse cloud-level labels only annotate the semantic classes that exist in a given point cloud. As shown in Figure 5.1, Phase-1 contains Pseudo Label Generation Module and Pseudo Label Selection Module.

Pseudo Label Generation Module

The pseudo label generation module aims to train a model using weak labels to generate initial point-level pseudo labels. We follow MPRM [109] to build our pseudo label generation module. Since the used weak labels only contain the information of a class existing in a given point cloud or not, MPRM trains a classification network. It uses KPConv [95] as the backbone network, and introduces the Point Class Activation Map (PCAM) to produce localization cues on the input point clouds. Besides, in order to mine more discriminative regions for PCAMs to determine class information, the MPRM module is introduced. To be specific, the MPRM module uses three kinds of attention mechanisms to produce three PCAMs with different discriminative regions. These PCAMs, plus a plain PCAM, are merged and upsampled to form the point-level pseudo labels. The detailed procedure can be found in [109].

Pseudo label selection module

As the pseudo labels are usually of poor quality, they are not suitable to directly as the semantic segmentation results. We need to retrain a semantic segmentation model using the produced pseudo labels, in order to get a better segmentation results. However, the incorrect predictions (also called noises) existing in the pseudo labels make the segmentation network not so efficient. To reduce the impact of the

noises, we introduce a pseudo label selection module.

We define $M_c(p)$ as the upsampled final PCAM of point cloud p for class c, where $c = \{0, ..., C-1\}, C$ is the number of classes. We add a softmax operation on the upsampled final PCAM across all classes, then the feature vector $f_i \in M_c(p)$ of point i is with length C. Each element in f_i is the corresponding class predicted probability, and we take this value to be the confidence of the predicted pseudo label for point i in class c. After gathering statistics of feature vectors, especially the incorrectly predicted ones, we find out that when the point scores of different classes are slightly different, the network is actually confused but still chooses the largest one corresponding to a class label. Therefore, we use a threshold θ to keep highly reliable points (i.e., confidence probability higher than θ) as labeled points, whose labels are generated from the pseudo labels, and consider the rest points as unlabeled points. For convenience, in the implementation, we maintain a mask M for each point cloud to mark the selected points. Without notification, the threshold is frozen in all experiments.

5.2.2 Phase-2: Semantic Segmentation Network Learning With Selected Pseudo Labels

After generation of the pseudo labels and the application of the pseudo label selection module, the points in a point cloud can be grouped into two classes: labeled and unlabeled. Instead of simply training a semantic segmentation network as the same as supervised learning methods with only the selected pseudo labels, we develop a three-branch framework to train the semantic segmentation network. The proposed three-branch framework not only leverages the selected pseudo labels, but also exploits the inherent characteristics of point clouds to assist in learning the semantic segmentation network. As shown in Figure 5.2, our retraining phase contains three branches: **Segmentation Branch**, **Siamese Branch**, **Classification**

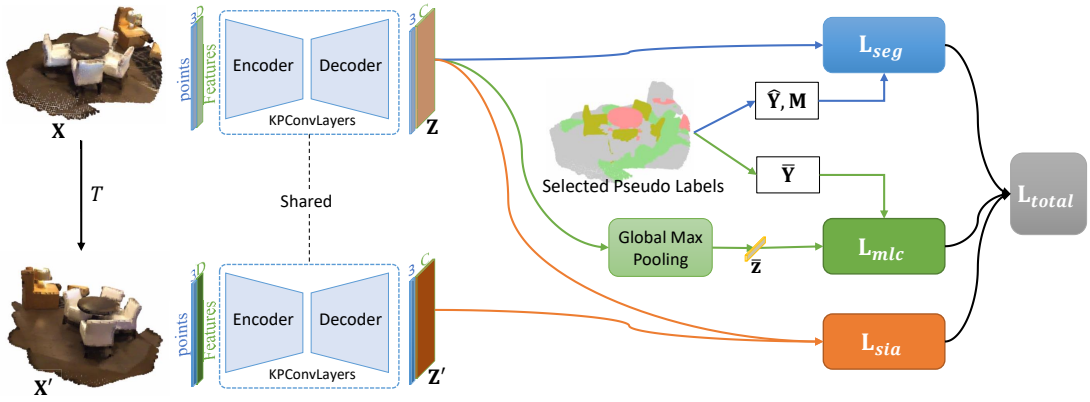


Figure 5.2 : Architecture of the three-branch framework in Phase-2. The segmentation network is an encoder-decoder architecture. The colored arrows (orange, green, blue) demonstrate the proposed three branches, Segmentation, Siamese and Classification Branch respectively. M is a mask denoting whether a point is labeled or not.

Branch.

Segmentation Branch. The segmentation branch receives a point cloud as input and output its semantic segmentation predictions. It is similar to the fully-supervised semantic segmentation task, but only applies to labeled points. Specifically, we denote an input sample as $\mathbf{X} \in \mathcal{R}^{N\times(3+D)}$, where N is the number of points, 3 denotes the XYZ coordinates and D is defined as the feature dimension of $\mathbf{X}$. A point cloud is embedded to obtain its features $\mathbf{Z} \in \mathcal{R}^{N\times C}$ by a semantic segmentation network, where C represents the number of segmentation classes. In this work, the segmentation network is implemented by the KPConvDeform [95] network. We denote $\hat{\mathbf{Y}} \in \mathcal{R}^{N\times C}$ as the one-hot version of the point-level pseudo labels of $\mathbf{X}$, where the i-th element $\hat{\mathbf{y}}_i \in \mathcal{R}^C$ is a on-hot vector that represents the semantic label of point i. We use a binary mask $\mathbf{M} \in \{0,1\}^N$ to represent if a point is labeled or not, i.e., the i-th element m_i is 1 for a labeled point and 0 otherwise. With the generated pseudo labels and the pseudo label selection module, we can

fulfill the per-point segmentation label $\hat{\mathbf{Y}}$ and mask $\mathbf{M}$ for all training point cloud samples. Therefore, the segmentation loss for the labeled points is defined as:

$$L_{seg} = -\frac{1}{K}\sum_{b}^{B}\sum_{i}^{N} m_{bi}\sum_{c}^{C}\hat{y}_{bic}\log\frac{\exp\left(z_{bic}\right)}{\sum_{c}\exp\left(z_{bic}\right)}, \tag{5.1}$$

where N is the number of points in a point cloud, B denotes the batch size, $K = \sum_{b,i} m_{bi} = \|\mathbf{M}\|_1$ is the normalization variable.

Reasonability Analysis: It is demonstrated that training a segmentation network with an adequate amount of labeled points, i.e., sufficiently large $\|\mathbf{M}\|_1$, can yield close results with its fully-supervised variation [119]. For example, [119] achieves competitive results using only 10% labeled points. To study the reasonability of our retraining phase with selected pseudo labels, we evaluate the point-level pseudo labels under the selection threshold, see Table 5.5. With the threshold used in our experiments, the proportions of correctly labeled points of all classes are more than 20%, and the mean proportion of correct labeled points is 37.2%, which is sufficiently large compared with [119]. We can reasonably infer that the upper bound of the performance from our segmentation network can be achieved by training a segmentation network with about 20% manually labeled points. However, the incorrect labels degrade the performance heavily. To alleviate the impact and avoid introducing disadvantageous factors, we should start from the data itself to exploit its characteristics without using pseudo labels. Receiving inspiration from self-supervised representation learning, we introduce our Siamese branch.

Siamese Branch. The Siamese branch aims to learn good point feature representations in a self-supervised fashion, which exploits the inherent characteristics of the training data and does not introduce labeling noise. For 3D point clouds, one of the most important characteristics of good feature representations is transformation invariance. Therefore, for a robust point cloud semantic segmentation network, the prediction for any point should be invariant under different views. While getting

the point cloud in different views is equal to applying specific transformations (e.g., rotation, flipping) on the point cloud. Hence, if we employ these transformations to a given point cloud, the predicted semantic labels of the initial point cloud and the transformed one should be the same. With this assumption, we introduce the Siamese branch.

As exhibited in Figure 5.2, we firstly apply a random transformation T on a given point cloud $\mathbf{X}$, and form a paired point cloud samples $\{\mathbf{X}, \mathbf{X}'\}$. The Siamese branch takes $\{\mathbf{X}, \mathbf{X}'\}$ as input, passes forward through two shared-parameter networks Φ followed by a softmax function g, and generates a paired point-wise predictions $\{\mathbf{Z}, \mathbf{Z}'\}$. The Siamese loss is designed to force the paired predictions to be consistent. In other words, we aim to minimize the divergence of the paired probabilistic predictions. In this paper, we choose L2 distance to estimate the consistency for simplification:

$$L_{sia} = \frac{1}{B \cdot N \cdot C} \sum_{b}^{B} ||g(\mathbf{Z}_b) - g(\mathbf{Z'}_b)||^2, \tag{5.2}$$

where N defines the number of points in a point cloud, B is the batch size, C represents the number of classes.

Analysis: Our Siamese branch has two advantages. On the one hand, it explores more information about the points without labels, which assists in learning better feature representations, and improve the robustness and expressiveness of the semantic segmentation network. On the other hand, it weakens the impact of the noisy pseudo labels in the segmentation branch implicitly.

Classification Branch. Our classification branch forms a multi-label classification task, which aims to generate the class predictions for a given point cloud. The classification branch is trained using the class labels of input point clouds, which derive from the point-level pseudo labels in Phase-1. Specifically, we get the class labels by doing a max-pooling operation to all points: $\bar{\mathbf{y}} = \max_i \hat{\mathbf{y}}_i$. We apply a global

max-pooling layer on the feature embeddings $\mathbf{Z}$ to get the class logits, denoting as $\bar{\mathbf{z}}$. The loss of the classification branch is:

$$L_{mlc} = -\frac{1}{B \cdot C} \sum_{b}^{B} \sum_{c}^{C} \bar{y}_{bc} \log \frac{1}{1 + \exp\left(-\bar{z}_{bc}\right)} + \left(1 - \bar{y}_{bc}\right) \left(\log \left(\frac{\exp\left(-\bar{z}_{bc}\right)}{1 + \exp\left(-\bar{z}_{bc}\right)} \right) \right), \tag{5.3}$$

Analysis: If a class does not exist in a point cloud, the classification branch will optimize the network so that no points are predicted as that class. As this classification branch is applied on both labeled and unlabeled points while the segmentation branch is only applied on labeled points, it explores more information than the segmentation branch. Therefore, these two branches complement each other and promote learning a more expressive network.

The final training objective of Phase-2 is obtained by combining all the three above losses:

$$L_{total} = L_{seg} + \lambda_1 L_{sia} + \lambda_2 L_{mlc}, \tag{5.4}$$

where λ_1 and λ_2 are loss scalars.

5.3 Experimental Setup and Results

5.3.1 Datasets and Experimental Settings

Datasets and Weak Labels Preparation

To assess the performance of the introduced framework, we carry out comprehensive experiments on ScanNet and S3DISs.

ScanNet [17] is a large-scale indoor dataset. It was collected from over 707 unique indoor environments. There are 1513 scenes with fully per-point annotations opening to the public and 100 scenes without any annotations for online testing. To get the performance on the testing dataset, researchers must submit their test

results to an online server (`http://kaldir.vc.in.tum.de/scannet_benchmark/`). Each scene is represented as a point cloud, and each point are provided with the XYZ coordinates and RGB color. This dataset provides 40 class labels, but only selected 20 classes are used for segmentation evaluation, including walls, floors, cabinets, beds, chairs, sofas, tables, doors, windows, bookshelves, pictures, counters, desks, curtains, refrigerators, shower curtains, toilets, sinks, bathtubs and other furniture. The rest classes are treated as unclassified and excluded from the evaluation. Therefore, there are many noises (unclassified points) in the ScanNet, as shown in Figure 5.3. In our experiments, we use the official train-val partition to further separate the 1513 point clouds, with 1201 point clouds as the training set and 312 point clouds as the validation set.

We prepare our training samples with weak labels as follows: Before training, we use a grid subsampling strategy to pre-process all point clouds with grid size $dl_0 = 0.04m$ following the data pre-processing of [95]. During the training stage, the networks we used receive spherical samples from each point cloud as input. Therefore, we follow [109] to prepare the training samples and their weak labels for Phase-1. Specifically, we uniformly sample each training point cloud into several spherical samples with radius $r = 2.0m$ and sampling interval $dl_1 = 2.0m$ along each dimension. We ignore the small spherical samples that contain less than 200 points for efficiency. Finally, the resulting average number of training samples in each scene is about 18.4. The weak label of each spherical sample is defined as a 20-dimensional vector indicating the classes existing in the spherical sample. So each scene only needs about 18 label vectors to be manually annotated, which costs quite lower than full point-level annotations required by fully-supervised segmentation methods. For Phase-2, we randomly sample spherical samples with radius $r = 2.0m$ from all training point clouds as our training samples. No manual annotations are required in Phase-2. We use the RGB color plus the constant 1 to form 4-dimensional vectors

as the input point features for both Phase-1 and Phase-2.

S3DIS [2] is another large-scale indoor dataset. It contains 271 point clouds with 13 semantic classes. Each point cloud is provided with XYZ coordinates and RGB values. All point clouds are distributed in six areas. In our experiments, we adopt results on Area-5 for evaluation, same as [92, 95]. Because the building where Area-5 comes from does not contain other areas, which is more fair to evaluate the generalizability performance of approaches. In total, there are 204 training point clouds and 68 testing point clouds. Please refer to the Section 3.3.1 for detailed description of the S3DIS.

For the experiments carried on S3DIS, our training samples associated with their weak labels are prepared as below. Same as the data pre-processing for the ScanNet, we also use a grid subsampling strategy to pre-process each point cloud with grid size $dl_0 = 0.04m$. During the training stage, we uniformly sample each training point cloud into several spherical samples with radius $r = 2.0m$ and sampling interval $dl_1 = 1.0m$ along each dimension for Phase-1. We also ignore the small samples that contain less than 200 points. Overall, the resulting average number of training samples is about 17.6. Each spherical sample is manually annotated and represented by a 13-dimensional vector indicating the classes existing in it. For Phase-2, we randomly sample spherical samples with radius $r = 2.0m$ from all training point clouds as our training samples. No manual annotations are required in Phase-2.

Evaluation Metrics. We use the following evaluation metrics in our experiments: IoU of every semantic class and the mean of IoUs over all classes (mIoU). We describe the detailed computations of IoU in Section 3.3.1.

Implementation Detail

Detailed Network Architecture. We apply KPConv [95] to implement the networks for both Phase-1 and Phase-2. In Phase-1, we follow MPRM [109] to

build the classification network for the pseudo label generation module. In Phase-2, we build our segmentation network according to KP-FCNN provided by [95]. The encoder consists of four blocks: two normal ResNet bottleneck blocks and two deformable ResNet bottleneck blocks. Each block downsamples the points by about half. The decoder is composed of four nearest upsampling layers and there is a unary convolution following each upsampling layer. The intermediate representations from the encoder are concatenated to the corresponding upsampled ones from the decoder through Skip links. Following the decoder, there is a classifier to predict the point-wise segmentation results, which is made of a unary convolutional layer and a softmax layer.

Training Configuration. During training, we use a Momentum SGB optimizer, and set the momentum as 0.98. We set 0.01 as our initial learning rate, and decay it by 10 every 100 epochs. Our batch size for training is set to 10. For Phase-1, the classification network is trained for 400 epochs. For Phase-2, our three-branches network is trained for 150 epochs. The loss scalars are set $\lambda_1 = \lambda_2 = 1$.

5.3.2 Performance on ScanNet

Quantitative Results on ScanNet Validation Set. In Table 5.1, we report our quantitative segmentation results on the ScanNet validation set. The baseline model is a standard point cloud semantic segmentation model having the same encoder and decoder as our semantic segmentation network, trained in a fully-supervised fashion using all initial pseudo labels. The difference between our baseline model and the MPRM model [109] is that MPRM adds a CRF post-processing step, which can refine the segmentation results and improve the performance. We can note that our model (Ours) is superior to the baseline model by increasing 2.8% mIoU. As for the class-specific IoUs, our model increases the performance on almost all classes. This demonstrates our method can reduce the impact of the noises

Table 5.1 : Segmentation results on ScanNet validation set. We provide the mIoU (%) and per-class IoUs (%). Bold represents the best results across all listed approaches.

Setting	mIoU	wall	floor	cabinet	bed	chair	sofa	table	door	wind.	book.	pic.	count.	desk	cur.	ref.	show.	toilet	sink	bath.	other.
MPRM [109]	43.2	59.4	59.6	25.1	64.1	55.7	58.7	45.6	36.4	**40.3**	67.0	16.1	22.6	**42.9**	66.9	24.1	39.6	**47.0**	**21.2**	44.7	28.0
Baseline	42.1	57.1	61.1	23.6	65.9	54.0	56.8	47.4	33.5	36.2	59.7	**17.7**	23.4	37.2	62.0	24.4	44.7	46.3	20.6	**46.6**	23.7
Ours	**44.9**	**59.4**	**64.8**	**31.4**	**67.3**	**57.1**	**58.7**	**48.6**	**37.5**	39.3	**68.5**	17.6	**25.3**	40.6	**67.5**	**24.5**	**49.3**	42.9	21.0	45.6	**30.2**

existing in the pseudo labels, and therefore learns a better segmentation network. The reason is the initial pseudo labels includes many incorrect predictions, which degrades the performance of a fully-supervised segmentation network. The results also demonstrate that a well-designed retraining method can get more benefit from the noisy pseudo labels than pure retraining in weakly-supervised point cloud semantic segmentation methods.

MPRM [109] reports the performance of the baseline with a CRF post-processing step, which refines the segmentation results by regularization using low-level features, such as the pairwise smoothness constrains on color and geometric space. We can observe that our model still outperforms MPRM by 1.7% mIoU without any post-processing steps. And on several classes such as floor, table, curtain (cur.) and shower curtain (show.), our model outperforms MPRM by large margins.

Qualitative Results on ScanNet Validation Set. Figure 5.3 depicts some visual results by choosing a few representative scenes from ScanNet validation set. As we can see, our model has the capability to gain visually high-quality segmentation results of objects existing in a simple environment and have approximate complete point clouds, such as the chairs, tables and floor in the first row. In addition, our model can give precise predictions of objects that are in complex environments but have approximate complete point clouds, such as the tables and the curtains in the

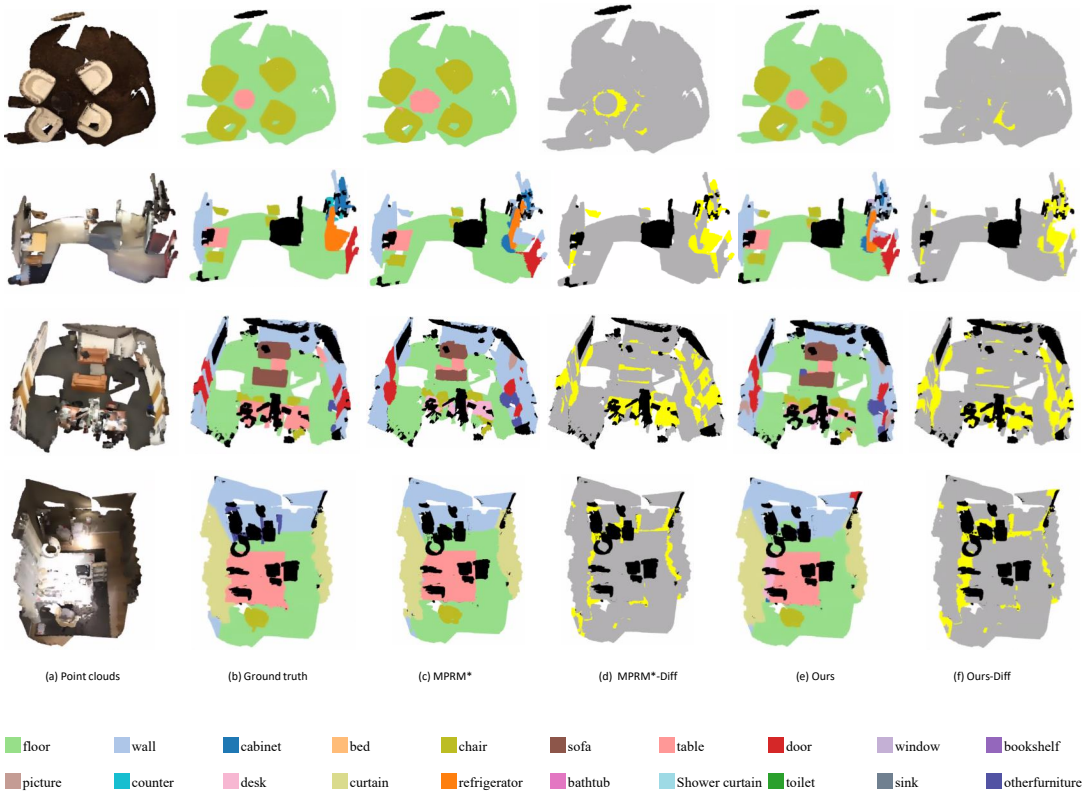


Figure 5.3 : Visualization of our segmentation results on ScanNet validation set. From left to right: (a) input point clouds, (b) ground truths, (c) predictions of MPRM* [109], (d) differences between ground truths and (c), (e) predictions of our proposed method, (f) differences between ground truths and (e). Yellow points in columns (d) and (f) denote incorrect predictions. Black points are unclassified points. MPRM* denotes our reproduced results.

second row, and the sofas in the third row.

For objects that are highly occluded and the object parts that are right next to others, our model produces poor predictions. Note that it is hard to learn good representations for these objects, even in a fully-supervised fashion; for another thing, the reliable pseudo labels of those objects or parts are less due to their initial poor pseudo labels and our pseudo label selection module, which exacerbates the difficulties in their segmentation.

Table 5.2 : Comparisons on ScanNet test set. We compare the fully-supervised (Ful.Sup.) and weakly-supervised (Weak.Sup.) methods separately. The results are from online benchmark. We provide the mIoU (%) and per-class IoUs (%). Bold fonts represent the best result in weakly-supervised methods, and underlined fonts represent the best results in fully-supervised methods.

Setting	Methods	mIoU	wall	floor	cabinet	bed	chair	sofa	table	door	wind.	book.	pic.	count.	desk	cur.	ref.	show.	toilet	sink	bath.	other.
Ful. Sup. (100%)	PointNet++ [74]	33.9	52.3	67.7	25.6	47.8	36.0	34.6	23.2	26.1	25.2	45.8	11.7	25.0	27.8	24.7	21.2	14.5	54.8	36.4	58.4	18.3
	TangentConv [91]	43.8	63.3	91.8	36.9	64.6	64.5	56.2	42.7	27.9	35.2	47.4	14.7	35.3	28.2	25.8	28.3	29.4	61.9	48.7	43.7	29.8
	PointConv [112]	66.6	81.3	95.3	64.4	75.9	82.2	75.3	58.8	50.4	64.2	69.9	20.3	47.5	56.4	77.9	58.6	75.4	90.2	66.1	78.1	42.8
	KPConv [95]	68.4	81.9	93.5	64.7	75.8	81.4	78.5	61.4	59.4	63.2	78.4	18.1	47.3	60.5	77.2	58.7	80.5	88.2	69.0	84.7	45.0
Weak. Sup.	MPRM [109]	41.1	**61.6**	62.1	28.5	**65.0**	**51.9**	**59.4**	36.9	38.6	30.6	47.5	11.7	**8.7**	**39.6**	**72.5**	33.8	44.3	**37.7**	**18.8**	47.9	25.0
	Ours	**42.5**	60.2	**63.6**	**32.4**	64.7	48.8	55.1	**39.8**	**40.1**	**36.1**	**52.2**	**17.6**	7.7	35.3	71.2	**34.0**	**56.5**	37.0	17.5	**52.5**	**28.1**

Quantitative results on ScanNet testing set. Our method is also evaluated on the public ScanNet testing set. We submit our predictions to the online official evaluation server to get the performance and report the segmentation results in Table 5.2. We compare our method with five leading approaches listed in Table 5.2, including one existing weakly-supervised point cloud semantic segmentation approach [109] and four fully-supervised approaches [74, 91, 112, 95]. Comparing our method with MPRM [109], we observe our model outperforms MPRM by 1.4% mIoU, and on several categories like window (wind.), picture (pic.) and shower curtain (show.), our model surpass MPRM by a significant margin, e.g., 5.5% on window, 5.9% on picture and 12.2% on shower curtain. Although our model performs not better than MPRM on several categories, the gap is small, e.g., 1.4% on wall, 0.3% on bed. Moreover, our results are without any post-processing steps. We also notice that, compared with MPRM, our model decreases the performance on three categories by 3.1% on chair, 4.3% on sofa and 4.3% on desk, respectively. This may result from the over-filtering of the pseudo labels of those categories by

the pseudo label selection module. As shown in Table 5.5, the initial pseudo labels (Phase1) of these three categories used by MPRM are relatively good, achieving the accuracy of 60.2% for chair, 77.4% for sofa and 72.0% for desk, respectively. However, after applying the pseudo label selection module, the selected pseudo labels (Our-sel) our model used cover correct labels of 50.8% for chair, 68.1% for sofa and 45.4% for desk, in terms of the whole training data. The useful supervision may be over-filtered for some good-performed categories, especially the desk. This phenomenon also inspires us to further improve the weakly-supervised point cloud semantic segmentation framework by designing more complex and adaptive pseudo label selection modules in future work.

KPConv [95] and PointConv [112] are the current state-of-the-art fully-supervised approaches. Although there is a performance gap between our method and the best fully-supervised approaches, the gap in several categories is small, such as bed, picture and curtain. Moreover, with only cloud-level weak labels, our model achieves better performance than PointNet++ [74], and narrows the gap with other advanced fully-supervised point cloud semantic segmentation approaches.

5.3.3 Performance on S3DIS

Quantitative Results on S3DIS. Our proposed weakly-supervised method is evaluated on S3DIS Area-5. As shown in Table 5.3, our approach is compared with the current advanced fully-supervised point cloud semantic segmentation approaches and semi-/weakly-supervised point cloud semantic segmentation approaches.

We implement the MPRM [109] by the official code as our baseline model, shown as MPRM* in Table 5.3. Comparing our model with the baseline model, we observe that our model (Ours) outperforms the baseline by 0.5% mIoU, and our model consistently increases the IoUs of most categories except for column and board. We also notice that the performance improvement on S3DIS is incremental compared

Table 5.3 : Comparison results on S3DIS testing set (Area-5). We group methods based on their required supervision types: fully-supervised (Ful.Sup.), semi-supervised (Semi.) and weakly-supervised (Weak.Sup.). We provide the mIoU (%) and per-class IoUs (%). The results are from publications. Bold fonts represent the best result in semi-/weakly-supervised methods, and underlined fonts represent the best results in fully-supervised methods.

Setting	Methods	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
Ful.Sup. (100%)	PointNet++ [74]	47.8	90.3	95.6	69.3	0.1	13.8	26.7	44.1	64.3	70.0	27.8	47.8	30.8	38.1
	DGCNN [108]	47.0	92.4	97.6	74.5	0.5	13.3	48.0	23.7	65.4	67.0	10.7	44.0	34.2	40.0
	TangentConv [91]	52.8	-	-	-	-	-	-	-	-	-	-	-	-	-
	PointWeb [129]	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 [95]	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	91.0	81.5	75.3	75.4	66.7	58.9
Semi.(10%)	Xu [119]	48.0	**90.9**	**97.3**	**74.8**	0.0	8.4	49.3	27.3	69.0	**71.7**	16.5	53.2	23.3	**42.8**
Semi.(1pt)	Xu [119]	44.5	90.1	97.1	71.9	0.0	1.9	47.2	29.3	62.9	64.0	15.9	42.2	18.9	37.5
Weak.	MPRM*	47.7	68.5	72.5	48.6	0.0	**14.9**	49.8	39.4	74.6	49.8	63.3	67.4	**44.6**	27.0
Sup.	**Ours**	**48.2**	69.7	73.1	48.7	0.0	14.4	**50.0**	**39.7**	**75.0**	50.7	**63.6**	**70.5**	43.9	27.4

with that on ScanNet. This is because objects in S3DIS are almost intact and contain less noises, which results in more reliable pseudo labels being generated and more selected pseudo labels being preserved. Therefore, the segmentation branch with the pseudo labels dominates the learning of our semantic segmentation network, which results in the incremental improvement of segmentation results on S3DIS.

Our method is also compared with the Xu [119], which addresses the point cloud semantic segmentation task without full supervision and reports the results on S3DIS. We should notice that Xu [119] uses both cloud-level weak labels and additional partial point-level labels. Specifically, except for the cloud-level weak labels, Xu [119](1pt) requires 1 point within each category to be labeled for each

training sample (less than 0.2% of all points), and Xu [119](10%) requires 10% of all points labeled for each input point clouds. From the aspect of labeling cost, our model (Ours) is approximate to the Xu [119](1pt) model, and much less than the Xu [119](10%) model. From the aspect of segmentation performance, our model (Ours) outperforms Xu [119](1pt) by a significant margin (3.7% mIoU), and achieves comparable overall performance (i.e., 48.2% mIoU) with Xu [119](10%).

We compare our weakly-supervised point cloud semantic segmentation approach with some popular fully-supervised point cloud semantic segmentation approaches

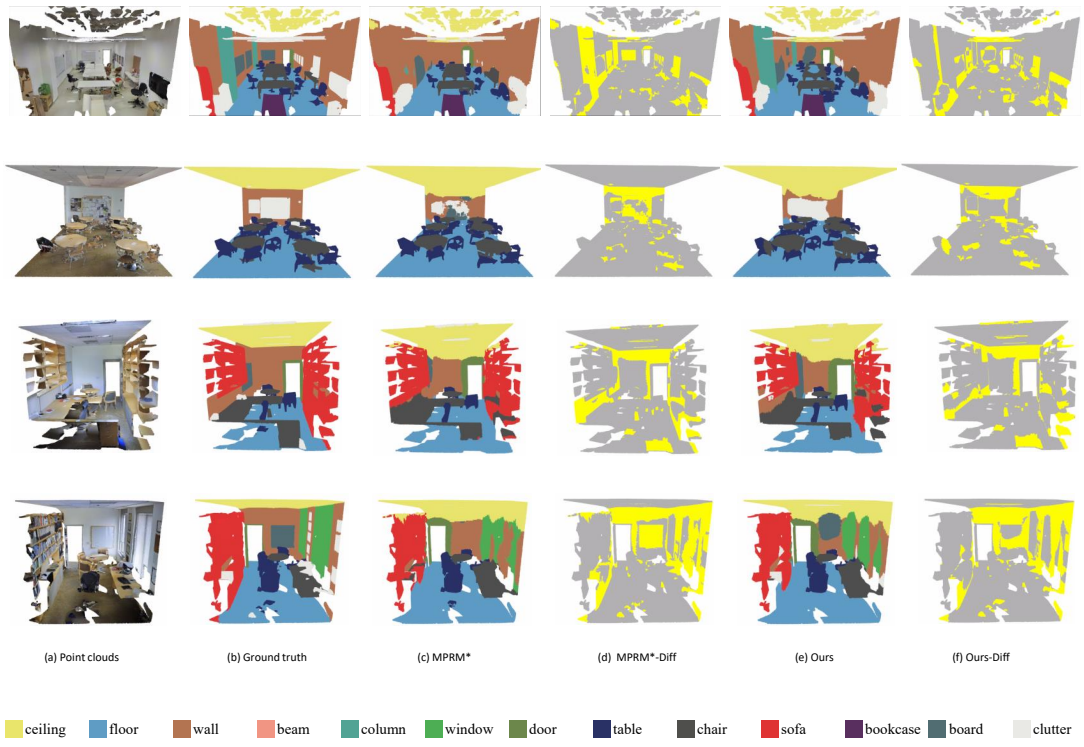


Figure 5.4 : Visualization of our segmentation results on S3DIS testing set. From left to right: (a) input point clouds, (b) ground truths, (c) predictions of MPRM* [109], (d) differences between ground truths and (c), (e) predictions of our proposed method, (f) differences between ground truths and (e). Yellow points in columns (d) and (f) denote incorrect predictions. MPRM* denotes our reproduced results.

[74, 108, 91, 129, 95]. We observe that our method outperforms PointNet++ [74] and DGCNN [108]. Meanwhile, a certain gap still exists between our proposed weakly-supervised point cloud semantic segmentation approach and the advanced fully-supervised point cloud semantic segmentation approaches like KPConv [95]. However, the performance gap on several categories is small, such as the window, bookcase and sofa.

Qualitative Results on S3DIS. We visualize our segmentation results on S3DIS testing dataset in Figure 5.4. We can observe our proposed weakly-supervised method obtains quite good segmentation results in several scenes, like the conference room and office. From the column (d) of Figure 5.4, we can easily observe our model is able to produce precise segmentation predictions of the central part of objects, and the incorrect predictions are mostly located in the boundary regions of objects.

5.3.4 Ablation Study

Necessity and effectiveness of our pseudo label selection module. As we analyzed in Section 5.2, the incorrect pseudo labels limit the performance of the retraining segmentation model. Moreover, training a segmentation network with a sufficient number of labeled points, e.g., 10% labeled points, is able to yield approximate results with its fully-supervised counterpart. Therefore, in this section, we compare the performance of the pseudo labels with/without our selection strategy.

Table 5.4 : Segmentation performance of pseudo labels and selected version on ScanNet training set. We report the mIoU (%) and per-class IoUs (%).

Setting	mIoU	wall	floor	cabinet	bed	chair	sofa	table	door	wind.	book.	pic.	count.	desk	cur.	ref.	show.	toilet	sink	bath.	other.
Phase1	46.3	54.4	57.9	31.6	68.1	49.1	67.4	48.3	39.5	41.8	66.1	26.5	23.7	44.9	71.1	35.6	53.1	42.4	18.6	52.1	33.2
Phase1-CRF	47.4	56.3	60.5	33.1	71.1	50.2	69.2	49.6	40.4	42.6	68.8	26.5	24.2	45.7	71.7	36.9	54.6	41.4	18.4	51.5	34.1
Our-sel	**56.4**	**62.2**	**67.3**	**40.4**	**80.4**	**65.6**	**77.4**	**62.9**	**53.3**	**52.2**	**81.6**	**31.7**	**26.1**	**57.6**	**82.1**	**49.9**	**65.2**	**48.4**	**21.5**	**59.1**	**43.1**

Table 5.5 : Comprehensive evaluation of the pseudo labels on ScanNet training set.

Accuray(%) of labeled points																				
Setting	OA	wall	floor	cabinet	bed	chair	sofa	table	door	wind.	book.	pic.	count.	desk	cur.	ref.	show.	toilet	sink	bath.	other.
Phase1	60.6	74.1	71.0	61.1	78.5	60.2	77.4	62.9	59.6	59.2	91.2	31.1	37.2	72.0	83.6	60.5	62.1	43.0	18.9	57.2	50.2
Our-sel	67.3	85.7	80.0	70.3	85.4	76.0	83.2	72.8	68.1	64.2	93.5	34.3	43.9	79.0	87.7	64.6	70.6	48.7	21.6	61.7	54.8
The proportion (%) of labeled points in whole points																				
Setting	mean	wall	floor	cabinet	bed	chair	sofa	table	door	wind.	book.	pic.	count.	desk	cur.	ref.	show.	toilet	sink	bath.	other.
Our-sel	55.2	52.5	46.6	49.3	79.0	66.8	81.9	60.4	45.3	58.9	66.2	69.8	63.8	57.4	81.3	49.0	76.5	96.2	93.1	81.7	53.4
The proportion (%) of correct labeled points in whole points																				
Setting	mean	wall	floor	cabinet	bed	chair	sofa	table	door	wind.	book.	pic.	count.	desk	cur.	ref.	show.	toilet	sink	bath.	other.
Our-sel	37.2	45.0	37.2	34.7	67.5	50.8	68.1	44.0	30.8	37.8	61.9	23.9	28.0	45.4	71.3	31.6	54.0	46.8	20.1	50.4	29.3

In Table 5.4, we show the mIoU and the per-class IoUs of the initial pseudo labels (Phase1) and our selected pseudo labels (Our-sel) on ScanNet training set with default threshold $\theta = 0.95$. Similar to [109], we also incorporate the CRF post-processing step (Phase1-CRF) to improve the segmentation performance by leveraging the low-level feature, and we take the results of Phase1-CRF as the initial pseudo labels in our experiments. We observe that our selection process increases the IoUs of the initial pseudo labels by a large margin on almost all classes, and the mIoU across all classes increased by 9%. In Table 5.5, we show the more comprehensive evaluation of our selected pseudo labels, consists of the overall accuracy (OA) and the per-class accuracy, the proportion of labeled points in whole points and the proportion of correct labeled points in whole points. We observe that the OA increases from 60.6% to 67.3%, and the accuracy of most classes increase by a large margin. From the increased IoUs and accuracy, we can infer that our selection strategy can filter incorrect pseudo labels, and as a result, it can reduce the impact of incorrect labels during the retraining stage. Moreover, for all classes, the remaining labeled points still constitute more than 45.3% of whole points in training data, and the

correct labeled points constitute more than 20% of whole points in training data. According to the previous reasonability analysis that training a segmentation network with a sufficient number of labeled points is able to yield approximate results with its fully-supervised variation, we can infer that the selected labels are sufficient for retraining the segmentation network. In sum, our selection strategy can filter incorrect pseudo labels with remaining sufficient supervision information. Therefore, it has the potential to increasing the performance of the retraining segmentation model.

Effect of the threshold of pseudo label selection.

As suggested by the previous study, both the number of labeled points and noises existing in the pseudo labels significantly affect the segmentation performance of Phase-2. Based on our pseudo label selection strategy, a higher threshold θ will remove more noises and preserve fewer point-level pseudo labels, while a lower threshold θ will preserve more noises. We show the segmentation results using segmentation loss only on ScanNet validation set with different thresholds in Table 5.6. We can observe that the mIoU gradually increases as the threshold increases from 0.80 to 0.95, but drops a lot when the threshold is greater than 0.95. The reasons may be that: 1) when the threshold is between 0.8 and 0.95, sufficient pseudo labels are preserved and more noisy labels are removed with the increasing threshold, resulting increasing mIoU. 2) when the threshold is greater than 0.95, very few pseudo

Table 5.6 : Performance of our method with different thresholds θ on ScanNet validation set. In order to evaluate the effect of the different thresholds, all models are trained using segmentation loss only.

θ	0.99	0.95	0.90	0.85	0.80
mIoU(%)	41.4	**42.9**	42.7	42.4	42.2

labels are preserved and insufficient supervision results in worse performance. When $\theta = 0.95$, the model achieves the highest mIoU 42.9%. Therefore, we use $\theta = 0.95$ for all experiments.

Effect of different branches. In this subsection, the importance of individual components of our three-branch retraining method in Phase-2 is also analyzed. We evaluate the different combinations of the introduced losses on ScanNet validation set in Table 5.7. We observe that adding the Siamese loss improves the segmentation performance by 1.3%, and the performance is further increased by the classification branch.

Table 5.7 : Impact of individual losses on ScanNet validation set.

seg	sia	mlc	mIoU(%)
✓			42.9
✓	✓		44.2
✓	✓	✓	**44.9**

Performance upper bound of our proposed method. As mentioned in Section 5.2, the upper bound of performance of our proposed framework is training a fully-supervised segmentation network with the manually annotated point-level labels. To analyze the gap and potentials of our method, we discuss our method with its upper bound counterpart (KPConv [95]) in this subsection.

Table 5.3 indicates that there still exists a certain gap between the fully-supervised segmentation method using point-level ground truth (KPConvDeform [95]) and our weakly-supervised method. Intuitively, the performance of our method is due to two aspects: the pseudo labels and the retraining method. We observe that for specific classes that are more discriminative, the Phase1 model gains a small gap to the fully-supervised KPConvDeform model or even better performance, such as

the beam, column, window, bookcase, sofa and board. For the rest classes, the gap between Phase-1 and KPConvDeform is bigger. This phenomenon demonstrates the classification network in Phase-1 can capture better discriminative features. On classes that exist in training samples in high frequency, like floors, walls, ceilings, and chairs, the supervised segmentation network tends to achieve better performance, and our retraining method is as well. We can observe that our final segmentation model (Ours) further reduces the gap of classes like ceilings, floors, walls, chairs by a large margin.

5.4 Conclusions

This chapter presents a new weakly-supervised point cloud semantic segmentation method using coarse cloud-level annotations. With the help of the proposed pseudo label selection module and three-branch retraining framework, our method can alleviate the ill-impact of the noisy pseudo labels and improve the robustness and expressiveness of the semantic segmentation network, and in turn, improve the final segmentation performance. Extensive experimental results demonstrate that our method outperforms other weakly-supervised semantic segmentation methods and achieves comparable and even better results compared with some fully-supervised methods.

Chapter 6

Point Contrast and Labeling for Weakly-supervised Point Cloud Semantic Segmentation

6.1 Introduction

This chapter focuses on the task of weakly-supervised point cloud semantic segmentation with incomplete labels, and the corresponding configuration is shown in Figure 2.5 (b). Recently-proposed point cloud semantic segmentation approaches, applying deep learning, have achieved outstanding results with the help of large-scale fully annotated training datasets [72, 74, 59, 91, 112, 95]. However, designing such fully annotated datasets involves a significant effort in terms of data cleansing and manually labeling, especially the point-level annotations. To deal with the annotation bottleneck, label-efficient approaches are highly desirable, aiming to complete the 3D point cloud semantic segmentation task based on only small amounts of point-level labels, also defined as weakly-supervised approaches. Very recently, some attempts [119, 127, 34, 63] have made good progress in label-efficient point cloud semantic segmentation, but there are still many challenges.

Most supervised approaches formulate the semantic segmentation in 3D point clouds as a per-point classification task and train their proposed neural networks with the loss function of cross-entropy. However, their performance degrades seriously due to the reduction of the number of labeled data. A major reason is that those deep learning models tend to overfit to a tiny amount of annotated data when trained with cross-entropy loss, limiting the representational ability of the learned

networks. Thus, existing attempts pay much attention to address this problem. [34] develops a *self-supervised pre-training* pretext on an additional rich source dataset to help the network learn good representations, and then fine-tune on the weakly-supervised semantic segmentation task. [63] explores the *self-training* strategy that takes high confident predictions as pseudo labels to introduce more supervision information. [119] develops a multi-branch framework that explores *additional constrains on unlabeled points*, *e.g.* self-supervision, spatial & color smoothness constraint. Each of the above approaches has its own strengths, but also has its limitations: 1) Self-supervised pre-training is an effective way to mitigate the overfitting problem, but the gap between the optimization objectives of self-supervised pretext tasks and the target task (semantic segmentation) limits the power of the pre-training model. 2) Using pseudo labels is a simple and effective way to address weakly-supervised semantic segmentation, but noises existing in pseudo labels will impact the training process, which is hard to be identified using predicted confidences, 3) Introducing more constraints on unlabeled points helps to learn better representations, but directly combining self-supervised task on unlabeled data with segmentation task on labeled data may result in in-convergent problem or suboptimal models, because of the gap between their optimization objectives.

Motivated by the above considerations, we propose the point contrast and labeling framework (PCL) for weakly-supervised point cloud semantic segmentation. Specifically, our proposed PCL framework introduces two more constraints on labeled or unlabeled points, except for the standard cross-entropy loss function. As contrastive learning helps to learn more discriminative feature representations by directly regularizing the learned feature space, we propose to adopt the contrastive learning technique to enhance the representational ability of the segmentation model. Specially, we introduce two relationships: the cross-sample point contrastive loss for cross-sample semantic consistency and the low-level similarity-based point con-

trastive loss for local smoothness. By introducing the two contrastive losses, the built model can make learned representations of the same category more compact and learned representations of different categories more separable, resulting in a significantly improved per-point classifier. Moreover, in order to maximize and highlight the role of the given limited supervision information, we adhere to the pseudo labeling strategy [56], which has been widely applied to many weakly-supervised methods [41, 40]. To relieve the impact of incorrect pseudo labels, we propose the pseudo label refinement module to dynamically refine the pseudo label during the training process. Instead of depending only on the final predictions, we consider the prediction history and use the ensemble predictions to generate pseudo labels, and update pseudo labels dynamically, which makes pseudo labels more robust and reliable. We evaluate the proposed method on a various set of label-efficient point cloud semantic segmentation tasks. The proposed method achieves top performance under multiple settings and datasets.

We summarize our main contributions from the three aspects below:

- We propose the PCL framework for weakly-supervised point cloud semantic segmentation, exploring massive unlabeled data to enhance feature learning.
- An extended contrastive learning technique is developed for weakly-supervised point cloud semantic segmentation, proposing two relations, cross-sample point contrast and low-level similarity-based point contrast, to directly supervise the feature space.
- We propose a pseudo label refinement module, which refines pseudo labels online and explores the prediction history to generate robust and reliable pseudo labels.

In the rest, Section 6.2 introduces the proposed PCL framework, and the details

of this method are also given. In addition, the details of our implementation and experiment results are also introduced and discussed in Section 6.3. At last, we give the conclusions in Section 6.4.

6.2 Methods

This section firstly introduces the overview of our presented point contrast and labeling (PCL) framework for weakly-supervised point cloud semantic segmentation. Then we give the detailed description of computing the cross-sample point contrastive loss (L_{cont}) with memory, the similarity loss (L_{sim}), and the pseudo label refinement module.

6.2.1 Overview of PCL

Figure 6.1 depicts our presented PCL framework. As shown in Figure 6.1, our training procedure is divided into two stages: a warm-up stage and a training stage. i) In the warm-up stage, we train the segmentation network (consisting of a backbone and a segmentation head) with the given sparse labels using a cross-entropy loss in a standard supervised fashion for $T_{warm-up}$ epochs, where $T_{warm-up}$ denotes the number of epochs for warm-up stage. The warm-up stage pushes the network to fit the sparsely labeled data, and initializes the backbone and segmentation head. ii) In the training stage, the network contains a backbone, a segmentation head and a projection head. We train the network with pseudo labels using three losses: cross-entropy loss, cross-sample point contrastive loss and similarity loss. Besides, we propose a pseudo label refinement module to update the pseudo labels dynamically, and a memory to enrich samples for the cross-sample point contrastive loss. The formal definitions are as follows.

A given point cloud is denoted as $\mathbf{P} = \{\mathbf{P}^l, \mathbf{P}^u\} \in \mathbb{R}^{N\times(3+D)}$, where N is the number of points, 3 denotes the XYZ coordinates, D denotes the additional feature

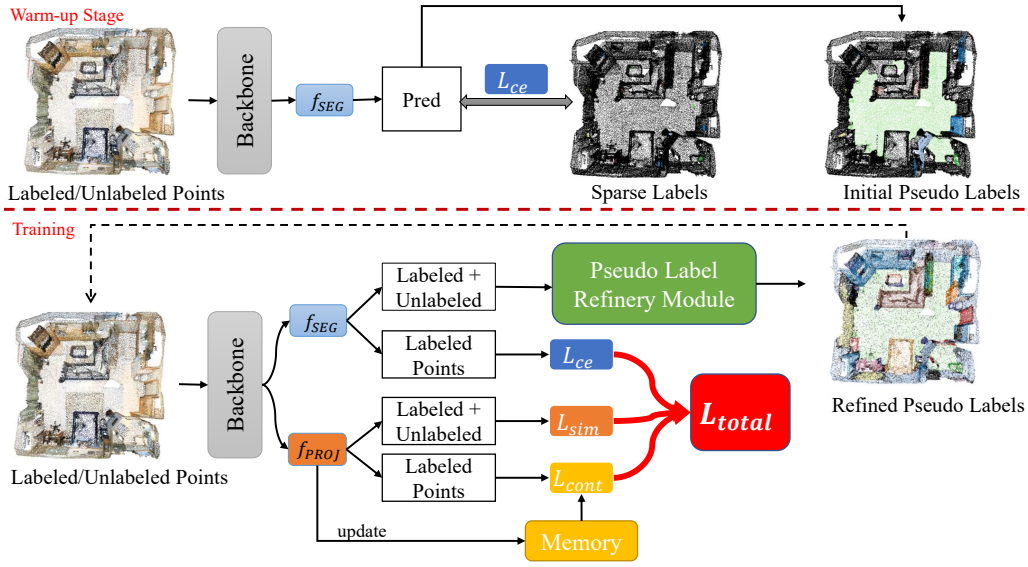


Figure 6.1 : Illustration of our PCL framework. The training pipeline has two stages: a warm-up stage and a training stage. i) In the warm-up stage, the network contains a backbone and a segmentation head f_{SEG}. We adopt the cross-entropy loss L_{ce} to train the network using the sparse labels. ii) In the training stage, the network contains two heads: a segmentation head f_{SEG} and a projection head f_{PROJ}. The segmentation head outputs segmentation predictions, and the projection head outputs per-point embeddings. The training loss function L_{total}consists of three components: L_{ce}, which is the cross-entropy loss and applied on the predictions of labeled points; L_{cont}, which is the cross-sample point contrastive loss and applied on the embeddings of labeled points with the maintained memory; and L_{sim}, which is the similarity loss and applied on the embeddings of both labeled and unlabeled points. Besides, the pseudo labels are taken as correct labels for the training stage, and the pseudo label refinement module is used to produce the pseudo labels on the fly. The black color on labels denotes unlabeled.

dimensions like RGB color, $\mathbf{P}^l = \{p_1^l, ..., p_{N^l}^l\} \in \mathbb{R}^{N^l \times (3+D)}$ represents the set of N^l labeled points and $\mathbf{P}^u = \{p_1^l, ..., p_{N^u}^u\} \in \mathbb{R}^{N^u \times (3+D)}$ represents the set of N^u labeled points and $N^l + N^u = N$. Its incomplete label is denoted as $Y = \{Y^l, Y^u\}$, where

$Y^l = \{y_i^l | y_i^l \in \{1, ..., C\}, i \in \{1, ..., N^l\}\}$ denotes the set of labels of labeled points, C is defined as the number of classes. We use $Y^u = \{y_i^u | y_i^u = -1, i \in \{1, ..., N^u\}\}$ to mark the unlabeled points and use -1 denotes the ignored label. In the warm-up stage, a batch of point clouds, $\{(\mathbf{P}_b, Y_b)\}_{b=1}^B$, where B denotes the batch size, are fed into the backbone and segmentation head f_{SEG} to generate the segmentation predictions $\{\mathbf{Z}_b\}_{b=1}^B$, where $\mathbf{Z}_b = \{z_{b,1}, ..., z_{b,N_b}\} \in \mathbb{R}^{N_b \times C}$, N_b denotes the point number of point cloud $\mathbf{P}_b$. We adopt the cross-entropy loss L_{ce} to train the network using the labeled points $\{\mathbf{P}_b^l\}_{b=1}^B$:

$$L_{ce} = -\frac{1}{\sum_b N_b^l} \sum_b^B \sum_i^{N_b^l} \sum_c^C {y'}_{b,i,c}^l \log \frac{\exp(z_{b,i,c})}{\sum_c \exp(z_{b,i,c})}, \tag{6.1}$$

where ${y'}_{b,i}^l$ represents the one-hot vector of point label $y_{b,i}^l$ and ${y'}_{b,i,c}^l$ is the c-th channel of ${y'}_{b,i}^l$.

After training the network for $T_{warm-up}$ epochs, we start the training stage with pseudo labels. A pseudo label $\hat{Y}$ for each point cloud $\mathbf{P}$ is taken as the correct label and refined on the fly. Details for generating and dynamically refining the pseudo labels are explained in Section 6.2.4. Taking a batch of point clouds $\{(\mathbf{P}_b, \hat{Y}_b)\}_{b=1}^B$ as input, the backbone generates the per-point embeddings for each point cloud. Then a segmentation head f_{SEG} receives the per-point embeddings as input and outputs segmentation predictions $\{\mathbf{Z}_b\}_{b=1}^B$. The cross-entropy loss L_{ce} are applied on labeled points, as Eq.(6.1). A projection head maps the per-point embeddings into low-dimensional embeddings $\{\mathbf{F}_b\}_{b=1}^B$, where $\mathbf{F}_b = \{\mathbf{F}_b^l, \mathbf{F}_b^u\} \in \mathbb{R}^{N_b \times K}$, K is the dimension, N_b is the number of points of $\mathbf{P}_b$, $\mathbf{F}_b^l$ denotes the low-dimensional embeddings of labeled points and $\mathbf{F}_b^u$ denotes the low-dimensional embeddings of unlabeled points. We apply the cross-sample point contrastive loss L_{cont} on $\mathbf{F}_b^l$ as Eq.(6.3) and apply the similarity loss L_{sim} on both $\mathbf{F}_b^l$ and $\mathbf{F}_b^u$ as Eq.(6.5). Details of computing losses L_{cont} and L_{sim} are explained in Section 6.2.2. We also maintain an embedding memory for L_{cont} and update the memory in every training step.

Ultimately, the total loss L_{total} used to optimize the network is formulated as:

$$L_{total} = L_{seg} + L_{cont} + L_{sim}, \tag{6.2}$$

6.2.2 Cross-sample Contrast

The cross-entropy loss only cares about the relative relationship between logits and can not directly constrain the learned representation [101]. To learn more discriminative feature representations, we introduce supervised contrastive learning to directly regularize the embedding space. Contrastive learning aims to learn representations by contrasting similar (positive) data pairs against dissimilar (negative) pairs. The basic principle of supervised contrastive learning is to construct many positive and negative sample pairs according to the given labels. Specifically, for a labeled point p_i^l with its class label $\bar{c}$, the positive samples are other points with the same class label $\bar{c}$, while the negatives are the points with the other class labels $C \backslash \bar{c}$. Then the supervised point-wise contrastive loss is defined as:

$$\mathcal{L}_{cont}(i) = \frac{1}{|\mathcal{S}_i^p|} \sum_{f_i{}^+ \in \mathcal{S}_i^p} -\log \frac{\exp\left(f_i \cdot f_i{}^+ / \tau\right)}{\exp\left(f_i \cdot f_i{}^+ / \tau\right) + \sum_{f_i{}^- \in \mathcal{S}_i^n} \exp\left(f_i \cdot f_i{}^- / \tau\right)}, \tag{6.3}$$

where $\mathcal{S}_i^p$ and $\mathcal{S}_i^n$ are the set of point embeddings of the positive and negative samples for point p_i, respectively. For the weakly-supervised settings, the point labels are incomplete, which results in limited exemplars, thus we allow the positive/negative samples and the anchor i to be from different point clouds, as shown in Figure 6.2. During training, our contrastive loss aims to pull the point embeddings belonging to the same class close, and push point embeddings belonging to different classes apart. Overall, the cross-sample point contrastive loss is:

$$L_{cont} = \sum_i \mathcal{L}_{cont}(i), \tag{6.4}$$

Memory. The definitions of positive and negative samples are two key ingredients in contrastive learning. As revealed by many studies [104, 45, 22], maintaining

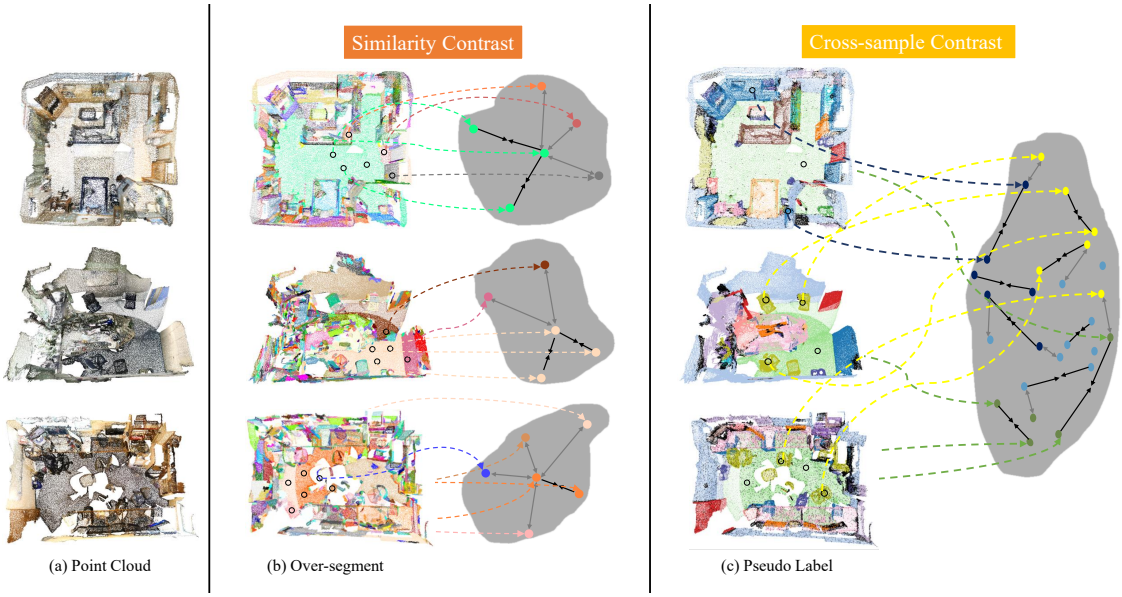


Figure 6.2 : Two types of point-level relationships for our cross-sample contrastive loss and similarity loss. From left to right: (a) the input point clouds, (b) over-segments with examples of similarity contrast relationship in the feature space, (c) pseudo labels with examples of cross-sample contrast relationship in the feature space. Black circles in (b) and (c) mark the sampled points, they are colored in the feature space by the corresponding over-segments labels or the pseudo labels. Pairs of black arrows connect positive pairs that should be brought close, while pairs of gray arrows connect negative pairs that should be pushed away.

a large memory is a critical way to enlarge the sets of the positive and negative samples and thereby help to learn good representations. Therefore, we build a memory bank to make use of massive data across the whole training dataset. Due to the number of pseudo labeled points increasing as the training process continues, directly storing all pseudo labeled points, like traditional memory, will greatly slow down the learning process. Therefore, we only store sampled points of each point cloud for efficient computation. Meanwhile, considering the class imbalance problem, to avoid being dominated by some large-portion classes (*i.e.* classes with very

few points may have little chance of being randomly sampled), we choose to maintain a point queue per class to form the class-balanced memory. Besides, to exhaust all pseudo labeled points to capture more point cloud content, we further build a segment memory bank that stores more representative embeddings absorbed from point cloud segments.

Specifically, for a dataset with $|\mathcal{C}|$ semantic classes, our **Class-balanced Pixel Memory** is built with size $|\mathcal{C}| \times M \times K$, where M represents the number of elements per class in the memory, K denotes the dimension of point embeddings. We randomly sample M_p pseudo labeled points per class from every point cloud in the latest mini-batch, and add their embeddings into the corresponding point queue to update the pixel memory bank. Our **Class-balanced Segment Memory** is built with the same size of pixel memory, $|\mathcal{C}| \times M \times K$, but stores segment embeddings. For each point cloud, we define the segment embedding of class $\bar{c}$ as the K-dimensional feature vector generated by average pooling all the embeddings of points labeled as class $\bar{c}$. As the number of the segment embeddings in each point cloud is not so much, we use all segment embeddings of the latest mini-batch to update the segment memory.

With the built memory, when computing the contrastive loss Eq.(6.3), the positive set $\mathcal{S}_i^p$ of an anchor point i belonging to class $\bar{c}$ is be composed of M point embeddings with the same class $\bar{c}$ in the class-balanced pixel memory and M segment embeddings with class $\bar{c}$ in the class-balanced segment memory, totally $2M$ positive samples. Similarly, the negative set $\mathcal{S}_i^n$ of an anchor point i is composed of the point embeddings and segment embeddings with the other classes. The memory allows our contrastive loss to explore point-to-point relations and point-to-segment relations, and thus helps to learn more discriminative feature representations.

6.2.3 Low-level Contrast

In the context of point cloud semantic segmentation, the relationships between points are very important. In general, the predictions of points with similar appearance or geometric properties are likely to be the same. With this observation, we propose the similarity loss by exploring low-level contrast. Its objective is similar to that of contrastive loss, and the difference only exists in the construction of training samples. We exploit low-level feature similarity to create the positive and negative samples for the similarity loss. Next, we will give the details to build the positive/negative samples.

As revealed in Figure 6.2, each point cloud is over-segmented to several clusters. For an anchor point i belonging to a cluster, the other points from the same cluster form the positive samples of i, denoted as $\mathcal{G}_i^p$. The points from the other clusters form the negative samples of i, denoted as $\mathcal{G}_i^n$. Note that the $\mathcal{G}_i^p$ and $\mathcal{G}_i^n$ are from a same point cloud. Then the similarity loss is formulated as:

$$\begin{aligned} L_{sim} &= \sum_i L_{sim}(i) \\ &= \sum_i \frac{1}{|\mathcal{G}_i^p|} \sum_{{f_i}^+ \in \mathcal{G}_i^p} -\log \frac{\exp\left(f_i \cdot {f_i}^+/\tau\right)}{\exp\left(f_i \cdot {f_i}^+/\tau\right) + \sum_{{f_i}^- \in \mathcal{G}_i^n} \exp\left(f_i \cdot {f_i}^-/\tau\right)} \end{aligned} \tag{6.5}$$

Over-segmentation. We follow the superpoint-generation method provided by [53] for over-segmentation. Because this over-segmentation method is unsupervised and computationally efficient. More importantly, the over-segmentation is characterized by adaptive local geometric complexity. For example, a generated cluster can be small to cover only a part of a geometric-complex object such as a chair and large to cover a whole simple shape such as a wall.

6.2.4 Pseudo Label Refinery

Using pseudo labels for self-training is popular in weakly-supervised learning. However, it is natural for the predicted probability to exist errors. These noisy

pseudo labels would harm the learning process significantly. Therefore, it is desirable to design a strategy to filter noisy labels. [63] presents that the region with the confidence less than a predefined threshold should be ignored. As classification confidence is a widely adopted criterion to measure the quality of predictions in self-training. However, as the network is easy to fit easy samples, using a fixed threshold will result in hard complex samples being filtered as noisy data. In this work, we develop a pseudo label refinement module for online pseudo label re-correction during training. The main idea is that the pseudo labels are refined based on the ensemble predictions(combining the prediction history) instead of the latest predictions.

For each point cloud $\mathbf{P}_i$, we define $\hat{Y}_i$ as its one-hot pseudo-label. In every training step, we take the segmentation predictions $\mathbf{Z_i}$ of point cloud $\mathbf{P}_i$ with confidences higher than a threshold δ as the currently correct predictions, and save them to a prediction history $H = \{H_1, ..., H_N\}$, where N is defined as the number of point clouds in the training set. After every T training epochs, we take a voting strategy to generate the latest predictions $\hat{Z}_i^T$ for every point cloud, called the latest soft pseudo labels. We define $\hat{Z}_i^E$ as the ensemble soft pseudo labels of point cloud $\mathbf{P}_i$, and update $\hat{Z}_i^E$ every T epochs:

$$\hat{Z}_i^E = \alpha \hat{Z}_i^E + (1 - \alpha)\hat{Z}_i^T, \tag{6.6}$$

where α is a momentum term which controls how far the ensemble reaches in the history of training. In the following, the pseudo labels is updated as following:

$$\hat{Y}_i = \hat{Z}_i^E / (1.0 - \alpha^{t+1.0}), \tag{6.7}$$

where t denotes the epochs.

6.3 Experimental Settings and Results

This section compares the proposed method with the state-of-the-art point cloud semantic segmentation models. We first introduce the details of our implementation

in Section 6.3.1. And then in Sections 6.3.2 and 6.3.3, we provide the performance of our method on ScanNet [17] dataset and Stanford 3D Indoor Scene (S3DIS) [2] dataset, respectively. To gain a deeper insight into our model, we conduct detailed ablation studies in Section 6.3.4. At last, some qualitative results are also shown and analyzed in Section 6.3.5.

6.3.1 Datasets and Experimental Setup

Datasets

Extensive experiments are carried out on ScanNet and S3DIS, which are two benchmarks for most of the weakly-supervised point cloud semantic segmentation published papers.

ScanNet [17] is a large-scale indoor dataset. It comprises 1513 scenes with fully per-point annotations opening to the public and 100 scenes without any annotations for online testing. Each scene is represented as a point cloud, and each point are provided with the XYZ coordinates and RGB color. ScanNet contains 20 selected classes for segmentation evaluation. In our experiments, we use the official train-val partition to further separate the 1513 point clouds, with 1201 point clouds as the training set and 312 point clouds as the validation set, same as [17, 34]. Evaluations are conducted on both the validation set and testing set. **ScanNet-LA** [34] is a weakly-supervised online benchmark dataset, whose data is the same as **ScanNet** and weak labels are in four settings including using {20, 50, 100, 200} labeled points per scene. More details about ScanNet can be found in the Section 5.3.1.

S3DIS [2] is another large-scale indoor dataset. It contains 271 point clouds with 13 semantic classes. Each point cloud is provided with XYZ coordinates and RGB values. All point clouds are distributed in six areas. In our experiments, we adopt results on Area-5 for evaluation, same as [92, 15, 63]. In total, there are 204 training point clouds and 68 testing point clouds. Please refer to the Section 3.3.1

for detailed description of the S3DIS.

Evaluation Metrics.

We adopt mIoU to assess the performance of our method, same as most point cloud semantic segmentation approaches [34, 63, 95]. We describe the details of computing mIoU in Section 3.3.1.

Annotation Configuration

We evaluate our method on various weakly-supervised annotation settings. For ScanNet, we consider five different training settings including making use of {20, 50, 100, 200} annotation points per point cloud (following benchmark ScanNet-LA of [34]) and {1t1c} per scene (one labeled point per object in each scene following [63]). For S3DIS, we consider three different training configurations including {1t1c, 1t3c, 0.01%, 0.1%, 1%} labeled points per point cloud, where {1t3c} indicates three labeled points per object in each scene.

Training Details

In all experiments of this work, we use the same and fixed backbone named Sparse Res-UNet [34, 115], which is implemented by MinkowskiEngine [15]. The projection head is constructed with two fully connected layers. We set 32 as the finally projected dimension. During training, the backbone is initialized by the pre-training model of [34]. SGD optimizer is used for training, and we set 0.1 as its initial learning rate. Meanwhile, the learning rate is gradually decreased with polynomial decay with a power of 0.9. In each experiment, the batch size is set to 6, with 8 sub-iterations per step on 1 NVIDIA RTX6000 GPU. The warm-up stage is trained for 2K steps. Finally, the model is trained for 10K steps in total. For ScanNet, the voxel size of used Sparse Res-UNet in our experiments is the same with 2.0 cm for both training, validation and testing. We adopt the over-segments

provided by official annotations to generate low-level segments. For S3DIS, the voxel size is fixed with 5.0 cm for both training and testing. We follow [53, 63] to generate low-level segments. For both ScanNet and S3DIS, the final full resolution results are generated by nearest-neighbor interpolation.

6.3.2 Performance on ScanNet

First of all, our method is evaluated on the ScanNet under different weakly-supervised settings, as well as the fully-supervised counterpart of our proposed method. We report the results from ScanNet online testing benchmark in Table 6.1 and Table 6.2.

Comparisons with weakly-supervised methods.

For fair comparisons with existing weakly-supervised methods, we compare our proposed method with three existing weakly-supervised methods under different settings:

i) MPRM [109] is trained subcloud-level labels, which only denote a category existing in a given point cloud or not, and no location information. With only 20 labeled points (20pts), our result (60.8%) outperforms MPRM [109] in a large margin (+17.6%). When increasing the labeled points, the performance increases continuously to 68.8% for 200pts. In regard to annotating cost, as reported in [109], the subcloud-level annotation takes around 3 minutes per scene, which is longer than the 20pts scheme (about 1 minute). We can infer that, under the same annotation cost, weakly-supervised methods can get more benefit from the point-level labels than subcloud-level labels. Because point-level labels can provide useful location information (even if very little), which is extremely important for the point cloud semantic segmentation task.

ii) CSC [34] and OTOC [63] are two public work that evaluates on the ScanNet

data-efficient online benchmark. Therefore, we report our semantic segmentation results with the same annotation configuration in Table 6.1. As shown n Table 6.1 we can easily find that the performance of both ours and the other two methods improves when increasing the labels from 20pts to 200pts. This phenomenon is reasonable and the performance of the fully-supervised counterpart can be seen as the corresponding upper-bound. Moreover, when training with 20 randomly annotated points per point cloud (20pts), our method achieves the top performance (60.8%), outperforming OTOC [63] by 0.6% and CSC [34] by 7.7%. When training with 50 randomly annotated points per point cloud, our method outperforms OTOC [63] by 1.0% and CSC [34] by 4.0%. When training with 100 randomly annotated points per point cloud, our method outperforms OTOC [63] by 1.0% and CSC [34] by 3.6%. Under the configuration of 200pts, our result is a little lower than OTOC [63], but it still outperform CSC [34] by 1.7%. The results also demonstrate that our method gains more improvement under the situation of training with fewer labels.

Table 6.1 : Comparisons with existing weakly-supervised methods for 3D point cloud semantic segmentation using the same ScanNet testing set (from online benchmark on 9 Jan 2022). Bold fonts mark the best result under different settings.

		Methods (mIoU(%))			
		MPRM [109]	CSC [34]	1T1C [63]	Ours
Setting	subcloud-level	43.2	-	-	-
	20 pts	-	53.1	59.4	**60.8**
	50 pts	-	61.2	64.2	**65.2**
	100 pts	-	64.4	67.0	**68.0**
	200 pts	-	66.5	**69.4**	68.8
	1t1c	-	-	69.1	**69.6**

iii) OTOC [63] proposes to use 1 annotated point per category per scene (1t1c). Our method under the 1t1c setting achieves a little higher performance than OTOC (+0.5%). Both OTOC and Ours trained with 1t1c (about 0.02% annotated points) outperform the corresponding counterpart trained with 50pts (about 0.02% annotated points). This also inspires us to develop a more efficient annotating strategy under a fixed cost.

Comparison with fully-supervised methods

We also train a fully-supervised counterpart of the proposed method, and compare it with the fully-supervised methods, including KPConv [95], RandLA-Net [35], MinkUNet [15]. As shown in Table 6.2, we can easily see that our method gain 71.8%, which is slightly lower than CSC [34]. This result is reasonable because ScanNet contains noises, and our method dynamically filters and refines the ground truth during training, which may generate predictions that differ from the given ground truth.

Overall, the results of fully-supervised methods can be seen as the upper bound of the performance of weakly-supervised methods. We can observe that with about

Table 6.2 : Performance of our fully-supervised baseline on ScanNet testing dataset. Bold represents the best result.

Methods	Annotation	mIoU(%)
KPConv [95]	100%	68.4
RandLA-Net [35]	100%	64.5
MinkUNet [15]	100%	72.2
CSC [34]	100%	**73.8**
Ours	100%	71.8

0.02% (1t1c) randomly annotated points, our proposed method (69.6%) reaches a performance comparable to that of the advanced fully-supervised point cloud semantic segmentation methods. This demonstrates that full supervision is unnecessary for point cloud semantic segmentation, which encourages us to develop more label-efficient methods to save annotating costs.

6.3.3 Performance on S3DIS

In order to further illustrate the effectiveness and generalizability of our proposed method, we also evaluated on S3DIS. Table 6.3 compares our methods with two groups of existing methods: 1) State-of-the-art fully-supervised methods trained

Table 6.3 : Comparisons with existing methods on S3DIS.

Methods	Annotations	mIoU(%)
KPConv [95]	100%	63.7
MinkUNet [15]	100%	68.2
CSC [34]	100%	**72.2**
OTOC [63]	100%	63.7
Ours	100%	68.8
Xu [119]	10%	48.0
Ours	0.01%	38.2
Ours	0.1%	60.5
Ours	1%	**64.2**
OTOC [63]	0.06%(1t3c)	55.3
OTOC [63]	0.02%(1t1c)	50.1
Ours	0.06%(1t3c)	**61.8**
Ours	0.02%(1t1c)	56.2

with 100% training labels; 2) Weakly-supervised methods trained with different types of weak labels.

Similar to Table 6.1, we also group the weakly-supervised methods into two subgroups. i) For randomly annotating labels, our method continuously improves performance when the number of annotated labels increases from 0.01% to 1%. When compared with Xu [119], even with 0.1% labels, our result outperforms Xu [119] with 10% in a large margin. ii) For the configuration of 1t1c (one labeled point per category per point cloud), with the same data preprocessing, our method outperforms OTOC [63] by 6.1%. When increasing the annotated points to 1t3c, our method achieves 61.8% mIoU, outperforming OTOC (55.3%) by 6.5%. In total, the configuration of 1t3c is equivalent to 0.06% annotated points. Comparing our results of 1t3c and 0.1% configurations, we can infer that with the fixed annotating budget (*i.e.* the same annotated points), the more exhaustive annotating strategies are more likely to result in better performance.

Compared with fully-supervised methods, our method (64.2%) with 1% labels performs even better than KPConv [95] (63.7%), and a little worse than MinkUNet [15] and CSC [34]. But for 1t1c setting, the gap between our method and the fully-supervised ones is quite large, compared with results on ScanNet 6.1. This may be owing to the differences between the two datasets. Point clouds in ScanNet contain many unclassified (ignored) objects and noises, which may affect the performance of fully-supervised methods. While our method trained with incomplete labels (e.g. 1t1c, 200pts) is designed to filter and recorrect labels during training, which will reduce the impact. But S3DIS is a clean dataset without unclassified objects and noises, therefore, fully-supervised methods can gain more increases than our method.

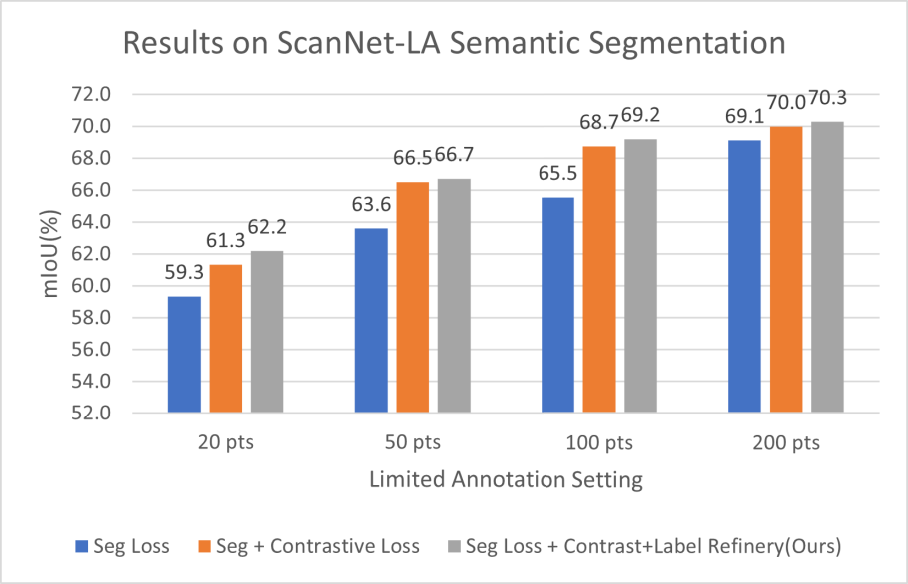


Figure 6.3 : Performance on ScanNet using different losses.

6.3.4 Ablation Study

Effectiveness of Different Losses. In this subsection, the importance of individual components of our framework is mainly analyzed and compared. We evaluate models trained with different combinations of the introduced losses on ScanNet validation set. We consider four annotation configurations provided by the ScanNet data-efficient benchmark, i.e. {20pts, 50pts, 100pts, 200pts}. As shown in Figure 6.3, adding the contrastive loss improves the segmentation performance by 2.0% for 20pts, 2.9% for 50pts, 3.2% for 100pts, 0.9% for 200pts, respectively. The performance is further increased by adding the label refinement module. Moreover, we can observe that the contrastive loss gains more performance improvement than the label refinement module.

Effectiveness of our pseudo labels. As we have analyzed before, using more annotations usually results in better performance or at least no performance degradation. To further investigate the effectiveness of our label refinement module, we evaluate the final pseudo labels using four metrics: overall accuracy (OA), mIoU,

mean of per-class accuracies (mAcc) and mean of per-class recalls (mRecall). We also report the corresponding segmentation results on testing set. Baseline denotes training without label refinement module and contrastive losses. Results denote our final results on testing set.

Table 6.4 displays the performance of our final pseudo labels on S3DIS. We notice that there is a high correlation between the segmentation performance and the quality of the pseudo labels. Specifically, we notice the pseudo labels gain the best results on all metrics with 1% annotated points. The OA and mRecall of the pseudo labels are all about 88%, which are quite close to the full annotations. We can observe that our testing result with 1% annotated points achieves 64.2%, which is also close to the result with full annotations, 68.8%.

The final pseudo labels gain the worse results on all metrics with 0.01% annotating points. The mRecall of the pseudo labels for the configuration of 0.01% annotating points is 53.3%, which means the pseudo labels cover more than 50% supervision, but the mAcc is only 58.9%, the incorrect pseudo labels hinder the network learning. Therefore there is a certain gap with the segmentation results(54.8% mIoU) of the baseline model trained with 0.1% annotated points.

Table 6.4 : Evaluation on Pseudo labels.

Supervision	Training Set				Testing Set	
	OA(%)	mIoU(%)	mAcc(%)	mRecall(%)	Result(%)	Baseline(%)
1%	88.9	83.5	93.6	87.9	64.2	61.7
0.1%	84.6	70.9	82.9	77.9	60.5	54.8
0.01%	70.3	42.6	58.9	53.3	38.2	34.5
1t1c(0.02%)	80.5	67.2	83.7	78.7	56.2	51.2
1t3c(0.06%)	83.7	74.4	87.4	82.4	61.8	56.7

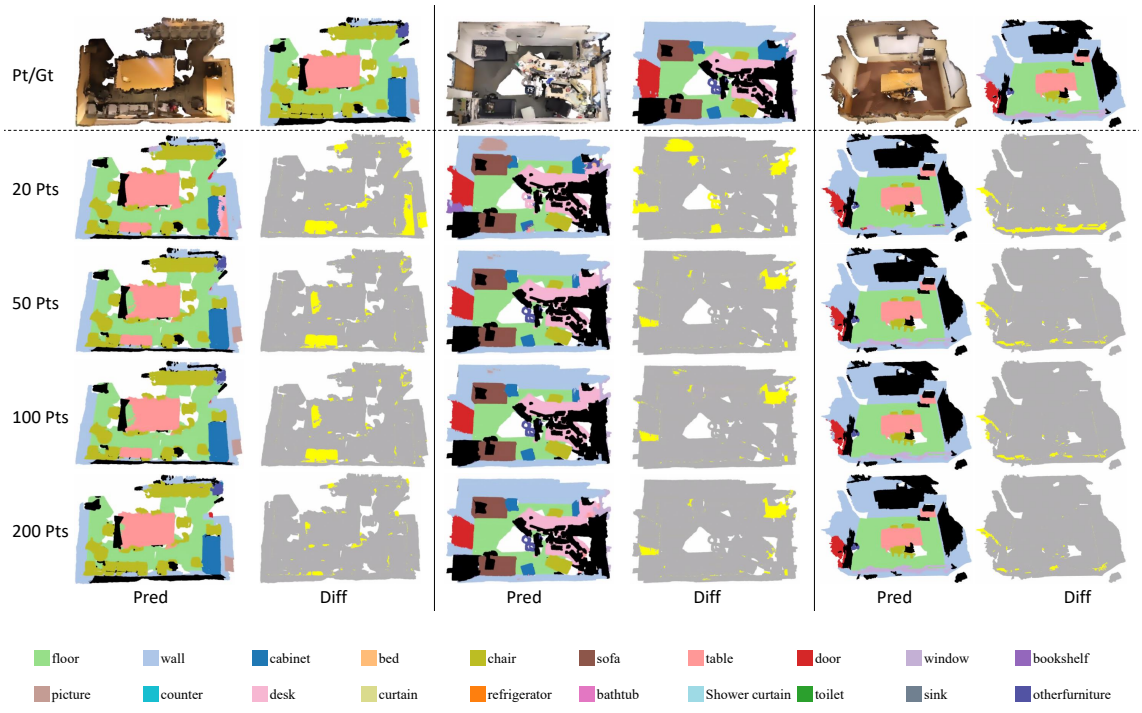


Figure 6.4 : Qualitative results on ScanNet validation set under different weak annotation settings. The first row shows the input point clouds (Pt) and their ground truths (Gt). The rest rows present our segmentation predictions (Pred) and their differences with ground truths (Diff), under weak annotation settings of $\{20, 50, 100, 200\}$ pts separately.

6.3.5 Qualitative Results on ScanNet and S3DIS

In this section, we visualize our segmentation results on ScanNet validation set and S3DIS testing set under different annotation configurations, respectively.

The clear qualitative segmentation results on ScanNet validation set are displayed in Figure 6.4. The first row in this figure shows the original point clouds and their corresponding ground-truth labels. In the ground truth, each color indicates a class and points with black color are ignored. The rest rows show our segmentation results (the Pred columns) and the corresponding difference with ground-truth (the Diff columns) under the annotation configurations of {20, 50, 100, 200} pts. In

the Diff columns, points with yellow color denote incorrect predictions. We observe that training with only 20 labeled points (20pts), our model can make very good predictions on most categories in a scene, such as floor, wall, chair and sofa. The incorrect predictions most exist in categories that have similar geometric structure (*e.g.* wall vs. window, wall vs. picture) or appearance (*e.g.* cabinet vs. table), and in the boundary regions (*e.g.* table vs. floor). With the increasing annotations, the segmentation results improve a lot and generate better predictions on categories with ambiguous geometric or appearance features, such as picture vs. wall. This demonstrates the amount of supervision have a key role in improve the segmentation performance and encourages us to gradually mining and introduce reliable pseudo labels during training. When training with 200pts, most objects can be predicted, and the most errors exist in the boundary regions. This inspires us to further enhance the segmentation performance by paying more attention to the boundary regions.

The qualitative results on S3DIS are demonstrated in Figure 6.5. Here, we visualize the segmentation results with different annotation configurations of {1t1c, 0.1%, 1%}. For the settings of random annotations, similar to ScanNet, the segmentation results are greatly improved when increasing the number of annotated points from 0.1% to 1%. We can observe that with 0.1% annotated points, our model can produce good predictions on classes with large scales (*e.g.* floor, ceiling) and with less diversity in geometric structure or appearance (*e.g.* chair, bookcase). On the contrary, our model may make more errors on classes with small scales (*e.g.* beam, door) or with various geometric structure or appearance (*e.g.* window and cluster). The reasons are: 1) For random annotations, large-scale classes are likely to have more annotated points, which helps the model to perform well on these classes. 2) Even if not have too much annotated points, it may be sufficient to learn to recognize some simple classes. As a matter of course, training with more annotations, 1%

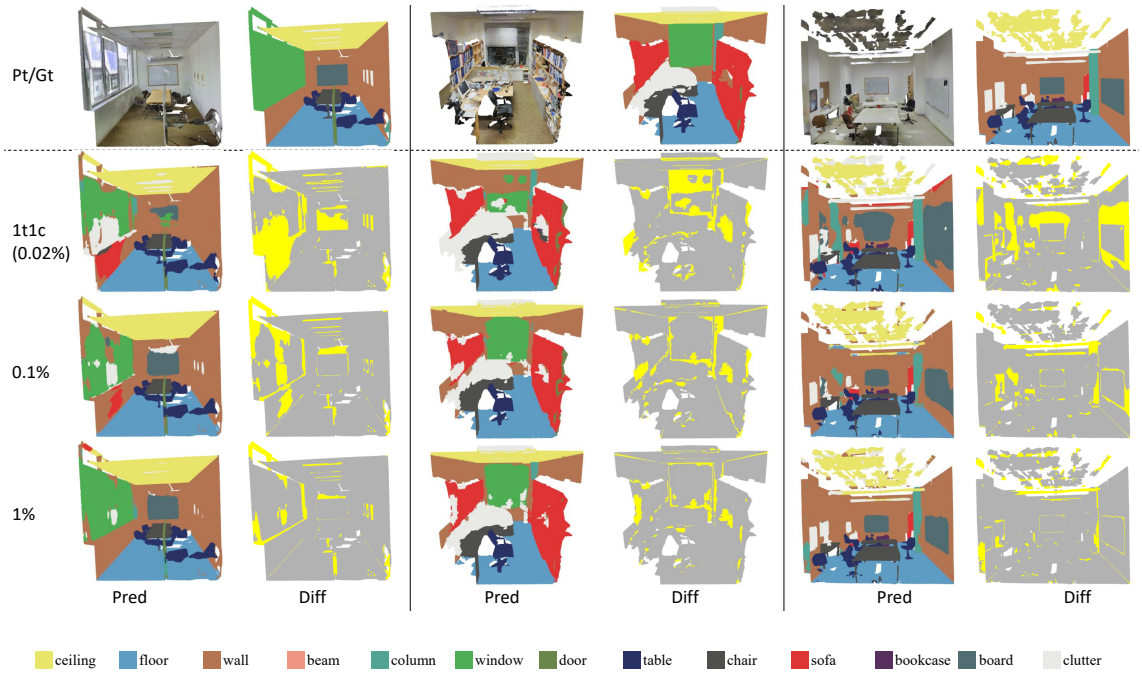


Figure 6.5 : Qualitative results on S3DIS under different weak annotation settings. The first row shows the input point clouds (Pt) and their ground truths (Gt). The rest rows present our segmentation predictions (Pred) and their differences with ground truths (Diff), under weak annotation settings of $\{1t1c, 0.1\%, 1\%\}$ separately.

annotated points, our method gains better predictions and performs well on most classes, such as wall, floor, ceiling, chair and table, and the incorrect predictions most exist in the boundary regions. For the setting of 1t1c, its total annotating budget in terms of *the percentage of annotated points* is only about 0.02%, which is much lower than the other two settings. However, our model can produce excellent perdition on more than half classes like chair, table, bookcase and floor. The reason is that this annotating strategy may gain more balanced annotation, which is helpful for small-scale classes. The results also inspire us to get effective annotations under a fixed budget.

6.4 Conclusions

In this chapter, a novel point contrast and labeling framework (PCL) is proposed for weakly-supervised point cloud semantic segmentation. With the proposed cross-sample point contrastive loss and low-level similarity based point contrastive loss, the proposed PCL framework can learn more discriminative feature representations by regularizing the feature space directly. Besides, we propose the pseudo label refinement module, which explores the prediction history to generate robust and reliable pseudo labels during the training process. Extensive experiments show the effectiveness and accuracy of our PCL framework and imply the practical label-efficient solution with a fixed annotating cost.

Chapter 7

Conclusions and Future Work

This chapter summarizes the major achievements of this thesis in dealing with the foundational tasks in 3D point cloud understanding. The potential directions for future research work are also introduced.

7.1 Conclusions

This thesis aims to exploit and design novel deep learning techniques for 3D point cloud understanding. As point clouds are irregular and usually large-scale, it is hard to capture long-short-range context, which is critical for semantic understanding tasks. Moreover, the data-hungry nature of supervised deep learning based approaches increases the difficulty of extending dense prediction approaches, *e.g.* point cloud semantic segmentation, to more application scenarios. In this thesis, we have proposed several methods to deal with these challenges. In the following paragraphs, we will summarize and highlight our contributions.

The first main contribution is the Long-Short-Term Context framework (LSTC) for supervised point cloud semantic segmentation, equipped with which enabled existing point cloud semantic segmentation approaches to capture long-short-term context in an arbitrary range, further improving the segmentation performance. Since natural scenes usually have a reasonable and coherent composition of objects, exploiting spatial context can help resolve the ambiguity of object labels against noises. Many existing point cloud semantic segmentation approaches takes a subdivided block of a whole point cloud and can not capture context beyond the input

block. It is convenient to extend the proposed LSTC framework to existing point cloud semantic segmentation approaches for capturing context over patches in a larger range. Even if some point cloud semantic segmentation approaches can deal with a whole large-scale point cloud, they may not consider the cross-sample relationships between point clouds. The proposed LSTC framework can be extended to capture context over multiple point clouds by establishing the Long-Short-Term feature bank with features from multiple point clouds.

The second main contribution is the weakly-supervised keypoint Siamese network for point cloud registration (KPSNet), which learns task-specific keypoint detector and feature extractor simultaneously without using manually annotated keypoints. The previous learning-based 3D keypoint description approaches takes a pair of local patches (*i.e.* the support regions of 3D keypoints) as input. Therefore, they requires dense annotated data that contains a number of matched 3D local patches for training. To reduce the annotating cost, the proposed KPSNet takes a pair of whole point clouds as input and only uses their relative transformation matrix as weak supervision for training. Moreover, the proposed KPSNet learns keypoint detector and feature extractor simultaneously in an end-to-end framework, which avoids the chicken-egg problem caused by separately learning keypoint detectors and descriptors.

The third main contribution is the weakly-supervised semantic segmentation method, which only uses coarse cloud-level labels and reduces the manual annotating cost greatly. The experimental results show the promising performance to catch up with supervised approaches. Different with the previous approaches that focus on generating better pseudo labels and retrain a segmentation model with all pseudo labels, the proposed approach pays more attention to reducing the ill-impact of noisy pseudo labels via additional unsupervised constrains. Therefore, the proposed approach can yield a additive gain in performance when applying a better pseudo

label generation module.

Finally, the fourth main contribution was the point contrast and labeling framework for label-efficient point cloud semantic segmentation, which used incomplete point-level labels as supervision. From the perspective of annotation cost, incomplete point-level labels is higher than coarse class labels. But regarding the segmentation performance, existing weakly supervised point cloud semantic segmentation approaches with incomplete point-level labels have significant advantages than with coarse weak labels, and are more valuable for real-world applications. The proposed approach obtained top performance on several label-efficient point cloud semantic segmentation tasks with the introduced cross-sample point contrastive loss, similarity loss, and the pseudo label refinement module. Therefore, it is deserved to further reduce the annotation cost, and further improve the segmentation performance by exploring the inner structure on unlabeled data and by mining more useful pseudo labels.

Overall, we have explored the field and proposed novel approaches from supervised learning to weakly-supervised learning for 3D point cloud understanding. We hope our research achievements and contributions from this thesis have provided the community with new and different insights into some of the existing challenges in 3D point cloud understanding.

7.2 Future Work

Deep learning on 3D data is still in the early ages of development, and there are many challenges remaining. In the following, we list several main challenges and introduce some of our future research directions.

- *Lack of benchmarks*: large-scale various benchmark datasets is very important for the rapid development of deep learning on 2D data. Many successful ap-

plications emerge based on the 2D deep learning techniques, which attracts much more attention to 2D deep learning and further promotes building more datasets covering a variety of application scenes. However, the quantity and diversity of 3D datasets are far from the 2D area, which limits 3D deep learning approaches to be applied to practical applications. Currently, the most popular 3D indoor dataset, ScanNet [17], only contains 1501 point clouds with dozens of annotated categories. While 3D outdoor datasets (*e.g.* Semantic3D [30], SemanticKITTI [4]) cover less than ten annotated categories, although with huge amounts of point clouds. Besides, different datasets are usually saved in different formats and require different data pre-processing, which increases the difficulty of developing and evaluating 3D deep learning approaches.

- *Lack of universal backbone architectures*: there are several successful deep architectures in the field of 2D deep learning, such as ResNets [33], serving as pre-trained backbones of many approaches for many 2D computer vision tasks. However, in the area of 3D, different kinds of architectures have been introduced by researchers for different tasks of 3D scene understanding. Although each architecture has its own advantages, it is hard to evaluate different approaches fairly, further improving these approaches and developing more applications.

- *Class imbalance*: class imbalance in complex scenes is a common phenomenon and has attracted much attention in the area of 2D scene understanding. This problem is even more severe in 3D data, resulting in many methods failing on rare categories for tasks of 3D scene understanding, especially for point cloud segmentation. However, rare works focus on the problem of class imbalance in the area of 3D.

- *Balance between accuracy and efficiency*: point clouds representing real scenes are usually on a large scale. Performing dense prediction tasks, like 3D scene semantic segmentation, often requires balancing the accuracy and efficiency, depending on the computational resources. Therefore, a lightweight segmentation system with high precision and high efficiency is extremely required.

Object Co-segmentation on Complex Scenes In recent years, 3D radar sensors have been developed rapidly, resulting in a mass of 3D data being collected for available uses. However, manually annotating 3D data is quite expensive. Co-segmentation aims to recognize and segment the common objects or parts among a collection of provided samples without any help from extra annotations. Some works have attempted to deal with co-segmentation of the 3D shape by segmenting single objects into several parts [14]. However, 3D object co-segmentation of complex scenes is still in its infancy, which is a valuable research direction.

Few-shot Point Cloud Semantic Segmentation Few-shot point cloud semantic segmentation aims to construct a semantic segmentation model using only a few annotated point clouds. Considering the challenges of lack of 3D annotated data and class imbalance on real scenes, we believe few-shot point cloud semantic segmentation is a promising direction.

Universal Framework for 2D and 3D Fusion Both 2D data and 3D data have their own advantages when applied in semantic understanding applications. Specifically, 2D images contain rich appearance information, and 3D point clouds provide precise geometric information. Intuitively, combining 2D images and 3D point clouds for visual recognition systems will bring the best performance. Therefore, in addition to exploring 2D and 3D scene understanding separately, we also aim to develop a universal framework to combine 2D and 3D data for real scene understanding.

Bibliography

[1] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "Spinnet: Learning a general surface descriptor for 3d point cloud registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, June 2021, pp. 11 753–11 762.

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 1534–1543.

[3] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, "D3feat: Joint learning of dense detection and description of 3d local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 6359–6367.

[4] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9297–9307.

[5] A. Boulch, B. Le Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks." *3DOR*, vol. 2, p. 7, 2017.

[6] A. Boulch, B. L. Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *3DOR*, 2017.

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[8] C. Chen, S. Qian, Q. Fang, and C. Xu, "Hapgn: Hierarchical attentive pooling graph network for point cloud segmentation," *IEEE Trans. Multimedia*, vol. 23, pp. 2335–2346, 2020.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2017.

[10] S. Chen, C. Duan, Y. Yang, D. Li, C. Feng, and D. Tian, "Deep unsupervised learning of 3d point clouds via graph topology inference and filtering," *IEEE Trans. Image Process.*, vol. 29, pp. 3183–3198, 2019.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2020, pp. 1597–1607.

[12] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 1907–1915.

[13] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[14] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "Bae-net: Branched autoencoder for shape co-segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 8490–8499.

[15] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 3075–3084.

[16] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 8958–8966.

[17] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 5828–5839.

[18] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 4, p. 76a, 2017.

[19] H. Deng, T. Birdal, and S. Ilic, "Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 602–618.

[20] ——, "Ppfnet: Global context aware local features for robust 3d point matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 195–205.

[21] A. Du, X. Huang, J. Zhang, L. Yao, and Q. Wu, "Kpsnet: Keypoint detection and feature extraction for point cloud registration," in *IEEE Int. Conf. Image Process.* IEEE, 2019, pp. 2576–2580.

[22] A. Du, S. Pang, X. Huang, J. Zhang, and Q. Wu, "Exploring long-short-term context for point cloud semantic segmentation," in *IEEE Int. Conf. Image Process.* IEEE, 2020, pp. 2755–2759.

[23] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1355–1361.

[24] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring

spatial context for 3d semantic segmentation of point clouds," in *IEEE Conf. Int. Conf. Comput. Vis. Worksh.*, 2017, pp. 716–724.

[25] G. Georgakis, S. Karanam, Z. Wu, J. Ernst, and J. Košecká, "End-to-end learning of keypoint detector and descriptor for pose invariant 3d matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 1965–1973.

[26] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 5545–5554.

[27] B. Graham, "Sparse 3d convolutional neural networks," *arXiv preprint arXiv:1505.02890*, 2015.

[28] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 9224–9232.

[29] J. Guerry, A. Boulch, B. Le Saux, J. Moras, A. Plyer, and D. Filliat, "Snapnet-r: Consistent 3d multi-view semantic labeling for robotics," in *IEEE Conf. Int. Conf. Comput. Vis. Worksh.*, 2017, pp. 669–678.

[30] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d. net: A new large-scale point cloud classification benchmark," *arXiv preprint arXiv:1704.03847*, 2017.

[31] M. Halber and T. Funkhouser, "Fine-to-coarse global registration of rgb-d scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 3, 2017.

[32] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 8160–8171.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[34] J. Hou, B. Graham, and M. Nießner, "Exploring Data-Efficient 3D Scene Understanding with Contrastive Scene Contexts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, jun 2021, pp. 15 587–15 597.

[35] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 11 108–11 117.

[36] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 984–993.

[37] H. Huang and Y. Fang, "Adaptive wavelet transformer network for 3d shape representation learning," in *Int. Conf. Learn. Represent.*, 2021.

[38] J. Huang and S. You, "Point cloud labeling using 3d convolutional neural network," in *Int. Conf. Pattern Recog.* IEEE, 2016, pp. 2670–2675.

[39] Y. Huang, Q. Wu, J. Xu, Y. Zhong, P. Zhang, and Z. Zhang, "Alleviating modality bias training for infrared-visible person re-identification," *IEEE Trans. Multimedia*, vol. 24, pp. 1570–1582, 2021.

[40] Y. Huang, Q. Wu, J. Xu, Y. Zhong, and Z. Zhang, "Unsupervised domain adaptation with background shift mitigating for person re-identification," *Int. J. Comput. Vis.*, vol. 129, no. 7, pp. 2244–2263, 2021.

[41] Y. Huang, J. Xu, Q. Wu, Z. Zheng, Z. Zhang, and J. Zhang, "Multi-pseudo

regularized label for generated data in person re-identification," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1391–1403, 2018.

[42] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[43] M. Jaritz, J. Gu, and H. Su, "Multi-view pointnet for 3d scene understanding," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2019.

[44] Z. Jian Yew and G. Hee Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 607–623.

[45] L. Jiang, S. Shi, Z. Tian, X. Lai, S. Liu, C.-W. Fu, and J. Jia, "Guided point contrastive learning for semi-supervised point cloud semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 6423–6432.

[46] L. Jiang, H. Zhao, S. Liu, X. Shen, C.-W. Fu, and J. Jia, "Hierarchical point-edge interaction network for point cloud semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 10 433–10 441.

[47] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3d instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 4867–4876.

[48] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 5, pp. 433–449, 1999.

[49] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 695–711.

[50] A. Komarichev, Z. Zhong, and J. Hua, "A-CNN : Annularly Convolutional Neural Networks on Point Clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[51] A. Kundu, X. Yin, A. Fathi, D. Ross, B. Brewington, T. Funkhouser, and C. Pantofaru, "Virtual multi-view fusion for 3d semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 518–535.

[52] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3d scene labeling," in *IEEE Int. Conf. Robot. Automat.* IEEE, 2014, pp. 3050–3057.

[53] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4558–4567.

[54] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3d semantic segmentation," in *Int. Conf. on Comput. Anal. Imag. Patterns.* Springer, 2017, pp. 95–107.

[55] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 9204–9214.

[56] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, no. 2, Jun 2013, p. 896.

[57] H. Lei, N. Akhtar, and A. Mian, "Octree guided cnn with spherical kernels for 3d point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 9631–9640.

[58] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point

cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 9397–9406.

[59] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Proc. Adv. Neural Inform. Process. Syst.*, vol. 31, pp. 820–830, 2018.

[60] S. Liu, X. Li, V. Jampani, S. D. Mello, and J. Kautz, "Learning propagation for arbitrarily-structured data," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 652–661.

[61] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network," in *AAAI*, vol. 33, no. 01, 2019, pp. 8778–8785.

[62] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 8895–8904.

[63] Z. Liu, X. Qi, and C.-W. Fu, "One thing one click: A self-training approach for weakly supervised 3d semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 1726–1736.

[64] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3431–3440.

[65] H. Lu, X. Chen, G. Zhang, Q. Zhou, Y. Ma, and Y. Zhao, "Scanet: Spatial-channel attention network for 3d object detection," in *ICASSP*. IEEE, 2019, pp. 1992–1996.

[66] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for

real-time object recognition," in *IEEE/RSJ Int. Conf. Intell. Robots Syst.* IEEE, 2015, pp. 922–928.

[67] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, "Vv-net: Voxel vae net with group convolutions for point cloud segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, October 2019.

[68] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019.

[69] C. Park, Y. Jeong, M. Cho, and J. Park, "Fast point transformer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 16 949–16 958.

[70] F. Poux and R. Billen, "Voxel-based 3d point cloud semantic segmentation: Unsupervised geometric and relationship featuring vs deep learning methods," *ISPRS Int. J. Geoinf.*, vol. 8, 2019.

[71] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 918–927.

[72] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 652–660.

[73] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 5648–5656.

[74] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inform. Process. Syst.*, 2017, pp. 5099–5108.

[75] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3577–3586.

[76] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Med. Image Comput. Comput. Assist. Interv.* Springer, 2015, pp. 234–241.

[77] D. Rozenberszki, O. Litany, and A. Dai, "Language-grounded indoor 3d semantic segmentation in the wild," *arXiv preprint arXiv:2204.07761*, 2022.

[78] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE Int. Conf. Robot. Automat.* Citeseer, 2009, pp. 3212–3217.

[79] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Underst.*, vol. 125, pp. 251–264, 2014.

[80] J. Sauder and B. Sievers, "Context prediction for unsupervised deep learning on point clouds," *arXiv preprint arXiv:1901.08396*, 2019.

[81] ——, "Self-supervised deep learning on point clouds by reconstructing space," in *Proc. Adv. Neural Inform. Process. Syst.*, vol. 32, 2019, pp. 12 962–12 972.

[82] X. Shen, C. Wang, C. Wen, W. Liu, X. Sun, and J. Li, "Discriminative learning of point cloud feature descriptors based on siamese network," in *Int. Geosci. Remote Sens. Symp.* IEEE, 2018, pp. 4519–4522.

[83] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Neighbors do help: Deeply exploiting local structures of point clouds," *arXiv preprint arXiv:1712.06760*, vol. 1, no. 2, 2017.

[84] C. Shi, X. Chen, K. Huang, J. Xiao, H. Lu, and C. Stachniss, "Keypoint matching for point cloud registration using multiplex dynamic graph attention networks," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 8221–8228, 2021.

[85] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2930–2937.

[86] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3693–3702.

[87] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.

[88] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi, "Discovery of latent 3d keypoints via end-to-end geometric reasoning," pp. 1–13, 2018.

[89] A. Tao, Y. Duan, Y. Wei, J. Lu, and J. Zhou, "Seggroup: Seg-level supervision for 3d instance and semantic segmentation," *IEEE Trans. Image Process.*, vol. 31, pp. 4952–4965, 2022.

[90] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. Adv. Neural Inform. Process. Syst.*, 2017, pp. 1195–1204.

[91] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 3887–3896.

[92] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *Int. Conf. 3D Vis. (3DVProc. ACM Worksh. 3D Obj. Retr.)*. IEEE, 2017, pp. 537–547.

[93] G. Te, W. Hu, A. Zheng, and Z. Guo, "Rgcnn: Regularized graph cnn for point cloud segmentation." ACM, 2018, pp. 746–754.

[94] A. Thabet, H. Alwassel, and B. Ghanem, "Self-supervised learning of local features in 3d point clouds," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2020, pp. 938–939.

[95] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.

[96] A. Thyagharajan, B. Ummenhofer, P. Laddha, O. J. Omer, and S. Subramoney, "Segment-fusion: Hierarchical context fusion for robust 3d semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 1236–1245.

[97] F. Tombari, S. Salti, and L. Di Stefano, "Unique shape context for 3d data description," in *ACM Worksh. 3D Obj. Retr.* ACM, 2010, pp. 57–62.

[98] A. Tonioni, S. Salti, F. Tombari, L. D. Stefano, and R. Spezialetti, "Learning to Detect Good 3D Keypoints," *Int. J. Comput. Vis.*, vol. 126, no. 1, pp. 1–20, 2018.

[99] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," in *Int. Conf. 3D Vis. (3DVProc. ACM Worksh. 3D Obj. Retr.)*. IEEE, 2016, pp. 323–332.

[100] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *J. Mach Learn. Res.*, vol. 9, no. 11, 2008.

[101] W. Wan, Y. Zhong, T. Li, and J. Chen, "Rethinking feature distribution for loss functions in image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 9117–9126.

[102] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 10 296–10 305.

[103] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Trans. Graph.*, vol. 36, p. 72, 2017.

[104] W. Wang, T. Zhou, F. Yu, J. Dai, E. Konukoglu, and L. Van Gool, "Exploring cross-image pixel contrast for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 7303–7313.

[105] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 7794–7803.

[106] X. Wang, J. He, and L. Ma, "Exploiting local and global structure for point cloud semantic segmentation with contextual point representations," in *Proc. Adv. Neural Inform. Process. Syst.*, 2019, pp. 4573–4583.

[107] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[108] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.*, vol. 38, pp. 1–12, 2019.

[109] J. Wei, G. Lin, K.-H. Yap, T.-Y. Hung, and L. Xie, "Multi-path region

mining for weakly supervised 3d semantic segmentation on point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 4384–4393.

[110] B. Wu, A. Wan, X. Yue, K. Keutzer, R. time Road-object Segmentation, L. Point, B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1887–1893.

[111] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *IEEE Int. Conf. Robot. Automat.* IEEE, 2019, pp. 4376–4382.

[112] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 9621–9630.

[113] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1912–1920.

[114] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1625–1632.

[115] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 574–591.

[116] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional shapecontextnet for point cloud recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018,

pp. 4606–4615.

[117] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2020, pp. 1–19.

[118] S. Xu, R. Wan, M. Ye, X. Zou, and T. Cao, "Sparse cross-scale attention network for efficient lidar panoptic segmentation," *arXiv preprint arXiv:2201.05972*, 2022.

[119] X. Xu and G. H. Lee, "Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 13 706–13 715.

[120] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 87–102.

[121] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 7652–7660, 2018.

[122] Z. Yang and L. Wang, "Learning relationships for multi-view 3d object recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 7505–7514.

[123] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang, "3d recurrent neural networks with context fusion for point cloud semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 403–417.

[124] H. Yu, F. Li, M. Saleh, B. Busam, and S. Ilic, "Cofinet: Reliable coarse-to-fine correspondences for robust pointcloud registration," *Proc. Adv. Neural Inform. Process. Syst.*, vol. 34, pp. 23 872–23 884, 2021.

[125] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 1802–1811.

[126] T. Zhang, G. Lin, J. Cai, T. Shen, C. Shen, and A. C. Kot, "Decoupled spatial neural attention for weakly supervised semantic segmentation," *IEEE Trans. Multimedia*, vol. 21, no. 11, pp. 2930–2941, 2019.

[127] Y. Zhang, Z. Li, Y. Xie, Y. Qu, C. Li, and T. Mei, "Weakly supervised semantic segmentation for large-scale point cloud," in *AAAI*, vol. 35, no. 4, 2021, pp. 3421–3429.

[128] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 16 259–16 268.

[129] H. Zhao, L. Jiang, and C.-w. F. Jiaya, "Pointweb : Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 1, 2019.

[130] L. Zhou, C. Gong, Z. Liu, and K. Fu, "Sal:selection and attention losses for weakly supervised semantic segmentation," *IEEE Trans. Multimedia*, vol. 23, pp. 1035–1048, 2021.

[131] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4490–4499.