

Technical University of Munich

TUM Department of Civil, Geo and Environmental Engineering

Chair of Computational Modeling and Simulation

# **Enhancing 3D Point Cloud Semantic Segmentation Using Multi-Modal Fusion With 2D Images**

Master Thesis

for the Master of Science Program Civil Engineering

Author: Changyu Du

Student ID:

Supervisor: Prof. Dr.-Ing. André Borrmann

M.Sc. Miguel Arturo Vega Torres

Date of Issue: 01. March 2021

Date of Submission: 30. August 2021

---

## Abstract

Deep learning algorithms can identify valid semantic information from the original 3D point cloud that can be used to create BIM models of the built environment. This is an important step in generating the digital twin of a building. Compared with traditional unimodal deep learning algorithms that directly process 3D point clouds, multimodal fusion algorithms that leverage 2D images as supplementary information for 3D scenes have greater performance advantages. It is worth noting that multimodal fusion algorithms do not exist independently but are derived from existing unimodal methods - 2D and 3D unimodal deep learning networks are chosen as the backbone networks to process the information from different modalities and fuse them at the right time.

In this study, the performance of an open-source multimodal algorithm, MVPNet, is improved on 3D semantic segmentation task by using KPConv as a more robust and stronger 3D backbone. Different modules of the two networks are meaningfully combined: the 2D-3D lifting method provided by MVPNet aggregates selected 2D multi-view images features into 3D point cloud, and then KPConv is used to fuse these features in 3D space to predict 3D semantic labels.

On a custom ScanNet dataset, the proposed network achieves a score of 74.40 mIoU on the 3D semantic segmentation task, outperforming the original MVPNet (+3.19 mIoU). In addition, rich ablation studies are designed to investigate the appropriate fusion structure, timing, and the effect of 3D color, etc.

## Zusammenfassung

Die Verwendung von Deep Learning Algorithmen zur Identifizierung gültiger semantischer Informationen aus der originalen 3D Punktwolke zur Generierung eines BIM-Modells der bebauten Umgebung ist ein wichtiges Mittel zur Erstellung eines digitalen Zwillings des Gebäudes. Im Vergleich zu traditionellen unimodalen Deep Learning Algorithmen, die 3D Punktwolken direkt verarbeiten, haben multimodale Fusionsalgorithmen, die 2D-Bilder als Zusatzinformationen für 3D Szenen nutzen, größere Leistungsvorteile. Es ist erwähnenswert, dass multimodale Fusionsalgorithmen nicht unabhängig voneinander existieren, sondern von bestehenden unimodalen Methoden abgeleitet werden - 2D und 3D unimodale Deep Learning Netzwerke werden als Backbone-Netzwerke gewählt, um die Informationen aus verschiedenen Modalitäten zu verarbeiten und zum richtigen Zeitpunkt zu fusionieren.

In dieser Studie wird die Leistung eines multimodalen Open-Source-Algorithmus, MVPNet, bei einer 3D semantischen Segmentierungsaufgabe verbessert, indem KPConv als robusteres und stärkeres 3D-Backbone eingesetzt wird. Verschiedene Module der beiden Netzwerke werden sinnvoll kombiniert: Die 2D-3D-Lifting Methode von MVPNet aggregiert ausgewählte 2D-Multiview-Bilder-Features zu einer 3D Punktwolke, und dann wird KPConv verwendet, um diese Features im 3D Raum zu fusionieren, um semantische 3D Labels vorherzusagen.

Auf einem benutzerdefinierten ScanNet Dataset erreicht das vorgestellte Netzwerk eine Score von 74,40 mIoU bei der 3D semantischen Segmentierungsaufgabe und übertrifft damit das ursprüngliche MVPNet (+3,19 mIoU). Darüber hinaus werden umfangreiche Ablationsstudien durchgeführt, um die geeignete Fusionsstruktur, das Timing und den Effekt der 3D Farbe etc. zu untersuchen.

---

## List of Contents

List of Figures	VII	
List of Tables	IX	
List of Abbreviations	X	
<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation .....	1
1.2	Research objectives.....	3
1.3	Reading guide.....	4
<b>2</b>	<b>Theoretical Basis</b>	<b>5</b>
2.1	From 2D to 3D .....	5
2.2	Fusion operations .....	6
2.2.1	Addition.....	6
2.2.2	Average mean and Maximum .....	6
2.2.3	Concatenation.....	7
2.2.4	Ensemble.....	7
2.2.5	Adaptive fusion module.....	7
2.3	Fusion timing .....	8
2.3.1	Early fusion .....	8
2.3.2	Late fusion .....	9
2.3.3	Middle fusion.....	10
2.4	Point cloud neighborhood .....	11
2.5	KD-tree .....	11
2.6	KPConv.....	12
<b>3</b>	<b>Related Work</b>	<b>16</b>
3.1	Object detection.....	16
3.1.1	Frustum-PointNet & Frustum-ConvNet & SIFR-Net.....	16
3.1.2	PointPainting.....	17
3.1.3	EPNet .....	18
3.1.4	3D-CVF (Cross-View Feature Mapping) .....	20

---

3.1.5	CLOCs .....	21
3.2	Semantic segmentation .....	23
3.2.1	Virtual Multi-view fusion .....	23
3.2.2	MVPNet (Multi-view PointNet).....	25
3.2.3	Unified point-based framework .....	26
3.3	Instance segmentation.....	27
3.3.1	3D-SIS .....	27
3.4	Summary .....	29
<b>4</b>	<b>Methodology</b> .....	<b>32</b>
4.1	Dataset .....	33
4.2	Preprocessing.....	33
4.2.1	Data Preparation.....	33
4.2.2	Point cloud sampling.....	34
4.2.3	Compute the overlap of images with point cloud .....	35
4.3	Data loading.....	36
4.3.1	The picking strategy.....	36
4.3.2	View selection.....	38
4.3.3	Data augmentation.....	41
4.3.4	Variable batch size.....	41
4.4	The Network .....	42
4.4.1	2D Network .....	45
4.4.2	Feature aggregation module.....	46
4.4.3	KP-FCNN.....	46
<b>5</b>	<b>Results and Analysis</b> .....	<b>49</b>
5.1	Network parameters.....	49
5.2	Hardware and software.....	50
5.3	Validation results.....	50
5.4	Computational time analysis .....	54
5.5	Ablation studies.....	57
5.5.1	Geometric feature .....	57
5.5.2	Fusion twice.....	58
5.5.3	3D point color.....	59
5.5.4	Number of views .....	59

---

6	Conclusions and Future Works	61
6.1	Conclusions .....	61
6.2	Discussion .....	64
6.2.1	Other possible workflows .....	64
6.2.2	Better fusion methods and better 3D networks .....	65
6.2.3	When to fuse.....	66
6.3	Contributions.....	67
6.4	Limitations and future works .....	67
	References	70
	Appendix A	77

## List of Figures

Figure 1 Early fusion.....	9
Figure 2 Late fusion.....	10
Figure 3 Middle fusion with short-cut.....	11
Figure 4 For a more straightforward demonstration, compare an image convolution to a KPConv on 2D points. (Thomas et al., 2019). .....	13
Figure 5 Illustration of the kernel points in stable dispositions. (Thomas et al., 2019). .....	13
Figure 6 Demostration of two net work architecture base on KPConv (Thomas et al., 2019).....	15
Figure 7 KPConv blocks used in KP-FCNN (Thomas et al., 2019).....	15
Figure 8 Pipline of PointPainting (Vora et al., 2020).....	18
Figure 9 Pipline of EPNet (Huang et al., 2020).....	19
Figure 10 LI-Fusion Module (Huang et al., 2020).....	20
Figure 11 Pipline of 3D-CVF (Yoo et al., 2020).....	21
Figure 12 CLOCs Fusion network achitecture (Pang et al., 2020).....	22
Figure 13 CLOCs mix the prediction results of 2D and 3D networks into a sparse tensor T, which is then fed into a convolutional network to process fused results.(Pang et al., 2020).....	22
Figure 14 Proposed virtual view selection approaches.(Kundu et al., 2020).....	24
Figure 15 Pipline of MVPNet (Jaritz et al., 2019).....	25
Figure 16 Pipline of Unified Point-based Network (Chiang et al., 2019).....	26
Figure 17 Pipline of 3D-SIS (Hou et al., 2019).....	27
Figure 18 The algorithm pipline of this study.....	32
Figure 19 Grid subsampling illustrated on 2D points.(Madali, 2021). .....	35
Figure 20 Compute the overlap of an image with scene point cloud using base point method. ....	36
Figure 21 For clarification, a 1D illustration of the spatially regular selection method with potential updates is shown (Thomas, 2020).....	38

Figure 22 (a) is a whole scene point cloud with labels. (b) is the input spherical subcloud (labels are only for better visualization). The red dots are the base points inside it. (c) is the dense point cloud formed by the projection of the 5 selected views. (d) is what the dense point cloud looks like in the full scene view. It can be seen that the selected 2D views cover the input subcloud very well, providing rich texture information..	40
Figure 23 The architecture of the 2D encoder-decoder network (Jaritz et al., 2019)	45
Figure 24 Three fusion architectures base on KP-FCNN.....	47
Figure 25 IoU, also known as Jaccard index (Wikipedia, n.d.)	50
Figure 26 Visualization results of baseline models and MV-KPConv..	54
Figure 27 Loss curves of MV-KPConv early fusion versions using rigid kernel. ....	55
Figure 28. Loss curves of MV-KPConv early fusion versions using deformable kernel. .....	55
Figure 29 Training time of the early fusion version of MV-KPConv using rigid kernel	55
Figure 30 Training time of the early fusion version of MV-KPConv using deformable kernel. ....	55
Figure 31 mIoU variation on the validation set during training for early fusion version of MV-KPConv using rigid kernel. ....	55
Figure 32 mIoU variation on the validation set during training for early fusion version of MV-KPConv using deformable kernel. ....	55
Figure 33 KPConv illustrated on 2D points. ....	58
Figure 34 Pipeline of xMUDA network (Jaritz et al., 2021)	65
Figure 35 Instead of simply joining and fusing the features of two modalities, xMUDA enables each unimodal feature to obtain complementary information from the cascaded features through an imitation mechanism (Jaritz et al., 2021). ....	66
Figure 36 Misidentification of part of the edge of a cabinet as a table	68
Figure 37 Original Mesh	68
Figure 38 Original mesh	68
Figure 39 MVPNet (counter in light blue).....	68
Figure 40 MV-KPConv (counter in light blue)	68



---

## List of Tables

Table 3.1 Summary of STOA Multi-modal Method (all the values are in percentage) “-” means missing information.....	29
Table 5.1 Semantic segmentation IoU scores on custom ScanNet dataset .....	52
Table 5.2 Runtime and model size comparison.....	56
Table 5.3 Influence of geometric features.....	58
Table 5.4 Fusion twice compared to early fusion.....	59
Table 5.5 Influence of point cloud color .....	59
Table 5.6 Impact of the number of images .....	60
Table 6.1 Visualization results of 3D semantic segmentation by MV-KPCConv on 28 validation scenarios.....	77

## List of Abbreviations

BEV	Bird's Eyes View
BIM	Building Information Modelling
CNN	Convolutional Neural Network
DT	Digital Twin
DoF	Degree of Freedom
FOV	Field of View
KNN	K-Nearest Neighborhood
MLP	Multi-Layer Perceptron
NFoV	Normal Field of View
NMS	Non-Maximum Suppression
IoU	Intersection over Union
mIoU	Mean Intersection over Union
ReLU	Rectified Linear Unit
ROI	Region of Interest
RPN	Region Proposal Network
SOTA	State of the Art

# 1 Introduction

## 1.1 Motivation

In the construction industry and related research sectors, the concept of Digital Twin (DT) is becoming increasingly essential. It is a concept associated with Industry 4.0 and aims to bring the building model, object information, and data received from sensors and actuators together as one system, generating a digital duplicate of the physical environment, states, and processes (Wahbeh et al., 2020).

In order to capture the current physical state of the built environment, 3D point clouds can be utilized to depict the precise details of the as-is physical environment (Tan Qu & Wei Sun, 2015). A Keyword here is Scan-to-BIM, which describes approaches that interpret generated point clouds and create a valid as-built BIM model from them. (Braun, 2020)

However, 3D point clouds do not provide any geometric primitives or semantic information, which makes it difficult to detect / classify objects in point cloud. For any further representations and automated assessment, the 3D point cloud needs to be processed by deep learning methods in order to obtain useful semantic data. (Stojanovic et al., 2020)

Deep learning has shown outstanding performance in a wide range of computer vision tasks, especially 2D image processing, such as object detection, semantic segmentation etc. As its name indicates, deep learning uses deep neural network and convolution operation to extract high-dimension features from training data, and this operation usually requires a structured data, as it originally designed for raster images. (Xie et al., 2020) However, different from the images, the data structure of 3D point cloud is irregular, continuous, unstructured, and unordered (Bello et al., 2020). This makes feature extractions on the point cloud challenging for traditional deep learning models. In order to solve this problem, preprocess the point cloud data before inputting it to network may need to be considered. Based on the representation of point cloud, the deep learning-based point cloud processing approaches can be briefly divided into three categories: *voxel-based*, *point-based* and *multi-view-based*.

The *voxel-based* approaches solved both unstructured and unordered problems of raw point cloud by partitions the point cloud into a fixed-resolution 3D grid, which has a discrete regular data structure. The voxelized data can be further processed by 3D convolutions, as in the case of pixels in 2D convolutions. However, the voxel-based methods require high memory consumption due to the sparsity of the voxels - voxel structures not only store occupied spaces, but also store free or unknown spaces. In addition, some spatial resolution and fine-grained 3D geometry information can also lose during voxelization. (Cui et al., 2021)

*Point-based* approaches process point cloud directly without transforming it into an intermediate data representation (Xie et al., 2020). A pioneering deep learning framework in this direction is PointNet (Qi, Su, et al., 2017). PointNet uses symmetric function (Max-pooling) to ensure the permutation invariance of input point cloud data and solved unordered problem. It also employs T-Net modules to align point clouds, to make sure that the object represented by the point cloud data is invariant to some spatial transformations, such as rotation and translation. For per-point feature extraction, it uses shared Multi-Layer Perceptron (MLP) to process individual points (Cui et al., 2021). In addition to the PointNet-based approach, some recent works also concentrated on defining specialized convolution operations for points, e.g., KPConv (Thomas et al., 2019). But applying the point-based methods directly on massive point cloud can be time-consuming and memory-expensive.

The traditional *Multi-view-based* methods try to represent the 3D point cloud by multi-view 2D images. These rendered images can then be processed by standard 2D convolutions and features from these views are then aggregated for accurate 3D scene recognition (X. Chen et al., 2017). The performance of early multi-view-based deep learning architecture was not satisfactory. The main reasons are that the approximate 2D projection leads to the limitations and loss of the geometric structure. Furthermore, because multi-view projected images must cover all spaces containing points, it is usually difficult to choose a suitable viewpoint for multi-view projection to cover all points in large and complex scenes (Xie et al., 2020).

In summary, employing these mono-modal algorithms to convert point clouds into other representations or directly process them, more or less will result in data loss or excessive computational costs. Of course, continuing to enhance the structure and perfor-

mance of such algorithms is an improvement path, but we might be able to get inspiration from the multi-view-based method and consider another direction: whether we can use mature 2D perceptron to improve the processing of 3D scenes.

If using a more accurate term to describe the commonality of the algorithms in this direction, it will be **Multi-Modality Fusion**. Broadly speaking, multimodal fusion gathers rich characteristics of complicated scenarios from various sensors and integrates them to gain more spatial and contextual information for robust, accurate, and fast scene understanding (Y. Zhang et al., 2020). Narrowly speaking, we want to employ mature 2D Convolutional Neural Network (CNN) to extract semantic features of 2D images to enrich the expression of point clouds, or obtain 2D region proposals to achieve more fast, robust and accurate results in 3D point cloud segmentation / detection tasks. It is also worth noting that deep multi-modal fusion methods do not exist independently but derive from existing mono-modal algorithms. The voxel- or point-based mono-modal approaches are chosen as the backbone network for processing data in a holistic or segregated manner (Y. Zhang et al., 2020).

## 1.2 Research objectives

The main goal of this thesis is to optimize an open-source multimodal deep learning algorithms called Multi-view-PointNet (MVPNet) to improve its metrics (mIoU, IoU) on the indoor point cloud semantic segmentation tasks.

The research aims to answer the following questions:

- Is it possible to improve MVPNet by changing its 3D network from PointNet to KPConv?
- How much better does the proposed MV-KPConv (the optimized MVPNet) perform compare to the baseline model in terms of mIoU and IoU on class doors and windows?
- Is the 2D – 3D lifting method provided by MVPNet feasible for different 2D input format?
- How do early, middle, and late fusions perform on MV-KPConv respectively? Does using two types of fusion in the network at the same time improve network performance?
- To what extent do 3D point cloud colors affect the proposed network that already fuse 2D image colors?

### 1.3 Reading guide

This thesis is structured in the following chapters:

- Chapter 2 presents the theoretical background of the methods used in this thesis. It includes the introduction to different fusion theories, as well as illustration of the principles of KPConv. This information provides useful theoretical basis for understanding the optimization proposed in this thesis.
- Chapter 3 gives an overview of state-of-art multimodal methods that use 2D images to support 3D point cloud processing. The performance of these method is summarized and their limitations are discussed.
- Chapter 4 explain the workflow of the proposed method in this research.
- Chapter 5 shows the performance of the proposed method and analyzes the method through a series of ablation studies.
- Chapter 6 answers all the research questions, discusses the findings during the research and summarizes the contributions of this thesis. The limitations of the proposed method and the possibility of its continued optimization will be provided.

## 2 Theoretical Basis

This section presents the theoretical background involved in the main methods used in this study. Starting from how to project 2D images to 3D, the different fusion operations, fusion time points, the point cloud neighbor types used in this paper and the KD tree data structure to accelerate the efficiency of neighbor search are introduced in turn. Finally, the basic principle of KPConv and the architecture of the KP-FCNN constructed based on it are illustrated.

### 2.1 From 2D to 3D

Typically, the feature map of a 2D image  $F_{2D}$  can be expressed as  $F_{2D} \in R^{C \times H \times W}$ , where  $C, H, W$  refer to the feature channels, height, and width dimensions of the feature map. However, for unstructured point cloud data, its feature map is usually represented as  $F_{3D} \in R^{C \times N}$ , where  $N$  denotes the number of points. To achieve the fusion of the two modalities, it is usually necessary to reproject the pixel-wise features into the 3D space so that they can be associated with the point-wise features. A common approach is to use depth information and camera projection matrix.

With the RGB-D camera, we can obtain the color image and its corresponding depth map. The depth map contains information about the distance of each pixel in the color image from the sensor. Let the coordinates of a pixel in the image be  $(u, v, d)$ , where  $d$  denotes the depth information, then its spatial point coordinates  $(x, y, z)$  under the camera coordinate system can be obtained by the following equation:

$$\begin{aligned} x &= \frac{(u - c_x) \cdot z}{f_x} \\ y &= \frac{(v - c_y) \cdot z}{f_y} \\ z &= \frac{d}{s} \end{aligned}$$

where  $f_x, f_y$  refers to the focal length of the camera on the  $x, y$  axes,  $c_x, c_y$  refers to the center of the camera sensor, and  $s$  refers to the scaling factor of the depth map. Usually, the four parameters  $f_x, f_y, c_x, c_y$  will be defined as the intrinsic matrix  $K$  of the camera:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix}$$

Finally, by adding the camera pose information the spatial point coordinates  $(x, y, z)$  in the world coordinate system can be obtained:

$$s \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot \left( R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + t \right)$$

where  $R$  and  $t$  are the camera pose.  $R$  represents the rotation matrix and  $t$  represents the translation vector, which together are the extrinsic matrix of the camera. The extrinsic matrix and the intrinsic matrix make up the camera projection matrix (D. Zhang, 2011).

Based on this simple matrix model, the coordinates of each pixel can be reprojected into the world coordinate system, enabling the alignment of 2D and 3D information.

## 2.2 Fusion operations

One of the core aspects of multimodal fusion is how to effectively fuse the feature maps of multiple modalities together through mathematical operations in a deep learning network. Denote  $M_1$  and  $M_2$  as two different modalities, their corresponding feature maps are  $f^{M_1}$  and  $f^{M_2}$ . Some typical fusion operations are summarized below (Feng et al., 2021).

### 2.2.1 Addition

The fused feature  $f^{MM}$  can be obtained by summing up the two feature maps element-wise. It is worth noting that before performing the summation operation, it is necessary to ensure that the feature maps of the two modalities have the same shape. Because in linear algebra, two matrices must have an equal number of rows and columns to be added.

$$f^{MM} = f^{M_1} + f^{M_2}$$

### 2.2.2 Average mean and Maximum

Similar to addition, feature maps can be fused by calculating element-wise averages or selecting the largest value of corresponding feature maps of all the modalities at each axis.



$$f^{MM} = \text{mean}(f^{M_1}, f^{M_2})$$

$$f^{MM} = \text{max}(f^{M_1}, f^{M_2})$$

### 2.2.3 Concatenation

Concatenation is a relatively common fusion operation. Instead of requiring the feature maps of the two modalities to have the same shape, as in the previously mentioned operations, the axes that perform the concatenation operation are allowed to have different lengths. This provides a great convenience. For example, when there is a 64-channel feature map  $f_i \in R^{64 \times N}$  and a 128-channel feature map  $f_j \in R^{128 \times N}$ , they can be fused into a 192-channel feature map  $f_k \in R^{192 \times N}$  by a simple connection operation. However, if addition or averaging operation is used here for fusion, it may be necessary to first perform a linear transformation of one feature map to match the shape of another feature map before performing the subsequent fusion operation.

$$f^{MM} = f^{M_1} \oplus f^{M_2}$$

### 2.2.4 Ensemble

This operation ensembles features from different sensing modalities. Ensembles are often used to fuse Regions of Interests (ROIs) in object detection networks (Feng et al., 2021). A classic example is Frustum PointNets (Qi et al., 2018). Frustum PointNets uses a pre-trained image detector to construct 2D bounding boxes that build frustums in 3D point clouds. Then, the point clouds within multiple frustums will be used for 3D objects detection.

$$f^{MM} = f^{M_1} \cup f^{M_2}$$

### 2.2.5 Adaptive fusion module

Although deep learning-based algorithms automatically learn representative features, multimodal input is likely to be poor in many circumstances. Multimodal data's redundancy, imbalance, uncertainty, and even contradiction may have a substantial impact on the model's performance. Simple fusion techniques like summation and concatenation mentioned above only go so far in assisting in the generation of optimal joint feature representations (Y. Zhang et al., 2020). To solve this problem, some adaptive fusion modules are designed to learn to explicitly weight feature maps from different modalities. Specific gating units that assign class- or modality-wise weights are frequently included in such fusion approaches. Much of the latest research has focused

on this area. Xu et al. propose an adaptive attention module for 3D object detection networks to fuse 2D and 3D sensors at the semantic level (Xu et al., 2021). The input points and 2D/3D semantic predictions are first utilized to learn an attention mask through PointNet-like module (Qi, Su, et al., 2017). The attention mask will then be used to adaptively fuse information from 2D and 3D semantic labels. Dai et al. also propose an attentional feature fusion approach that is suitable for most common scenarios of 2D image fusion (Y. Dai et al., 2021). In summary, the features of the two modalities are averaged with the learned weights  $w^{M1}$  and  $w^{M2}$  by the adaptive fusion module  $A$ , and then the combined feature map  $f^{MM}$  is output by a simple fusion operation such as concatenation:

$$f^{MM} = A(w^{M1} \cdot f^{M1} \oplus w^{M2} \cdot f^{M2}), \quad \text{with } w^{M1} + w^{M2} = 1$$

## 2.3 Fusion timing

Fusion architectures can be classified based on when information from multiple modalities are integrated during processing. Three general categories are early fusion, middle fusion and late fusion.

### 2.3.1 Early fusion

Early fusion refers to the integration of monomodal feature sets before learning concepts (Snoek et al., 2005). This operation learns the joint features of multiple modalities at an early stage, fully exploiting the information of the raw data (Cui et al., 2021). Early fusion networks normally involve relatively minimal memory and computational complexities since this process allows robust cross-modal information interaction and applies only one single stream for learning purposes (L. He, 2019). However, this comes at the price of model rigidity. When an input is replaced with a new sensing modality or the input channels are extended, for example, the early fused network must be entirely retrained. Early fusion is also sensitive to spatial-temporal data misalignment among sensors, which can be caused by calibration errors, differing sampling rates, and so on. Passing all modalities through the same network does not always address the fundamental data disparities between them (Feng et al., 2021).

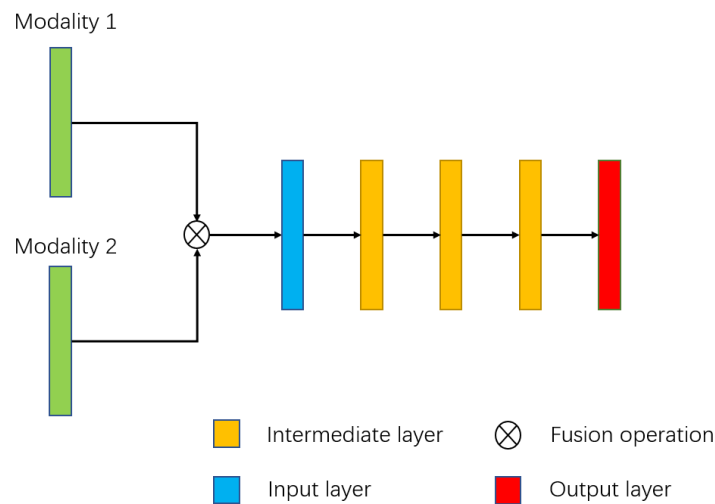


Figure 1 Early fusion

### 2.3.2 Late fusion

Late fusion processes each modality on a separate path and fuses the outputs in the decision / result level. Because multiple single paths can use different architectures and configurations, this approach allows for more modeling flexibility. Single modality algorithms can be trained with data from their own sensors. As a result, there is no requirement to synchronize or align multimodal data with other modalities. Only the final fusion step necessitates data that has been simultaneously aligned and tagged. Late fusion approaches provide greater scalability, making it easier to add or delete modalities (Atrey et al., 2010). When a new sensory modality is introduced, just its domain-specific network needs to be trained, leaving existing networks unaffected. It does, however, have substantial compute costs and memory requirements. Furthermore, it discards a wealth of intermediate features that could be quite useful when fusing (Feng et al., 2021).

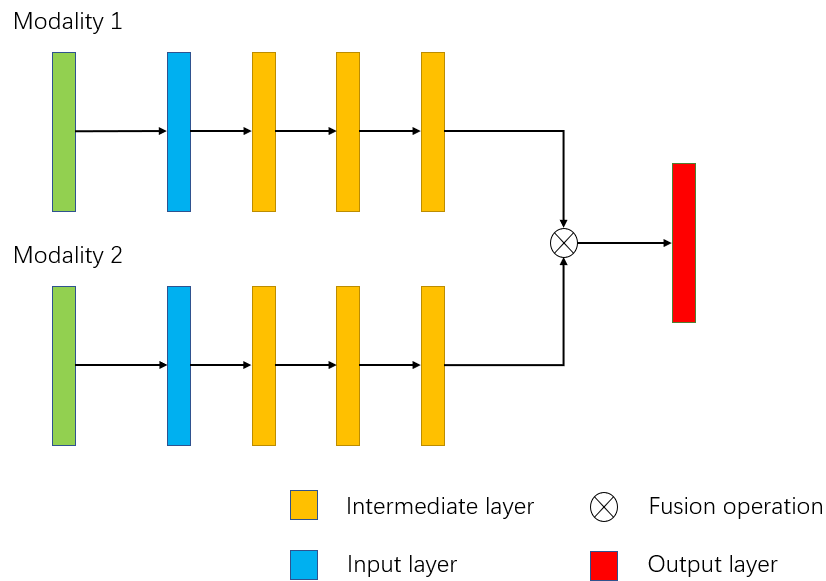
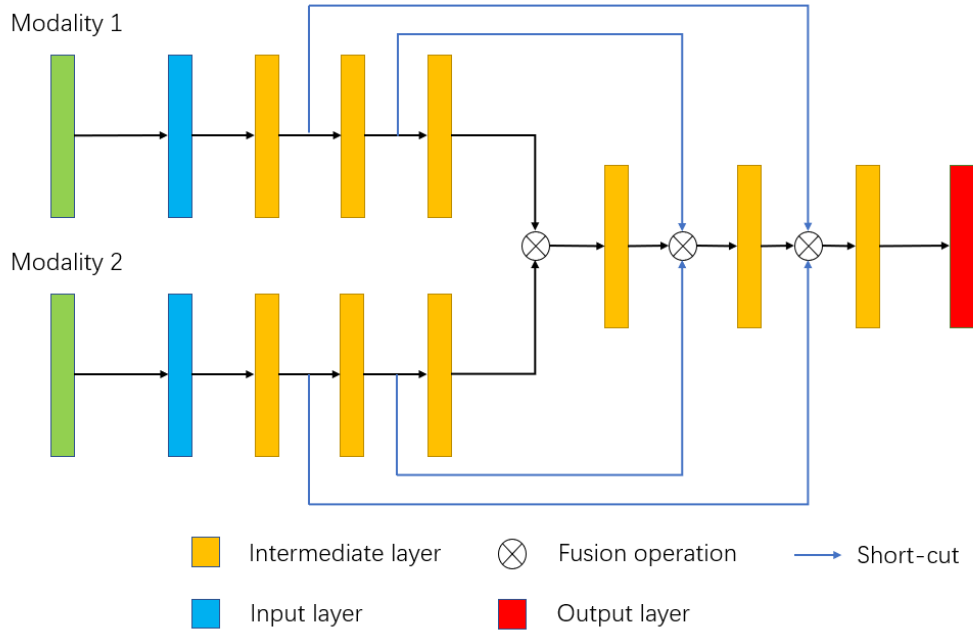


Figure 2 Late fusion

### 2.3.3 Middle fusion

The term "Middle fusion" refers to a middle ground between early and late fusion. At the intermediate layers, it mixes feature representations from several perceptual modalities. This enables the network to learn across modalities at different depths using multiple feature representations. This is the most complicated fusion method. It's also difficult to tell whether the added complexity results in meaningful improvements: given the specific network structure, finding the optimal way to fuse the middle layers is not easy (Pang et al., 2020).



*Figure 3 Middle fusion with short-cut*

## 2.4 Point cloud neighborhood

When working with 3D point clouds, searching the neighbors of a point is a frequently used method. In this study, the spherical neighborhoods and K nearest neighborhoods (KNN) are used.

The spherical neighborhood refers to the points  $x_i$  contained within a sphere centered at one point  $x_0$  and with radius  $r \in R$ . Note both  $x_i$  and  $x_0$  belong to point cloud  $P$ .

$$N_{radius}(x_0, P, r) = \{x_i \in P \mid \|x_0 - x_i\| \leq r\}$$

The KNN of point  $x_0$  are the  $K$  points that are the closest to  $x_0$ .

$$N_{KNN}(x_0, P, K) = \{x_i \in P \mid \|x_0 - x_i\| \leq \|x_0 - x_{i+1}\|, i < K\}$$

Each type has its own characteristics. Spherical neighborhoods have a fixed geometric volume that is determined by a radius, whereas KNNs can be found anywhere in space. However, KNN can have a fixed number of neighborhoods, but the number of neighbors in spherical neighborhoods can be variable (Thomas, 2020).

## 2.5 KD-tree

In order to reduce the complexity of the neighbor search, KD-tree will be used as an acceleration data structure to enhance efficiency.

The KD-Tree is a binary tree structure which always divides the data along perpendicular hyperplanes to the axes. In 2D, the method begins by locating a midpoint along the horizontal axis and then splits the space into two halves perpendicular to the axis at this midpoint. The center point along the vertical axis is determined for each subdivision. Subdivisions are then split into halves at that midpoint by a line perpendicular to the vertical axis. The preceding procedures are continued until the partition space can no longer be split further (Vega Torres et al., 2020). Because partitioning is done simply along the data axes, the construction of a KD-tree is very rapid. Once constructed, the nearest neighbor of a query point can be identified using only  $O[\log(N)]$  distance computations (Bentley, 1975). Many highly optimized open-source KD-tree implementations are available. In this study, the Scikit-learn library (Pedregosa et al., 2011) was used.

## 2.6 KPConv

Kernel Point Convolution (KPConv) is a novel point convolution operator inspired by image-based convolution, but instead of kernel pixels, a set of kernel points are utilized to define the area where each kernel weight is applied. Let  $x_i$  and  $f_i$  be the points from point cloud  $P \in R^{N \times 3}$  and their corresponding features from  $F \in R^{N \times D}$ . The kernel point convolution of  $F$  by a kernel  $g$  at a point  $x \in R^3$  can be defined as:

$$(F * g)(x) = \sum_{x_i \in N_x} g(x_i - x) f_i$$

$$\text{with } N_x = \{x_i \in P \mid \|x_i - x\| \leq r\}$$

Where  $N_x$  are the radius neighbor points of  $x$  (Thomas et al., 2019). The kernel function  $g$  takes  $x_i - x$  as input, which are the neighbors positions centered on  $x$ . For sake of clarity, we use  $y_i$  to represent these positions. Since spherical neighborhood is used,  $y_i$  are always in the sphere  $S_r^3 = \{y \in R^3 \mid \|y\| \leq r\}$ . This  $S_r^3$  is thus the definition domain of  $g$ . Similar to image convolution kernels, the  $g$  is required to apply different weights to different areas inside this domain. The Kernel points are used to define these areas in 3D space. Let  $\{\tilde{x}_k \mid k < K\}$  be the  $K$  kernel points and  $\{W_k \mid k < K\}$  be the related weight matrices that transfer features from dimension  $D_{in}$  to  $D_{out}$ . The kernel function  $g$  for  $\forall y_i \in S_r^3$  can be defined as:

$$g(y_i) = \sum_{k < K} h(y_i, \tilde{x}_k) W_k, \quad \forall k < K, \tilde{x}_k \in S_r^3, W_k \in R^{D_{in} \times D_{out}}$$

Where  $h$  is the linear correlation between  $\tilde{x}_k$  and  $y_i$ :

$$h(y_i, \tilde{x}_k) = \max(0, 1 - \frac{\|y_i - \tilde{x}_k\|}{\sigma})$$

where  $\sigma$  is the influence distance of the kernel points, which is relevant to the input density (Thomas et al., 2019).

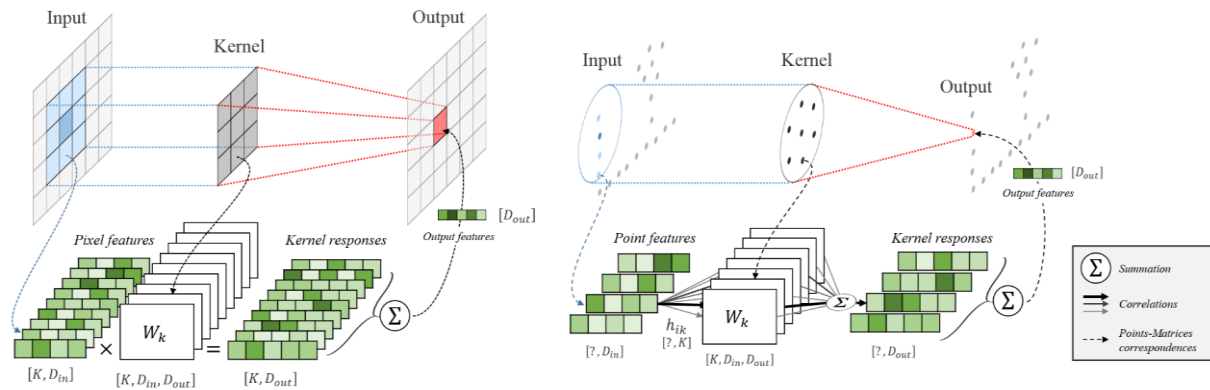


Figure 4 For a more straightforward demonstration, compare an image convolution to a KPConv on 2D points. Each pixel feature vector in the image is multiplied by a weight matrix  $W_k$  assigned by the kernel's alignment with the image. In KPConv, input points are not aligned with kernel points, and the number of input points can vary. As a result, each point feature is multiplied by all the kernel weight matrices, with a correlation coefficient  $h_{ik}$  based on its position relative to the kernel points (Thomas et al., 2019).

The kernel points required for the convolution operation defined above need to be regularly placed in the spherical domain. The distribution of kernel points within the sphere can be defined in advance, i.e., the rigid kernel. Figure 5 shows several rigid kernel variants with  $K$  points.

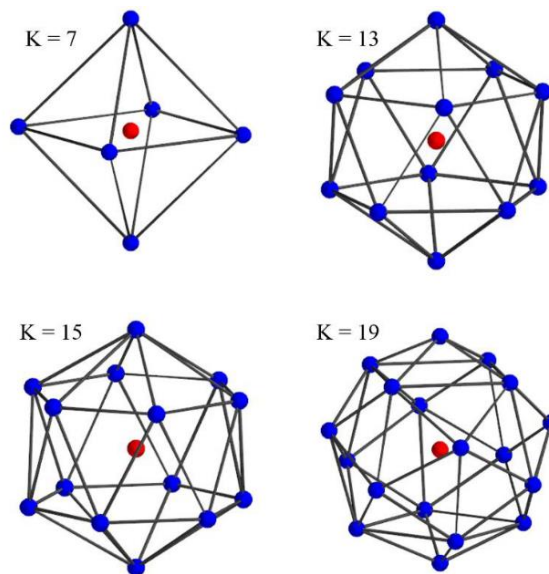


Figure 5 Illustration of the kernel points in stable dispositions. The red dot is a Kernel point that is fixed at the center of the sphere. The blue dots are the remaining kernel points that form a regular polyhedron inside the sphere (Thomas et al., 2019).

When given a large enough  $K$  to cover the spherical domain, the rigid version of KPConv is already very efficient. However, because these locations are continuous in space, the kernel point position can be learned by network to adapt local geometry of point cloud. By learning a set of  $K$  shifts  $\Delta(x)$  for every convolution location  $x \in R^3$ , the deformable KPConv can be defined as (Thomas et al., 2019):

$$(F * g)(x) = \sum_{x_i \in N_x} g(x_i - x, \Delta(x)) f_i$$

$$g_{deform}(y_i) = \sum_{k < K} h(y_i, \tilde{x}_k + \Delta_k(x)) W_k$$

In order to generate sensible offsets, additional “fitting” regularization loss and “repulsive” regularization loss are defined for the deformable kernel. For rigid kernel, a cross-entropy loss is used.

Base on the KPConv operator, KP-CNN and KP-FCNN are designed for the classification and the segmentation tasks (see Figure 6). KP-FCNN is composed of encoder and decoder. The encoder part has five layers. Each layer contains a strided KPConv block and followed by a standard KPConv block, while the first layer consists of two standard KPConv blocks. These convolutional blocks are designed like bottleneck ResNet blocks (K. He et al., 2016). Instead of the image convolution, KPConv is used here followed by batch normalization and leaky ReLU activation (see Figure 7).

The decoder part employs nearest upsampling to get the final point-wise features. Skip connections are used to transport the features between intermediate layers of the encoder and the decoder. These features are concatenated to the upsampled features and processed by a unary convolution, which is the equivalent of a  $1 \times 1$  convolution in image. The segmentation head of the network is two fully connected layers followed by softmax to predict the semantic label of each point. As a next-generation point-based approach, KP-FCNN demonstrates strong performance and outperforms Point-Net (Qi, Yi, et al., 2017) on multiple datasets.



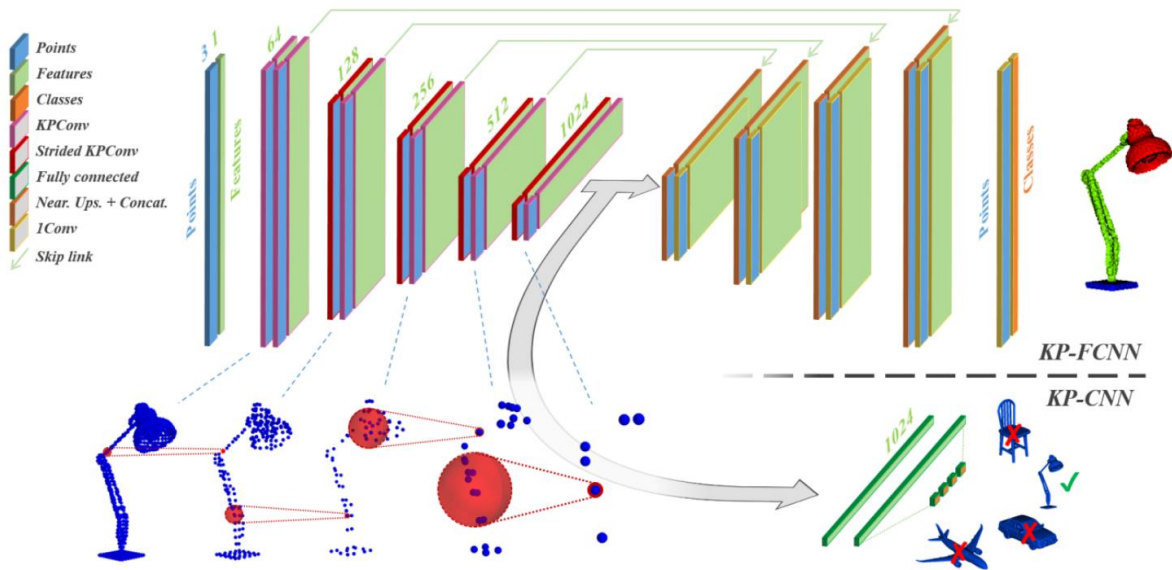


Figure 6 Demonstration of two net work architecture base on KPConv (Thomas et al., 2019)

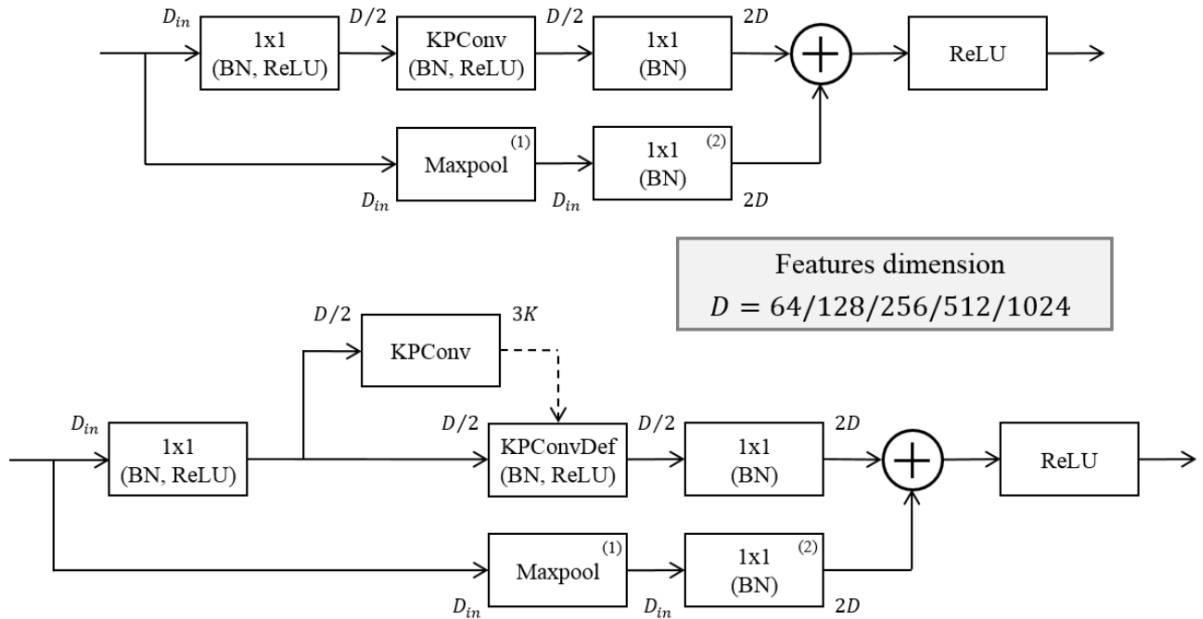


Figure 7 KPConv blocks used in KP-FCNN. Above is the rigid KPConv and below is the deformable version. Optional blocks: shortcut max pooling(1) is only needed for strided KPConv, and shortcut 1x1 convolution(2) is only needed when  $D_{in} \neq 2D$ .  $D$  are the green number above layers in figure 6. The shortcut feature and main feature are summed before output. (Thomas et al., 2019)

## 3 Related Work

In this section, the state of the art (SOTA) multimodal algorithms using both 3D point clouds and 2D images will be presented. Not only in the field of point cloud semantic segmentation, multimodal algorithms also show great potential for object detection and instance segmentation tasks. At the end of this section, their characteristics as well as performance on various datasets will be compared and summarized in a table.

### 3.1 Object detection

The goal of the object detection task is to classify and localize one or more objects in an image or point cloud. Its output are usually one or more bounding boxes for localization of objects (e.g., defined by a point, width, and height), and a class label for each bounding box.

#### 3.1.1 Frustum-PointNet & Frustum-ConvNet & SIFR-Net

The current multi-modal fusion method for 3D Object detection can start from the earliest **Frustum-PointNet** (Qi et al., 2018), which is a typical result level fusion method. The idea behind result level fusion is to limit the 3D search space for 3D object detection by using the results of off-the-shelf 2D object detectors, which reduces computation and improves run time (Cui et al., 2021). The detailed process is as follows: First, a 2D object detector is used to predict the 2D bounding box of the interested objects in the image and determine their categories. The 2D region proposal is then projected into the 3D point cloud through the calibrated camera projection matrix and thus become 3D frustum proposals. Rotate the point cloud data in the frustum so that the coordinate axis is the center viewing angle. PointNet (or PointNet++) is then used for instance segmentation on the converted point cloud data. The result of instance segmentation is used as a mask to obtain all point clouds belonging to an instance - this operation filters the background or noise points. The centroid of obtained point cloud will be calculated as the origin of the new coordinate system. Regression is performed by a T-Net module to get the residual of the target centroid and the current coordinate origin, and then the point cloud is translated. After the point cloud is translated to the calculated target centroid, the center, size and orientation of the 3D bounding box are regressed through PointNet (or PointNet++) to obtain the final output.

Wang et al. made some improvements based on the Frustum-PointNet framework and proposed a **Frustum-ConvNet** (Wang & Jia, 2019). Instead of one frustum for each object, they generated a sequence of frustums along the frustum axis for each 2D region and applied PointNet to extract features for each frustum. The frustum-level features were reformed to generate a 2D feature map, which was then fed into a fully convolutional network for 3D box estimation (Lu & Shi, 2020).

Another improvement comes from **SIFR-Net** (Zhao et al., 2019) proposed by Zhao et al.. The highlight of SFIR-Net is, they not only used the 2D bounding box generated to reduce the processing area of the point cloud, but also used 2D CNN to extract the features (color information) of the image and input it into the Point-UNet module (a 3D instance segmentation network), where the image features are concatenated with the point cloud information to enhance the performance. It can be said that SIFR-Net has initially realized both feature level and result level fusion. This method achieves significant improvement on both indoor and outdoor datasets as compared to Frustum-PointNets.

In summary, the main advantage of Frustum-based method is that the 3D search space is limited using 2D results to reduce computational cost. But due to sequential result-level fusion, the overall performance is limited by the image detector: when the 2D detector fails, the solution will fail. The redundant information from image is also not fully leveraged. In addition, SIFR-Net performance better on indoor dataset as compared to Frustum-PointNets and Frustum-ConvNet.

### 3.1.2 PointPainting

**PointPainting** (Vora et al., 2020) uses 2D semantic prediction results to do the per-point fusion. They first apply semantic segmentation on image to obtain 2D semantic maps with segmentation scores. Then the lidar points are transformed by a homogeneous transformation followed by a projection into the semantic maps. Once the lidar points are projected into the image, the segmentation scores for the relevant pixel are appended to the point cloud as additional channels to create the “painted” point cloud. The authors argued that the fused point clouds can be consumed by any point cloud network that learns an encoder, since PointPainting just changes the input dimension of the lidar points. To demonstrate this flexibility, the fused point cloud is fed into multiple existing point cloud detectors, such as PointRCNN (Shi et al., 2019), VoxelNet

(Zhou & Tuzel, 2018) and PointPillar (Lang et al., 2019) to achieve significant performance improvement.

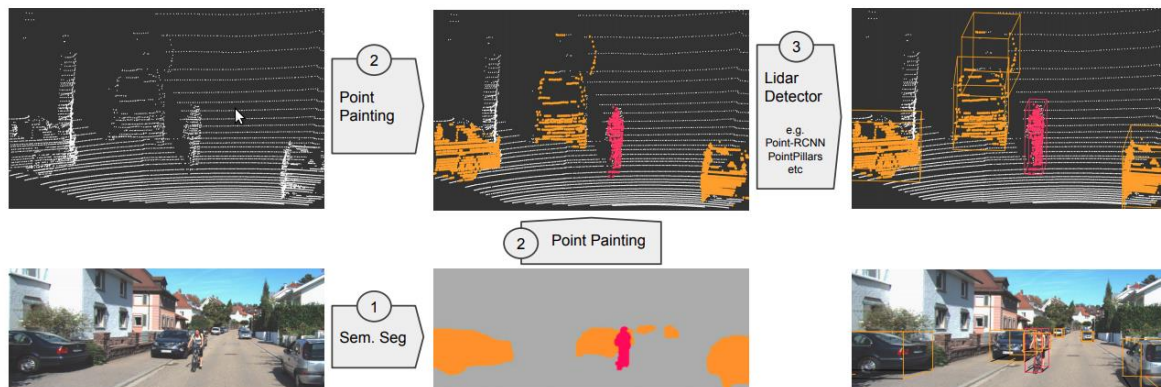


Figure 8 Pipeline of PointPainting (Vora et al., 2020)

The novelty of PointPainting is that it fuses high-level image semantics to points rather than directly appends RGB information to points. Adding RGB information to points directly results in the loss of the majority of texture information, leaving the fusion useless - only a small percentage of pixels have corresponding points due to the resolution mismatch between dense RGB and sparse depth. (Cui et al., 2021)

However, the PointPainting approach makes the image and LiDAR models highly coupled. This requires the LiDAR model to be retrained when the image model changes, which reduces the overall reliability and increases the development cost. Furthermore, 3D search space is also not limited which leads to high computational cost. (Guo et al., 2020)

### 3.1.3 EPNet

The **EPNet** (Huang et al., 2020) is composed of a geometric stream and an image stream, which produce the point features and image semantic features, respectively. The LiDAR-guided Image Fusion (LI-Fusion) modules are designed to enhance the point features with corresponding image semantic features in different scales, leading to more discriminative feature representations.

Image Stream uses camera images as input and uses a series of convolution operations to extract semantic information. A simple network structure is adopted, consisting of four lightweight convolutional blocks, each convolutional block is composed of two 3x3 convolutional layers, a batch normalization layer and a ReLU activation function.

Geometric Stream uses lidar point clouds as input, and contains 4 pairs of Set Abstraction (SA) and Feature Propagation (FP) layers to extract features. Both SA and FP layers come from Pointnet++, as mentioned in chapter 2.4. In simple terms, SA is used for downsampling and FP is used for upsampling.

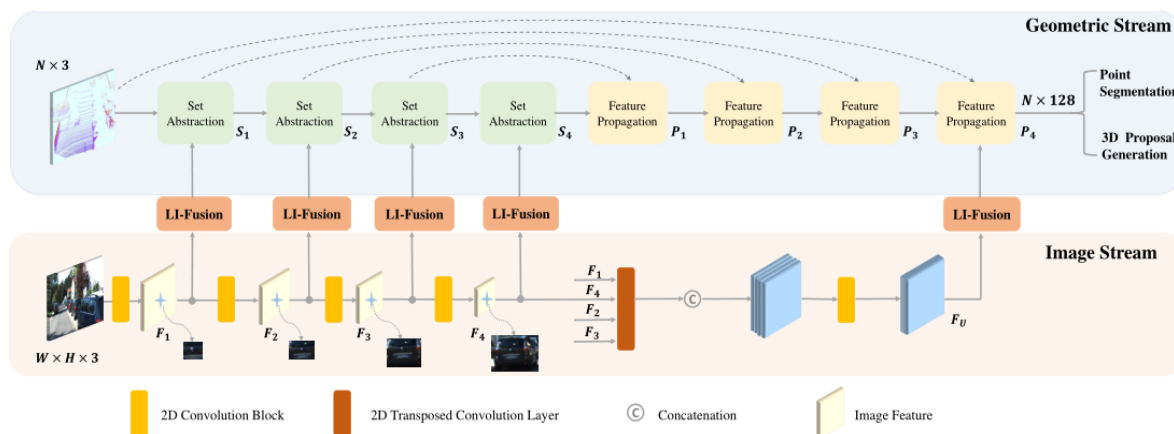


Figure 9 Pipeline of EPNet (Huang et al., 2020)

The highlight of this network is the proposed LiDAR-guided Image Fusion (LI-Fusion) module, which adaptively fuses the features from both modalities. It consists of a grid generator, an image sampler, and a LI-Fusion layer. The grid generator establishes the correspondence between the point cloud and the camera image in a point-pixel manner. The image sampler is used to get the semantic feature for each point from its corresponding pixel, and output the point-wise image feature. LI-Fusion layer then utilizes the LiDAR feature to adaptively estimate the importance of the image feature in a point-wise manner. In detail, they train the network to learn a weight map  $w$ , then concatenate the LiDAR feature with the semantic image feature multiple weights. In doing so, useful image features can be used to enhance the features of the points, while suppressing the features that interfere with the image. EPNet shows good performance in indoor dataset. It is free of image annotations, namely 2D bounding box annotations, as opposed to Frustum-PointNet.

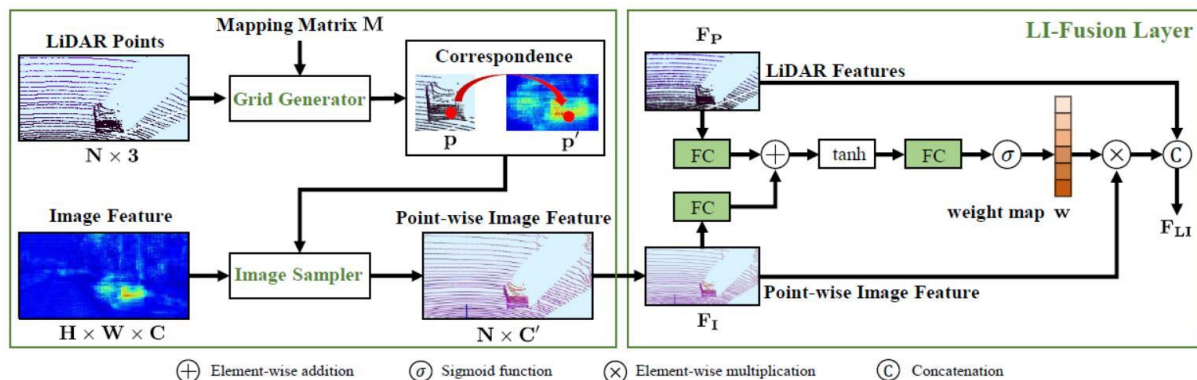


Figure 10 LI-Fusion Module (Huang et al., 2020)

### 3.1.4 3D-CVF (Cross-View Feature Mapping)

**3D-CVF** (Yoo et al., 2020) followed the basic idea of *multi-view-based* method that mentioned in chapter 1.1 and proposed a novel fusion method. The network consists of following parts: *The LiDAR pipeline* uses voxel-based backbone network to voxelize the point cloud and pass it through six 3D sparse convolution layers to get the LiDAR feature map of 128 channels in the BEV (Bird’s eyes view) domain. In parallel to the LiDAR pipeline, the multi-view camera images are processed by the CNN backbone network in *camera pipeline*. The camera features are fed into *cross-view feature (CVF) mapping module*, which use the *auto-calibrated projection module* to transform the camera-view features into the Bird’s eyes view (BEV) features (represent the feature map of image information on lidar-voxel, in order to align camera and lidar). The *adaptive gated fusion network* can selectively fuse the camera and LiDAR features depending on the relevance to the object detection task, i.e., calculate the respective weights of camera and LiDAR features and then concatenate the features. Proposals are generated from the *joint camera-LiDAR feature map* obtained by the above fusion. The author argue that the joint camera-LiDAR feature map does not have sufficient spatial information, so they further use PointNet to extract the camera features and point cloud features separately, then use the *3D-Roi-based fusion network* module to fuse them with the join camera-LiDAR feature map. This fused feature is finally used to produce the final detection results.

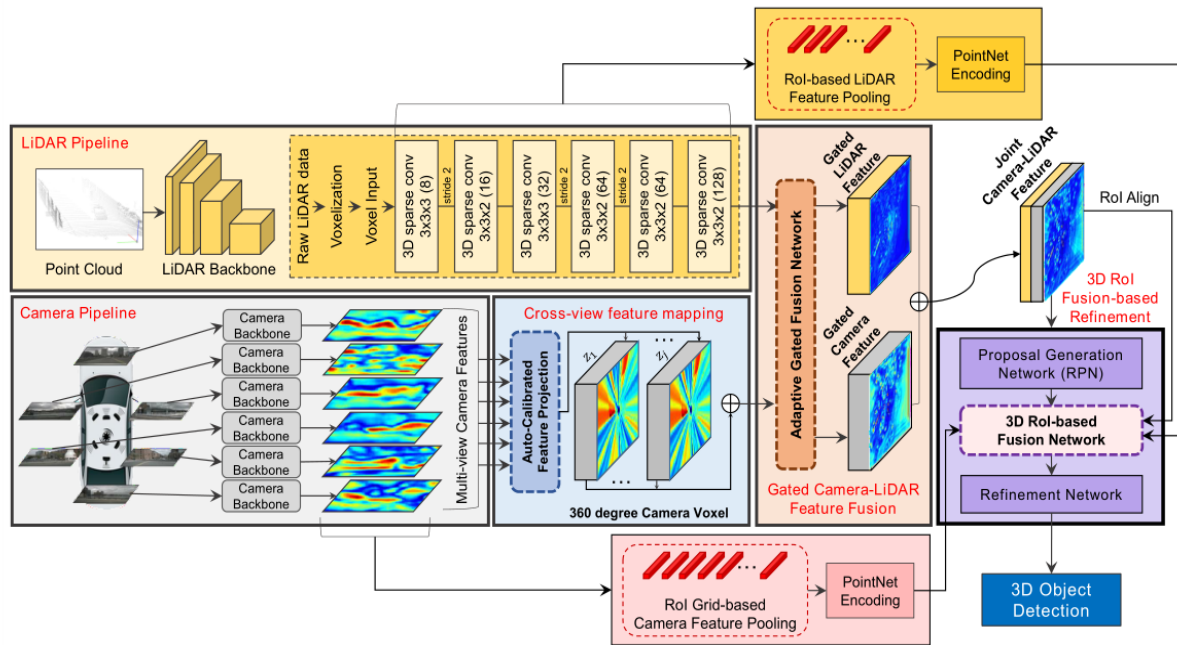


Figure 11 Pipeline of 3D-CVF (Yoo et al., 2020)

Unfortunately, this method was not tested on the indoor dataset. The biggest difference between this approach and the previous multi-modal networks is the fusion method. Previously, pixel-to-point fusion was used (e.g., EPNet, PointPainting etc.). This paper first converts the image information to the BEV (Bird’s eyes view) to obtains the representation of the camera information on the voxel, then fuse camera-voxel with point cloud-voxel. In addition, the author did a total of two fusions, which further improved the accuracy.

### 3.1.5 CLOCs

**CLOCs** (Pang et al., 2020) is a novel Camera-LiDAR Object Candidates fusion network. It uses fusion at the decision-level, i.e., late-fusion. The advantage of this fusion is that the two modal network structures do not interfere with each other and can be trained and combined independently, i.e., it has low-complexity and very good flexibility. However, this fusion strategy has also certain shortcomings as mentioned in section 2.3.2.

The pipeline of CLOCs is: First, any 2D and 3D detectors generate  $k$  2D detection candidates and  $n$  3D detection candidates (2D, 3D box); Then CLOCs takes the 2D and 3D detection candidates and do the fusion, the final output from CLOCs would be  $n$  fused confidence scores for  $n$  3D detection candidates; Finally, replace the old confidence scores (from the 3D detector) with the new fused confidence scores from

CLOCs for post processing stage. These  $n$  3D detection candidates with the corresponding CLOCs fused confidence scores are treated as the input for 3D detector post-processing functions (Non-Maximum Suppression (NMS) or other filtering) to generate final predictions.

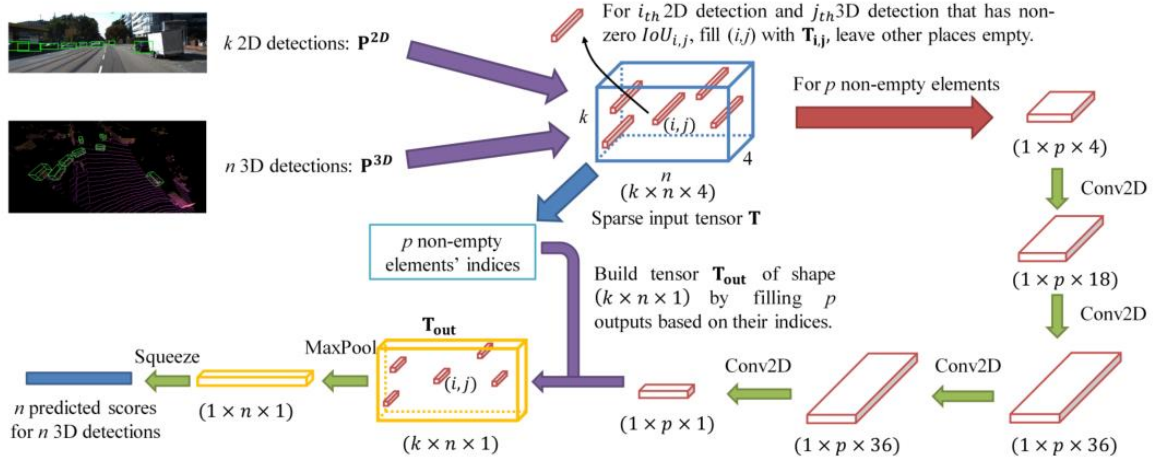


Figure 12 CLOCs Fusion network achitecture (Pang et al., 2020)

The general output of a 2D object detector is a set of 2D bounding boxes in the image plane and corresponding confident scores (see the Figure 13). The  $k$  2D detection candidates in one image can be defined as a  $\mathbf{P}^{2D}$  set. For  $i_{th}$  detection  $\mathbf{P}_i^{2D}$ ,  $x_{i1}$ ,  $y_{i1}$  and  $x_{i2}$ ,  $y_{i2}$  are the pixel coordinates of the top left and bottom right corner points from the 2D bounding box. The  $s_i^{2D}$  is the 2D confident score. The output of 3D object detectors are also 3D oriented bounding boxes in LiDAR coordinate and confident scores.  $\mathbf{P}^{3D}$  is the set of all  $n$  3D detection candidates in one LiDAR scan. For  $i_{th}$  detection  $\mathbf{P}_i^{3D}$ ,  $[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i]$  is the 7-digit vector for 3D bounding box.  $s_i^{3D}$  is the 3D confident score. Note that the author takes 2D and 3D detection results without doing NMS.

$$\begin{aligned}
 \mathbf{P}^{2D} &= \{\mathbf{p}_1^{2D}, \mathbf{p}_2^{2D}, \dots, \mathbf{p}_k^{2D}\}, \\
 \mathbf{p}_i^{2D} &= \{[x_{i1}, y_{i1}, x_{i2}, y_{i2}], s_i^{2D}\} \\
 &\text{2D detection result} \quad \xrightarrow{\text{fill in the sparse input tensor}} \quad \mathbf{T}_{i,j} = \{IoU_{i,j}, s_i^{2D}, s_j^{3D}, d_j\} \\
 \\
 \mathbf{P}^{3D} &= \{\mathbf{p}_1^{3D}, \mathbf{p}_2^{3D}, \dots, \mathbf{p}_n^{3D}\}, \\
 \mathbf{p}_i^{3D} &= \{[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i], s_i^{3D}\} \\
 &\text{3D detection result}
 \end{aligned}$$

Figure 13 CLOCs mix the prediction results of 2D and 3D networks into a sparse tensor  $T$ , which is then fed into a convolutional network to process fused results.(Pang et al., 2020)



For  $k$  2D detections and  $n$  3D detections, the author builds a  $k \times n \times 4$  tensor  $T$ , where  $IoU_{i,j}$  is the  $IoU$  between  $i_{th}$  2D detection and  $j_{th}$  projected 3D detection,  $s_i^{2D}$  and  $s_j^{3D}$  are the confident scores for  $i_{th}$  2D detection and  $j_{th}$  3D detection respectively. The  $d_j$  represents the normalized distance between the  $j_{th}$  3D bounding box and the LiDAR in XY plane (ground). The elements  $T_{i,j}$  with zero  $IoU$  are eliminated. In this way, the 2D and 3D results can be expressed as a four-dimensional tensor of coefficients, which can then be directly input to the convolutional network for fusion.

What this paper proposes is a novel and flexible fusion method, not a certain network structure, so there are many possibilities for the choice of 2D detectors and 3D detectors. The author did an experiment with PV-RCNN (Shi et al., 2020) + Cascade RCNN (Cai & Vasconcelos, 2018) on KITTI dataset and got quite good results.

## 3.2 Semantic segmentation

The goal of semantic segmentation is to predict a semantic label for each point in the input point cloud, or for each pixel in the input image. It can be said that the semantic segmentation detects the objects at the pixel / point level.

### 3.2.1 Virtual Multi-view fusion

As mentioned in introduction, the classic *multi-view-based* method cannot reach state of the art performance on the standard 3D segmentation benchmark due to the challenges of occlusion, illumination variations, difficulty choosing an appropriate viewpoint as well as camera posture misalignment in the RGB-D scan dataset (Kundu et al., 2020). However, **Virtual MV-Fusion** (Kundu et al., 2020) proposed a new view-based 3D semantic segmentation method, which overcomes these problems and achieved a high ranking on the ScanNet (A. Dai et al., 2017) benchmark. Its central concept is to employ synthetic images generated from a "virtual view" of a 3D scene rather than processing raw photographic images captured by a physical camera.

Authors first choose virtual viewpoints for each 3D scene throughout the training stage. They select camera intrinsic, camera extrinsic, which channels to render, and rendering parameters (e.g., depth range, backface culling) for each virtual view. It should be noted that the virtual view enables for the free selection of camera parameters that are most effective for 2D semantic segmentation tasks. For example, as for camera intrinsic, they use a pinhole camera model with significantly higher field of view (FOV) than the original cameras, providing larger context that leads to more accurate 2D semantic

segmentation. As for camera extrinsic, they use a mixture of the different sampling strategies to generate many novel views. And through channel rendering they let 2D views to capture additional channels that are difficult to capture with real cameras, e.g., normal and coordinates. They turn on backface culling in the rendering so that the backface do not block the camera views. This allows to select views from outside a room (behind a wall) to see more context from the view that are not physically possible with real cameras. It can potentially improve model performance.

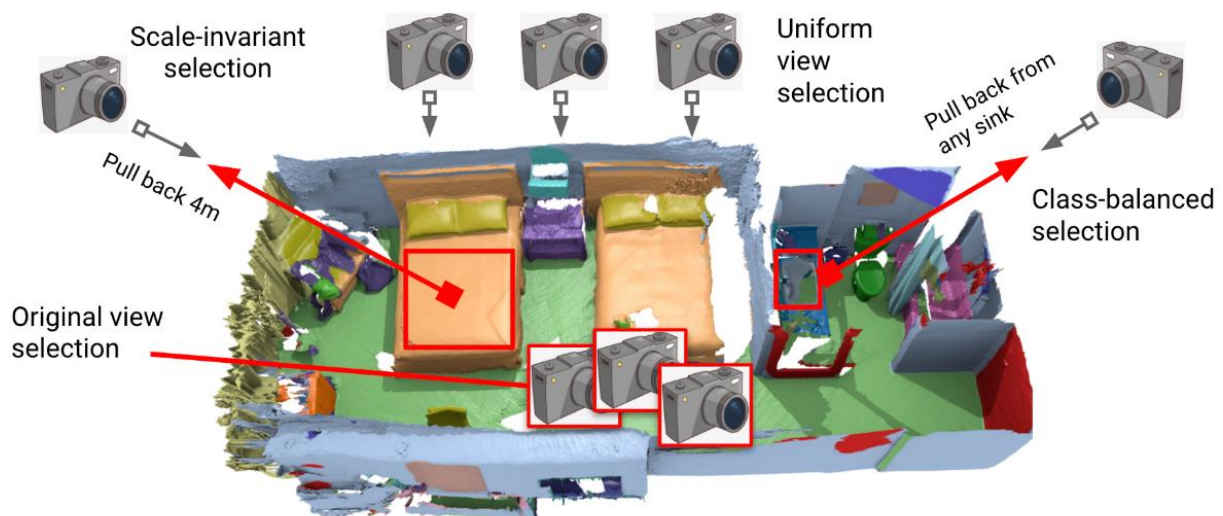


Figure 14 Proposed virtual view selection approaches. This approach gives us the possibility to freely choose the camera pose and parameters - even if it is not physically possible. (Kundu et al., 2020)

After the intrinsic & extrinsic and rendering parameters & channels are selected, they then generate training data by the selected virtual views and ground truth semantic labels. The 2D semantic segmentation models are trained using these data and this trained model will also be used in the inference stage. At inference stage, they select and render virtual views using a similar approach as in the training stage, but without the ground truth semantic labels. They conduct 2D semantic segmentation on the rendered virtual views using the trained model, project the 2D semantic features to 3D, then derive the semantic category in 3D by fusing multiple projected 2D semantic features. (Kundu et al., 2020)

This method has demonstrated excellent performance on both indoor datasets. It also shows that the simple method of carefully selecting and rendering virtual views enables multi-view fusion to outperform almost all recent 3D convolutional networks. However, notice that it might restrict the method to be used only over previously acquired data,

and not in real-time applications, since the pose of the camera is not physically possible. In addition, the code in this paper is not open source.

### 3.2.2 MVPNet (Multi-view PointNet)

**MVPNet** (Jaritz et al., 2019) takes dense multi-view images and sparse point cloud as input and fuses them to predict the semantic labels for each point.

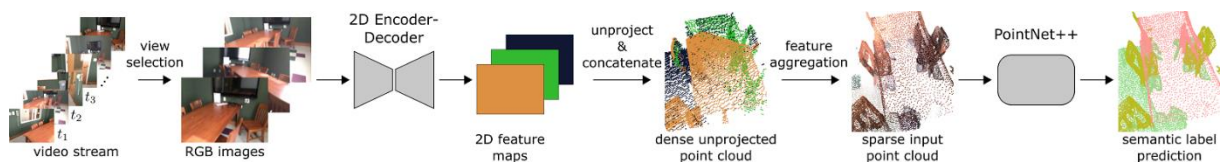


Figure 15 Pipeline of MVPNet (Jaritz et al., 2019)

Authors first divide the whole scene into chunks (around 90 chunks for an average scene) following PointNet++ implementation. For each chunk, a fixed number of 2D views (RGB-D frames) are selected so that the chunk is maximally covered. Those views are then fed into a 2D encoder-decoder network in order to obtain feature maps of same size as the input images. The pixels in images (feature maps) are unprojected into 3D space through the camera projection matrix to form a dense point cloud covering the entire chunk. Then, the features of the dense unprojected point cloud are aggregated into the sparse input point cloud to augment each point with these 2D image features. Finally, PointNet++ is applied to process the multi-view feature augmented point cloud and do the semantic segmentation task.

The novelty of this work is that they establish pixel-point-point correlations to transfer 2D features to the canonical 3D point cloud space and express the pixel-like 2D features as point cloud-like 3D features. Note that the pixel-point-point association here can be understood as follows: pixel-point can be explained as each pixel having a mapping relationship to a point in the dense point cloud generated by its projection. This means that the features of a pixel on the feature map of an image can be interpreted as the features of a point on the dense point cloud. The point-point represents the association between a dense point cloud and a sparse point cloud. This allows point features on a dense point cloud to be transferred to a sparse point cloud.

The advantage of this correlation is that once all modalities are represented in a 3D point cloud, the distance between two data points may be properly described in the continuous domain without discretization errors. This allows correlation between two data points to be precisely defined. Based on this, they have designed a novel feature

aggregation module which includes a shared MLP in order to adaptively distill 2D semantic features for 3D point cloud. According to the authors, the whole 2D-3D feature lifting module is differentiable, which enables end-to-end training and provides a great deal of flexibility. Furthermore, this 2D-3D lifting approach makes it feasible to fully exploit image features from a geometric perspective for 3D networks that require point clouds as input. (Jaritz et al., 2019) However, as a relatively old 3D network, PointNet++ may limit the performance of the overall model. Therefore, it would make sense to further improve the performance of MVPNet with a more powerful, SOTA 3D network.

### 3.2.3 Unified point-based framework

**Unified point-based framework** proposed by (Chiang et al., 2019) learns 2D textural appearance, 3D structures and global context features from point clouds. This method uses point-based networks to extract local geometric features and global context from sparsely sampled point sets without voxelization (Guo et al., 2020).

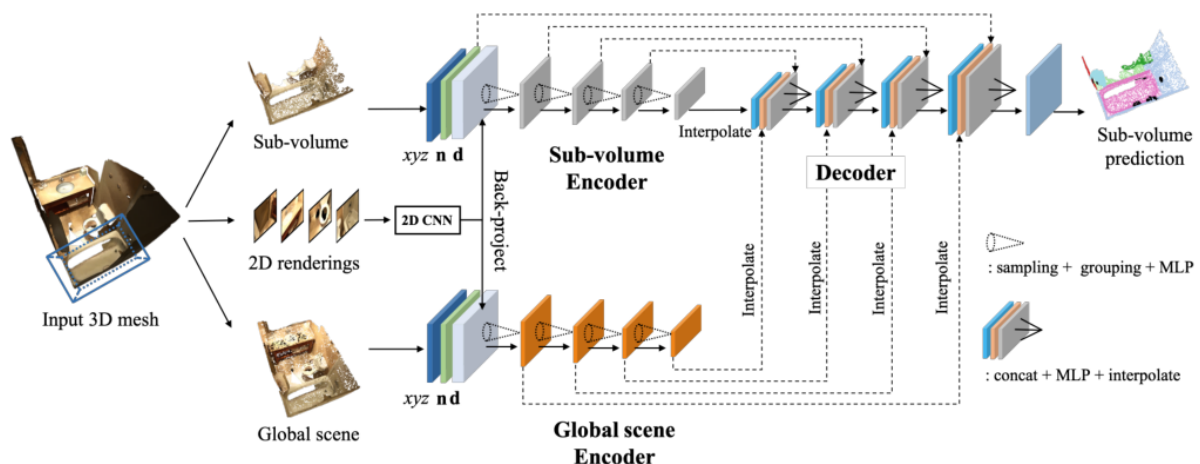


Figure 16 Pipeline of Unified Point-based Network (Chiang et al., 2019)

The framework consists of four parts: (1) They apply a 2D CNN to extract 2D features from rendered images and back project them into the 3D coordinates. The 2D features are interpolated and concatenated with 3D point features as inputs for 3D point-based networks; (2) Locally, a sub-volume encoder extracts local details in a target 3D sub-volume; (3) Globally, global context encoder extracts global context priors from a sampled sparse scene pointsets; (4) The decoder fuses the 2D image features, local features, and global feature in a concatenate manner. Then they deploy multiple MLPs and feature propagation layers from Pointnet++ to decode the aggregated features and upsample points to original input size. Finally, the point features are passed through an output layer consisting of two MLPs, which predicts the semantic labels for each

point in the sub-volume. The test results of this model on the ScanNet benchmark are slightly worse than those of MVPNet.

### 3.3 Instance segmentation

Instance segmentation is the process of recognizing each object instance for each known object in an image or point cloud. It combined object detection and semantic segmentation. The bounding box is used to classify individual objects and localize each object instance. To classify each pixel / point into a fixed set of categories without differentiating object instances, each instance must also be segmented.

#### 3.3.1 3D-SIS

(Hou et al., 2019) proposed a **3D fully convolutional Semantic Instance Segmentation** (3D-SIS) network to achieve semantic instance segmentation on RGB-D scans. The network consists of two parts: 3D detection backbone and 3D mask backbone. The multi-view image features (color information) are first extracted and downsampled using ENet based network (Paszke et al., 2016). Noted that here they choose to “summarize” the RGB information by decreasing the dimension of image features, which solved the mismatch problem between a high-resolution image feature map and a low-resolution voxelized point cloud feature map. They then back project each of these features back to 3D voxel grid using the corresponding depth image, camera intrinsic, and 6 DOF (Degree of freedom) poses. The image feature in 3D voxel representation will concatenate (fuse) with 3D voxels which contain geometric information.

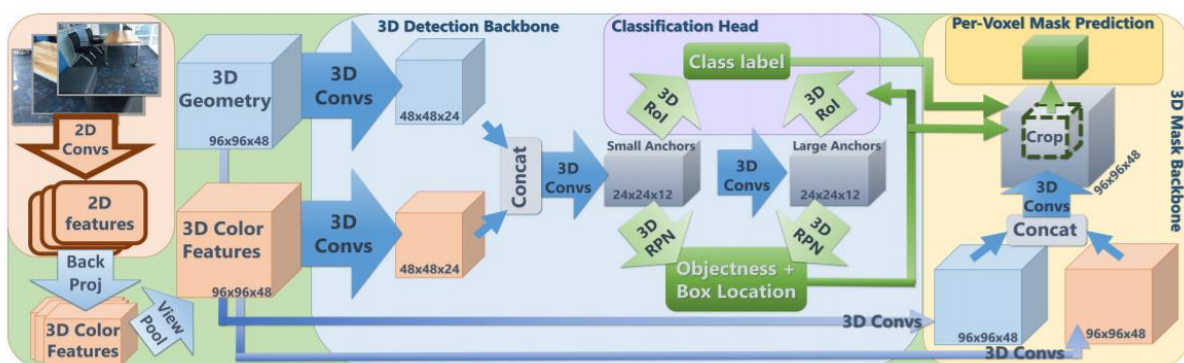


Figure 17 Pipeline of 3D-SIS (Hou et al., 2019)

In the 3D detection branch, the features of fusion data are further extracted through a 3D CNN block, and then enter the RPN (region proposal network). The network has two inputs, one is a feature layer with a smaller receptive field, which represents a small anchor, and the other is a layer with a larger receptive field, which represents a

large anchor. After training the RPN, use the ROI (Region of interest) generated by the RPN as input to train the classification branch. In the 3D mask branch, a 3D CNN takes images, point cloud features and 3D object detection results to predict per-voxel instance labels.

### 3.4 Summary

Table 3.1 Summary of STOA Multi-modal Method (all the values are in percentage) “-” means missing information

Network	Task	2D Backbone	3D Backbone	Fusion Operation and Method	Fusion Timing	KITTI <sup>1</sup> [AP]			ScanNetV2			S3DIS [mIoU]	SUN-RGBD [mAP]
						Cars	Pedestrians	Cyclists	Door [IoU]	Window [IoU]	mIoU		
Frustum-PointNet (Qi et al., 2018)	Object detection	Mask-RCNN (K. He et al., 2018)	PointNet++ (Qi, Yi, et al., 2017)	Ensemble: using region proposal of 2D detector to narrow the detection range of 3D network	Late; Decision level	69.79	42.15	56.12	-	-	-	-	54.0
Frustum-ConvNet (Wang & Jia, 2019)	Object detection	Faster R-CNN (Ren et al., 2017)	PointNet; Fully convolutional network	Ensemble: Using region proposal from RGB image detector to build frustums	Late; Decision level	76.39	43.38	65.07	-	-	-	-	57.5
SIFR-Net (Zhao et al., 2019)	Object detection	YOLOv3 (Redmon & Farhadi, 2018); ResNet-50 (K. He et al., 2016)	Point-UNet	Ensemble: using region proposal of YOLO detector; Feature concatenation: image features are extracted by ResNet and combined with point cloud as input to 3D network	Early; Late; Decision & Feature level	72.05 <sup>2</sup>	60.85 <sup>2</sup>	60.34 <sup>2</sup>	-	-	-	-	58.4
PointPainting (Vora et al., 2020)	Object detection	DeepLabv3 (L.-C. Chen et al., 2017)	PointRCNN (Shi et al., 2019)	Feature concatenation	Early	71.70	40.97	63.78	-	-	-	-	-
EPNet (Huang et al., 2020)	Object detection	2D CNN	PointNet-like layers	Feature concatenation; Adaptive fusion module; Multiplication; Addition	Middle	79.28	-	-	-	-	-	-	59.8

<sup>1</sup> Listed here are the average precision (AP) of 3D detection task for each category at moderate difficulty.

<sup>2</sup> results on KITTI validation set for 3D object detection.

Network	Task	2D Backbone	3D Backbone	Fusion Operation and Method	Fusion Timing	KITTI [AP]			ScanNetV2			S3DIS [mIoU]	SUN-RGBD [mAP]
						Cars	Pedestrians	Cyclists	Door [IoU]	Window [IoU]	mIoU <sup>3</sup>		
3D-CVF (Yoo et al., 2020)	Object detection	ResNet	PointNet	Feature concatenation; Adaptive fusion module	Early; Late	80.05	-	-	-	-	-	-	-
CLOCs (Pang et al., 2020)	Object detection	PV-RCNN (Shi et al., 2020)	Cascade-RCNN(Cai & Vasconcelos, 2018)	The 2D and 3D predictions are mixed and represented as sparse tensor and then input to 2D CNN for fusion.	Late; Decision level	80.67	-	-	-	-	-	-	-
Virtual MVFusion (Kundu et al., 2020)	Semantic segmentation	DeepLabv3; Xception65 (Google, 2014)	-	Average: projecting multiple image features to each 3D point, the fused feature at each point obtained by averaging all these features.	Late; Decision level	-	-	-	66.4	72.8	74.60	65.38 <sup>4</sup>	-
MVPNet (Jaritz et al., 2019)	Semantic segmentation	UNet-ResNet34	PointNet++	Feature concatenation	Early	-	-	-	55.3	60.8	64.10	62.43 <sup>4</sup>	-
Unified point-based Network (Chiang et al., 2019)	Semantic segmentation	DeepLab	PointNet++ like layers	Feature concatenation	Early	-	-	-	56.1	59.8	63.40	-	-
3D-SIS (Hou et al., 2019)	Instance segmentation	ENet	3D ResNet blocks	Feature concatenation	Early	-	-	-	32.0 <sup>5</sup>	23.5 <sup>5</sup>	38.2 <sup>5</sup>	-	-

<sup>3</sup> Mean Intersection over Union (mIoU) of 20 classes

<sup>4</sup> Area 5 of S3DIS is used here as test set.

<sup>5</sup> (Mean) average precision AP at overlap 0.50 (AP 50%), metric for instance segmentation.



From the above literature review, we can see that multimodal fusion methods not only perform well in 3D semantic segmentation tasks, but also show great potential in 3D object detection and instance segmentation tasks. Since the main goal of this study is to effectively fuse complementary information from 2D images and 3D point cloud in order to achieve better 3D scene understanding on real-world data, three multimodal fusion algorithms that rank high on the ScanNet 3D semantic segmentation benchmark are evaluated in the literature review. As one of the few open-source multimodal algorithms on the benchmark, MVPNet has the potential to continue to improve while demonstrating its excellent performance.

As mentioned in Chapter 3.2.2, MVPNet proposed a novel method to augment point cloud with 2D semantic features. A small number of images that can accurately cover a given point cloud can be selected in real-time by their proposed algorithm. The semantic features extracted by these images through the 2D network can also be transferred to the 3D point cloud through an end-to-end feature aggregation module. Since the final output of this flexible 2D-3D lifting method is a 3D point-like feature, it is possible to process these features using a different point-based 3D network.

As one of the best point-based 3D networks, KPConv accepts a similar input format as PointNet++. Compare to PointNet, the point convolution is better suited to extract fine-grained per-point and local geometry (Cui et al., 2021). Moreover, the KP-FCNN based on KPConv is more powerful, and it outperforms (68.4 mIoU) PointNet++ (33.9 mIoU) on the ScanNet 3D semantic benchmark. Therefore, it would be interesting to use KP-FCNN instead of PointNet++ to improve the metrics of MVPNet, or to integrate the modules of MVPNet into KP-FCNN to study the chemistry between them.

## 4 Methodology

The algorithm pipeline of this study is illustrate in Figure 18. The code of this method is mainly based on the KPConv implementation, and many modules and functions from original MVPNet are meaningfully incorporated.

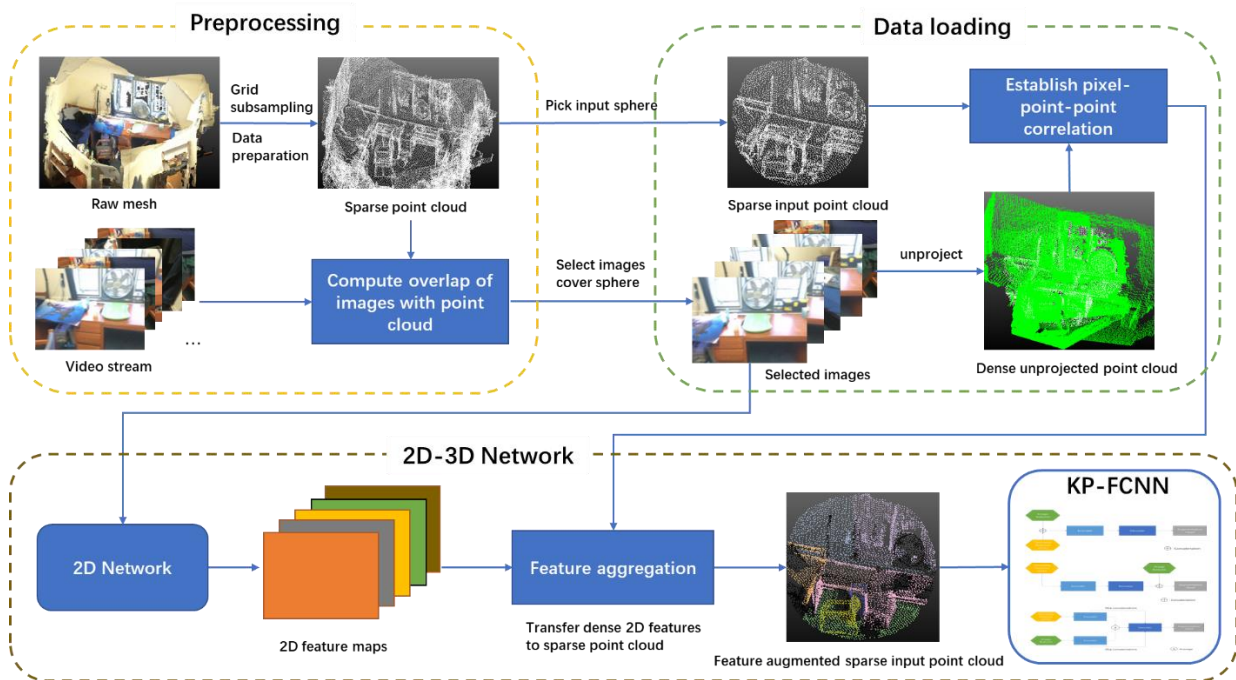


Figure 18 The algorithm pipeline of this study

The method starts with preprocessing stage. First, useful 2D and 3D information are extracted from the raw data and transformed into serialized files (pickle). Scene point clouds are subsampled to reduce the computational cost. The overlap of each video frame with the scene point cloud is computed in order to know the coverage area of each image. During the data loading phase, spherical sub-clouds are selected from the scene point cloud as input to the 3D network. Using the overlap information provided in the preprocessing phase, a certain number of images that can cover the input sphere well can be selected in real-time. Each pixel of these images is unprojected into 3D space to form a dense point cloud. Using KNN, the input sparse point cloud and the dense unprojected point cloud can be correlated. At this moment, the association between the image pixels, the dense unprojected point cloud, and the sparse input point cloud is established. Finally, the selected images are fed into a pre-trained 2D network to obtain feature maps of the same size as the input images. The feature aggregation module receives these feature maps associated with dense point

cloud, as well as the sparse spherical sub-clouds as inputs, and then transfers 2D semantic features from the dense point cloud to each sparse point through the previously established KNN correlations. Finally, the sparse point cloud augmented by the image features is fed into KP-FCNN to fuse with the geometric information. Several different fusion architectures are tested in KP-FCNN.

## 4.1 Dataset

The ScanNet (A. Dai et al., 2017) dataset covers various indoor scenes such as offices, bedrooms, and bathrooms etc., with a total of 2.5 million frames collected using an Occipital structure sensor attached to iPad Air2, which is a commodity RGB-D sensor with design similar to the Microsoft Kinect v1. The complete dataset contains 1201 training scenes and 312 validation scenes. Each scene's data contains an RGB-D sequence, corresponding camera postures, and a whole scene mesh annotated with 20 semantic classes. The test set includes 100 scenes with disguised ground truth for benchmarking.

Due to disk capacity limitations and because doors and windows are more generic and common objects for interior scenes, only scenes containing these two classes were selected from all training and validation sets by using the browse tool provided on the ScanNet website<sup>1</sup>. This sub-dataset containing 118 training scenes and 28 validation scenes was used in this study.

## 4.2 Preprocessing

### 4.2.1 Data Preparation

The idea of data preparation is similar to MVPNet. Each scene in the downloaded ScanNet dataset contains both 2D and 3D information. 3D information includes the whole scene mesh and 3D semantic labels stored in .ply format. 2D information is stored in .sens file, and the 2D frame color, depth, camera intrinsic, camera pose and 2D labels need to be extracted from this raw data by running the SensReader script provided by ScanNet. The extracted RGB images, depth maps and 2D labels are stored in .png or .jpg format with a native resolution of  $640 \times 480$ . To save disk space

---

<sup>1</sup> <http://www.scan-net.org/>

and data loading time, these images are then compressed to  $160 \times 120$  using PIL (Python Image Library). Although the pixel count is much reduced, they are still sufficient to provide rich texture information (Jaritz et al., 2019).

Accordingly, 3D information also needs to be processed. All the original 3D information of each scene is stored as mesh in a .ply file, and the XYZ coordinates, RGB values and 3D labels of the points can be obtained by accessing vertices. The extracted 3D information of each scene will be stored in a dictionary in the form of numpy array. Eventually, all this dictionary data in the training set (validation set) is dumped into a pickle (.pkl) file and stored on disk. This approach can greatly speed up the reading of data.

At this point, the data preparation is complete. The 3D information of each scene from each split can simply be retrieved from the pickle file by the “load” command; the 2D information can be loaded directly from the folder by the function provided by PIL, so there is no need to also dump it into pickle form.

#### 4.2.2 Point cloud sampling

Point clouds of real scenes usually show different densities. This is related to the scanning techniques and equipment used to capture these point clouds. For example, fixed laser scans often suffer from bad sampling, with a huge number of points close to the scanner and few points far from it. Different densities affect the results and efficiency of point cloud processing methods. It will be difficult to discern object shapes if the density is too low, and the computation time will be too long if the density is too high. The trade-off between performance and efficiency is determined by the size of the object and the level of detail required by the processing method. Subsampling methods can adjust the density of points to different scales. From the original point cloud, these methods generate new point cloud with a reduced density. These methods are typically used to balance the density of the entire scene (Madali, 2021).

In this study, grid subsampling provided by KPConv is used as sampling strategy. It projects the point cloud into a 3D grid, where each voxel retains only one point closest to the barycenter of the voxel. This method has many advantages. It equalizes the density of the point cloud, the density of point cloud can be easily controlled through voxel size, and is very fast.

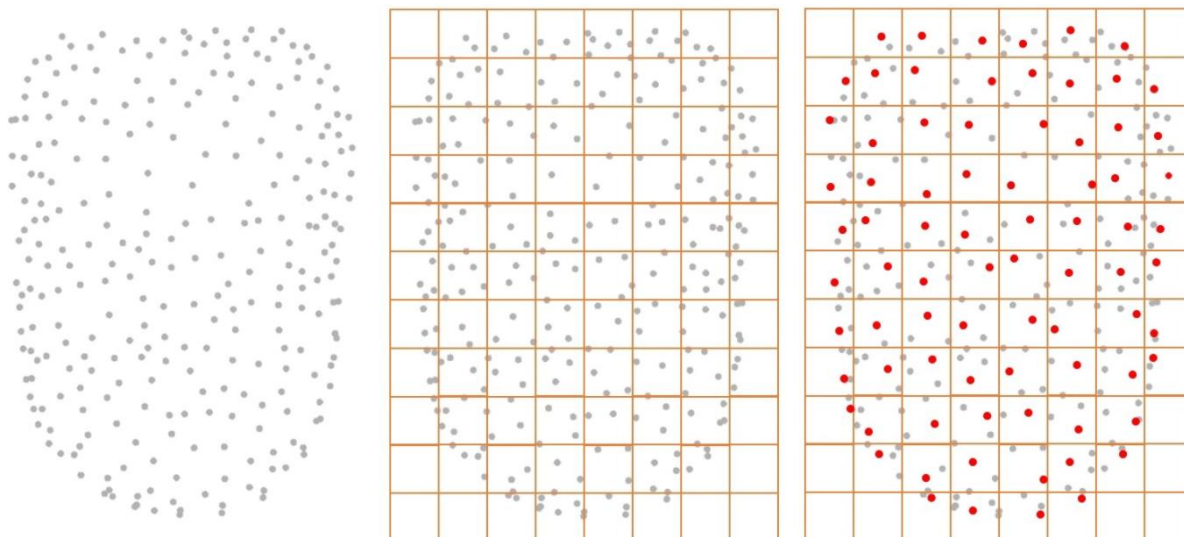


Figure 19 Grid subsampling illustrated on 2D points. Each small grid contains a number of points, and the point closest to the center of the grid is taken as the sampling point (Madali, 2021).

### 4.2.3 Compute the overlap of images with point cloud

ScanNet offers a variety of indoor scenes of varying sizes. Small scenes, such as bathrooms and bedrooms, usually contain around 1500 RGB-D images taken within the scene; large scenes, such as living rooms and offices, can have around 6000 images. These RGB-D images are actually frames in the video stream, and there is a strong overlap between consecutive frames. It would be redundant and computationally expensive to process them all. Also, although the scene point cloud have been grid sampled before being fed into network, the scene is still too large to be segmented as a whole. Therefore, KPConv actually takes spheres in the whole scene point cloud and then segments the sub-point cloud contained inside the sphere. Due to these two characteristics, we need to know which region of the whole scene point cloud is covered by each frame, so that we can accurately select the images that maximize coverage of the input sphere from the redundant video stream during data loading process.

First, 6000 base points are randomly selected in the whole scene point cloud after being sampled by the grid subsampling. Note that the original MVPNet picks 4000 base points on an unsampled point cloud. To improve accuracy on sparser point clouds, the number of base points is increased. After picking base points, a KD-tree is created for the base point cloud, for later neighborhood search. Then, all RGB frames belonging to this scene are looped over. For each frame, it is projected into the 3D domain using its depth map, pose and the camera intrinsic matrix. For each point in the projected point cloud, find its nearest neighbor (only one and up to 1 cm away) in the whole

scene base point cloud. Eventually, the so-called overlap is actually a Boolean type mask array with the length of the number of base points selected in the scene. If a nearest neighbor base point is found, then the index of this base point is noted down and the Boolean value of this index position in the mask array is assigned to true. For example, if 3 points are selected as base points in a scene (this is only an example, there are actually 6000 base points), they might be the 3rd, 415th and 801st points in the whole scene points. Their indices can be summarized as an array like  $[3, 415, 810]$ . If the mask array for frame  $X$  is  $[True, True, False]$ , then we can say that frame  $X$  covers base point 0 and base point 1, but uncovers base point 2. From this we can know which area of the original point cloud is covered by this frame by knowing which base points it can cover. Finally, the overlap of all images of a scene with this scene point cloud can be expressed as a 2D mask array  $pointwise\_rgb\_overlap \in Boolean^{NB \times NF}$ , where  $NB$  refers to the number of all base points in the scene and  $NF$  refers to the number of all RGB frames in the scene.

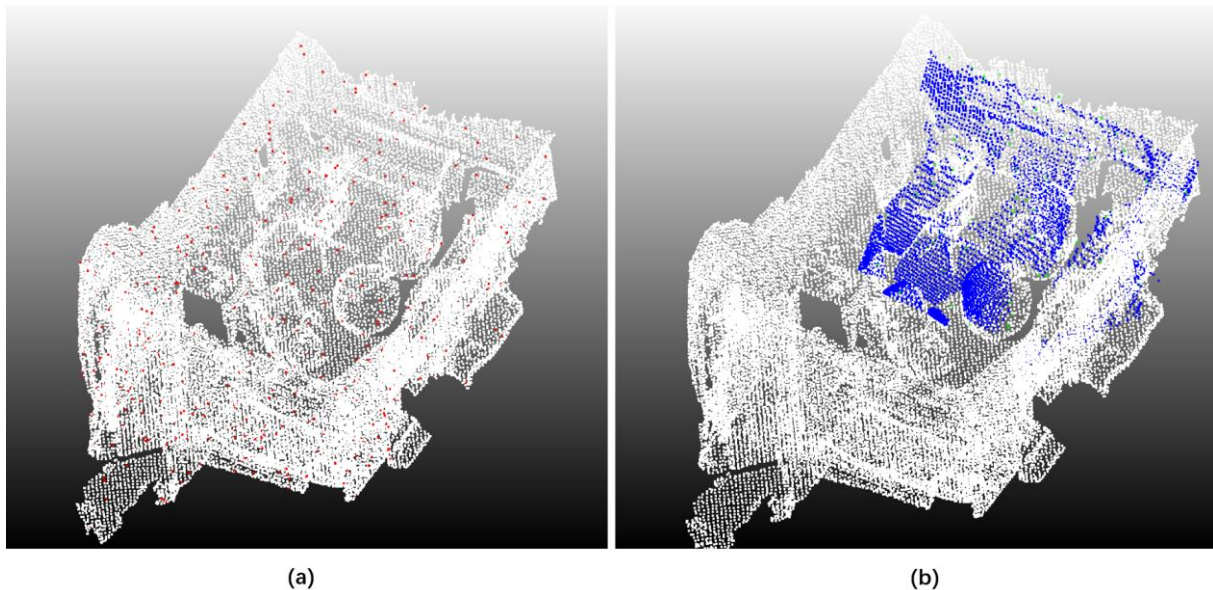


Figure 20 Compute the overlap of an image with scene point cloud using base point method. The white points are the whole scene point clouds after grid subsampling. The red points in (a) are the selected base points. The blue points in (b) are the dense point cloud formed by projecting the image into 3D space. The green points in (b) represent the red base points covered by the dense point cloud.

### 4.3 Data loading

After preparing the data and knowing the coverage area of each image, the data loader for the network can be built. The following sections introduce several key steps in the data loading process.

#### 4.3.1 The picking strategy

Like mentioned before, the scene in dataset is too big to be processed as a whole by the network. Thus, the input to KP-FCNN is actually the spherical sub-clouds of the scene. The spheres that contain the input sub-clouds are smaller than the entire scene but large enough to cover some objects. It is possible to randomly select the location of the input spheres in the scene, but the prerequisite for using this approach is that the density of the scene point cloud is relatively uniform. If the point cloud density is not balanced, then regions with higher point densities have a higher probability of being selected and will therefore be sampled more often than regions with lower densities. This creates a bias in the network training (Thomas, 2020).

In order to have every region of the scene sampled evenly, KPConv uses a potential picking strategy to select the input spheres. First, the scene point cloud is continue subsampled to obtain some coarse points that are uniformly distributed in the scene. These points will be the potential locations for spherical center. An initial potential value will be assigned to each coarse point, and when a point is selected, its potential value will be increased and the potential values of its surrounding coarse points will also be updated by using Tukey weights. The function of Tukey weights is defined as follows:

$$Tukey(d) = \begin{cases} \left(1 - \left(\frac{d}{d_{Tukey}}\right)^2\right)^2, & \text{if } d \leq d_{Tukey} \\ 0, & \text{if } d > d_{Tukey} \end{cases}$$

The  $d_{Tukey}$  is defined as  $d_{Tukey} = R/3$ , where  $R$  is the radius of input sphere. The magnitude of the Tukey weight is related to the distance  $d$  from the center of the sphere. Points that are closer than the distance  $d_{Tukey}$  are updated. This range was chosen to be lower than the radius of the actual input spheres, which causes overlap between the input spheres. This guarantees that each point is tested multiple times by different sphere locations. The closer the point is to the center of the sphere, the greater the weight and the greater the increase in potential value, and vice versa. Each new sphere center is picked as the **minimum** of the potentials across the scene (see Figure 21). In this way, the spatial regularity of the picking can be ensured by tracking the potential value of each coarse point (Thomas, 2020).

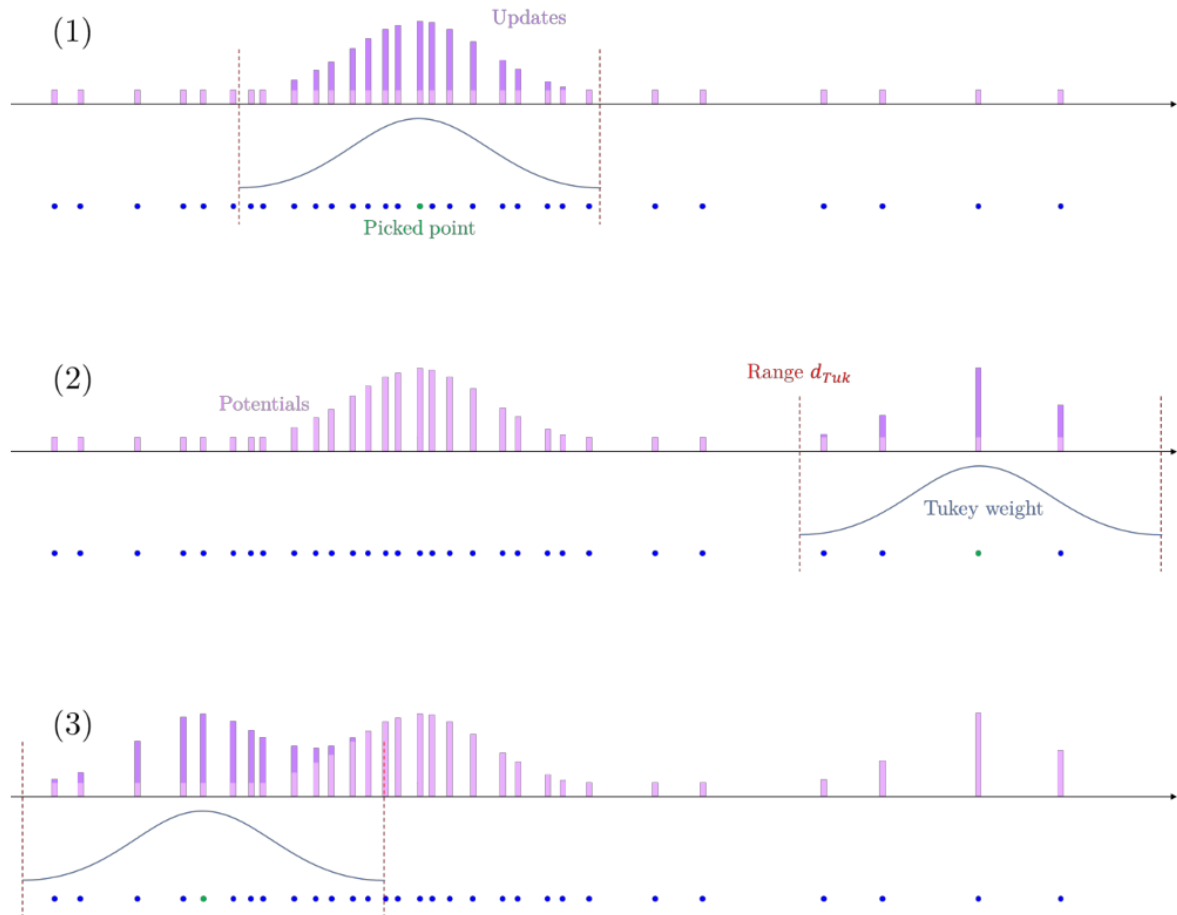


Figure 21 For clarification, a 1D illustration of the spatially regular selection method with potential updates is shown (Thomas, 2020). As showed in (1), the selected center of the sphere (green dots) has the highest potential value, and the potential values added to the surrounding coarse points decrease in order of distance. (2) shows that only the potential values of points in the range  $d_{Tuk}$  are updated. (3) demonstrates that each time the point with the lowest potential value in the scene is selected as the new input sphere center. This strategy helps the network to be robust to varying densities and pick the same amount of sphere in every location of the scene.

### 4.3.2 View selection

During data loading, the overlap information obtained by chapter 4.2.3 is used to select the RGB-D frames on-the-fly with a greedy algorithm (Jaritz et al., 2019). Knowing which base points are included in the input sphere sub-cloud, the overlap information can continue to be filtered to know which images these base points can be covered by. The image that covers the largest number of base points is selected first. After one selection is done, all the base points covered by this image are set to invalid and then the next round of image selection is started until the desired number of images are selected. For example, imaging the overlap information for an input sphere looks like this:

$$Overlap = \begin{bmatrix} [True, True, False], \\ [True, False, False], \\ [True, False, False], \\ [False, True, False] \end{bmatrix}$$



This 2D mask array has the shape (4, 3), where 4 means there are 4 base points in the sphere sub-cloud and 3 means there are a total of 3 images in this whole scene (for ease of explanation only). Since Boolean values are actually 0 and 1, we can add up the rows in this matrix above along the columns. As a result, we can get an array like this [3, 2, 0]. The indices of this array represent the IDs of the images of this scene, and the values inside the array represent the number of base points covered by each image in the spherical sub-cloud. Therefore, the first image has the maximum coverage since it covers 3 base points. Since the first 3 base points have been covered, the overlap matrix has to be adjusted before the next selection round starts:

$$Overlap\_update = \begin{bmatrix} [False, False, False], \\ [False, False, False], \\ [False, False, False], \\ [False, True, False] \end{bmatrix}$$

The selected views need to be projected into 3D space to form a dense point cloud that completely covers the spherical sub-cloud. For each point in the spherical sub-cloud, find its K nearest neighbors in the dense point cloud and return the indices of these neighbors. Based on MVPNet's experience, the value of K was set to 3 in method.

---

**Algorithm 1:** Select the views covering the input sphere sub-cloud and establish the correspondence between dense pixels and sparse point clouds

---

**Inputs:**

$F \in R^{NF \times H \times W \times 3}$ : RGB images of a scene.  $NF$  refers to number of all frames in the scene.

$D \in R^{NF \times H \times W \times 1}$ : Image corresponding depth maps.

$M \in R^{4 \times 4}$ : Camera intrinsic matrix.

$P \in R^{4 \times 4}$ : Camera pose.

$S_{sparse} \in R^{NS \times 3}$ : Input spherical sparse sub-point cloud, with  $NS$  points.

$O_{images} \in B^{NB \times NF}$ : Overlap of all images of a scene with this scene point cloud.  $NB$  refers to number of all base points in the scene.

$I_{base} \in R^{NBS}$ : Indices of the base points contained in the input spherical sub-cloud.  $NBS$  refers to number of base points in sphere.

$NV$ : Number of views to select.

**Outputs:**

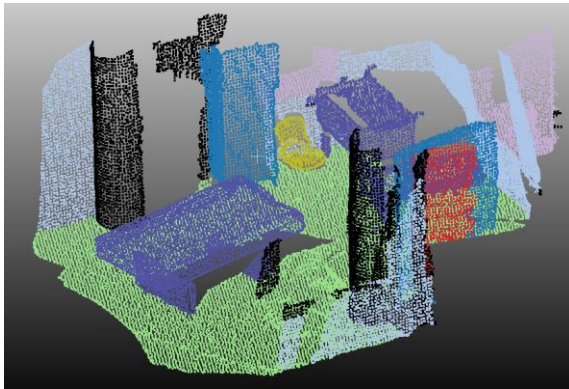
$F_{select} \in R^{NV \times H \times W \times 3}$ : Selected RGB views with  $NV$  selected views, 3 RGB channels.

$S_{dense} \in R^{NV \times H \times W \times 3}$ : Dense point clouds formed by projected frames with  $NV$  selected views, 3 XYZ channels.

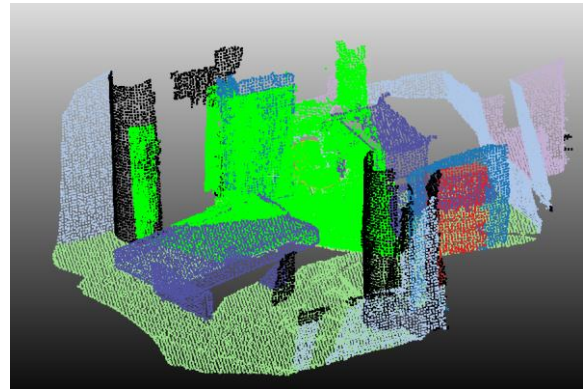
---

$KN \in R^{N \times K}$ : Indices of the  $K$  nearest neighbors in the  $S_{dense}$  for each point in the  $S_{sparse}$ .

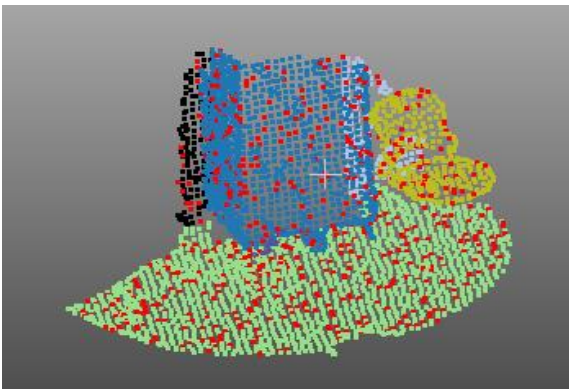
1.  $Overlap \leftarrow O_{images}[I_{base}]$   $\triangleright Overlap \in B^{NBS \times NF}$
2. **for**  $i = 0$  **to**  $NV$  **do**
3.      $frameID \leftarrow ChooseFrameIDWithMaximumNumberOfTrue(Overlap)$
4.      $F_{select} \leftarrow F[frameID]$
5.      $D_{select} \leftarrow D[frameID]$
6.      $Overlap[Overlap[:, frameID]] \leftarrow False$
7. **end**
8.  $S_{dense} \leftarrow Projection(D_{select}, M, P)$
9. **for**  $point$  **in**  $S_{sparse}$  **do**
10.      $KN \leftarrow \mathbf{find} \ K \ \mathbf{nearest} \ \mathbf{neighbor} \ \mathbf{in} \ S_{dense}$   $\triangleright KN \in R^{NS \times K}$
11. **end**
12. **return**  $F_{select}, S_{dense}, KN$



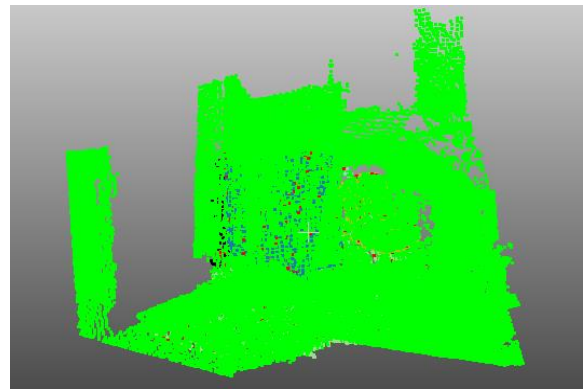
(a)



(d)



(b)



(c)

Figure 22 (a) is a whole scene point cloud with labels. (b) is the input spherical subcloud (labels are only for better visualization). The red dots are the base points inside it. (c) is the dense point cloud formed by the projection of the

5 selected views. (d) is what the dense point cloud looks like in the full scene view. It can be seen that the selected 2D views cover the input subcloud very well, providing rich texture information.

### 4.3.3 Data augmentation

Data augmentation is used for both 2D and 3D data to increase the variety of input and help the robustness of the network. The augmentation strategy for 2D images follows the original MVPNet. Color jitter and flip are applied to the selected images (Jaritz et al., 2019). As for 3D data, the input point clouds to KPConv are scaled independently in each dimension. The scaling factor is picked uniformly in  $[0.8, 1.2]$ . Also, they rotate the point clouds around the vertical axis with a random angle in  $[0, 2\pi]$ . Besides these, a gaussian noise is added to the point coordinates to perturb their positions (Thomas, 2020).

### 4.3.4 Variable batch size

The point cloud input to KPConv is a spherical sub-cloud. In the preprocessing stage, a KD tree is built up for the whole scene point cloud data sampled by the grid subsampling. Later, in the data loading phase, the potential strategy is used to select the location of the sphere center point. Given the radius of the sphere and the known spherical center point, the KD tree can be used to query all neighbors of the center point within the radius, thus obtaining a spherical sub-cloud. Since the distance criterion is used to obtain the spherical neighborhoods, it is not guaranteed that the number of points within all spheres is the same, since spheres at different locations may contain different objects. A network usually processes batches of a few point clouds, but in case of KPConv, the input point clouds have varying sizes, so they cannot be stacked along a new "batch" dimension. Thus, the point and point-wise feature tensors need to be stacked along their first dimension (number of points). Because the number of points can vary greatly, they employ a **variable** batch size, picking as many batch elements as possible until a certain total number of batch points is attained (Hermosilla et al., 2018). This limit is established to ensure that the average batch size matches the target batch size (set as a hyperparameter).

However, it is not only the points that need to be fed into the network, but also the selected images  $F_{select} \in R^{NV \times H \times W \times 3}$ , the dense projected point cloud  $S_{dense} \in R^{NV \times H \times W \times 3}$  and the KNN indices  $KN \in R^{NS \times K}$  from chapter 4.3.2. They need to be stacked along the "batch" dimension, because the 2D network and the feature aggregation module provided by MVPNet require such an input format. So here a little trick

is used: the  $F_{select}$  and  $S_{dense}$  tensors are given an expanded dimension as the batch dimension and let them be stacked along this dimension. The size of one dimension of the  $KN$  is the variable number of points  $N$ , so it is not credible to apply the above method. Instead of expanding a new dimension, the  $KN$  of each batch element is appended to a list to be fed into the network.

#### 4.4 The Network

The following pseudo-code describes the pipeline of the proposed network.

First, the images are processed by a 2D network (see section 4.4.1) to obtain 2D semantic feature maps  $feature2D \in R^{B \times C_{feat} \times NV \times H \times W}$ , where  $B$  denotes the batch size,  $C_{feat}$  denotes the number of feature channels (64),  $NV$  denotes the number of images selected for each input sub-cloud of batch elements, and  $H$  and  $W$  denote the height and width of the images. In Section 4.3.2 we establish the pixel-point-point correlation, which is expressed as  $KN \in R^{NS \times K}$ , i.e., the indices of the  $K$  nearest neighbors in the dense projected point cloud for each point in spherical sparse sub-point cloud  $S_{sparse} \in R^{NS \times 3}$ . Since the dense point cloud is nothing but a projection of each pixel of the images into the 3D space,  $KN$  can also be interpreted as the indices of the  $K$  nearest neighbor pixels in the images for each point in  $S_{sparse} \in R^{NS \times 3}$ . Since  $KN$  is input to the network as a list, a for loop is used to extract the  $KN_{batch_i}$  belonging to each batch element. For each batch element, the  $KN_{batch_i}$  and  $GatherNeighborPointsByIndices$  functions can be used to obtain the semantic features of the  $K$  nearest neighbor pixels in the 2D feature maps  $feature2D_{batch_i}$  for each point in the input sparse sub-cloud  $S_{sparse} \in R^{NS \times 3}$ . The same operation is done on the dense projected point cloud  $S_{dense}^{batch_i}$  of each batch element. As the outputs of the above operations, the processed  $Useful\_feature2D_{batch_i}$  and the  $Useful\_S_{dense}^{batch_i}$  of all batch elements are concatenated along their dimension of the number of points ( $NS$ ), so that they can later be used by the *FeatureAggregation* module. Here  $Useful\_S_{dense} \in R^{1 \times 3 \times N \times K}$  represents the coordinates of the  $K$  nearest neighbor points of each point in the input sparse point cloud  $S_{sparse} \in R^{N \times 3}$  in the dense point cloud  $S_{dense} \in R^{B \times NV \times H \times W \times 3}$ . The  $Useful\_feature2D \in R^{1 \times C_{feat} \times N \times K}$  can be interpreted as the semantic features at the  $K$  nearest neighbor points in the dense point cloud  $S_{dense} \in R^{B \times NV \times H \times W \times 3}$  for each point in the input sparse point cloud  $S_{sparse} \in R^{N \times 3}$ , thanks to the pixel-point-point correlation (see Chapter 3.2.2).

In fact, these two tensors can be interpreted as **grouped points** in dense unprojected point cloud  $S_{dense}$ . Take  $Useful\_feature2D$  as an example, its dimension is  $1 \times C_{feat} \times N \times K$ . Imagine that there are  $N$  points in the input sparse point cloud, and each point has  $K$  nearest neighbors in the dense point cloud, then  $N \times K$  actually expresses  $N$  groups of  $K$  points in the dense point cloud, and  $C_{feat}$  means that each of these points has  $C_{feat}$  channels of 2D semantic features.  $Useful\_S_{dense} \in R^{1 \times 3 \times N \times K}$  tensor is the same thing, and they have almost the same shape - the only difference is that here each point has only 3 channels of geometric features, i.e., coordinates XYZ. These grouped neighborhood points in  $S_{dense}$  are then fed into the *FeatureAggregation* module with  $S_{sparse}$ , which includes a shared MLP inspired by (Liang et al., 2018) in order to distill a new feature for each point in  $S_{sparse}$  from its  $K$  nearest neighbors in  $S_{dense}$  (Jaritz et al., 2019). (See section 4.4.2)

Subsequently, the distilled 2D feature tensor is concatenated with geometry feature (Z) and a constant tensor with all values of 1 to form a fused feature. The reason why only Z coordinates are used instead of XY coordinates is explained in Section 5.5.1. This constant 1 feature is to prevent black/dark points from being ignored, since KPConv considers a point with all feature equal to zero equivalent to empty space (Thomas et al., 2019). This fused feature will then be fed into KP-FCNN. Section 4.4.3 shows some architectures of the KP-FCNN used in this study.

---

**Algorithm 2:** The forward function of the proposed network

---

**Inputs:**

$F_{select} \in R^{B \times NV \times H \times W \times 3}$ : Selected RGB images stacked along batch dimension with  $NV$  selected views, 3 RGB channels,  $B$  refers to batch size,  $H$  and  $W$  are height and width of image.

$KN_{List} \in R^{NS \times K}$ : A list of Indices of the  $K$  nearest neighbors in the  $S_{dense}$  for each point in the  $S_{sparse}$ . The length of the list is the batch size.  $NS$  refers to various number of input points in each batch element (see algorithm 1).

$S_{dense} \in R^{B \times NV \times H \times W \times 3}$ : Dense point clouds formed by projected frames with  $NV$  selected views, 3 XYZ channels, stacked along batch dimension.

$S_{sparse} \in R^{N \times 3}$ : Sparse input point clouds with  $N$  points.

$Z \in R^{N \times 1}$ : Z coordinates of sparse input point clouds as geometric feature

**Output:**

$Prediction \in R^{N \times 20}$ : Network prediction, 20 refers to 20 semantic classes.

---

1.  $B \leftarrow \text{GetVariableBatchSize}(F_{select})$
2.  $feature2D \leftarrow \text{Network2D}(F_{select})$   $\triangleright feature2D \in R^{B \times C_{feat} \times NV \times H \times W}$
3.  $feature2D \leftarrow \text{Reshape}(feature2D)$   $\triangleright feature2D \in R^{B \times C_{feat} \times NV \times H \times W}$
4.  $S_{dense} \leftarrow \text{Reshape}(S_{dense})$   $\triangleright S_{dense} \in R^{B \times 3 \times NV \times H \times W}$
5.  $feature2D_{List} \leftarrow []$
6.  $S_{dense}^{List} \leftarrow []$
7. **for**  $i = 0$  **to**  $B$  **do**
8.  $KN_{batch\_i} \leftarrow KN_{List}[i]$   $\triangleright KN_{batch\_i} \in R^{NS \times K}$
9.  $feature2D_{batch\_i} \leftarrow feature2D[i, :, :]$   
 $\triangleright feature2D_{batch\_i} \in R^{1 \times C_{feat} \times NV \times H \times W}$
10.  $Useful\_feature2D_{batch\_i} \leftarrow$   
 $\text{GatherNeighborPointsByIndices}(feature2D_{batch\_i}, KN_{batch\_i})$   
 $\triangleright Useful\_feature2D_{batch\_i} \in R^{1 \times C_{feat} \times NS \times K}$
11. **append**  $Useful\_feature2D_{batch\_i}$  **to**  $feature2D_{List}$
12.  $S_{dense}^{batch\_i} \leftarrow S_{dense}[i, :, :]$   $\triangleright S_{dense}^{batch\_i} \in R^{1 \times 3 \times NV \times H \times W}$
13.  $Useful\_S_{dense}^{batch\_i} \leftarrow \text{GatherNeighborPointsByIndices}(S_{dense}^{batch\_i}, KN_{batch\_i})$   
 $\triangleright Useful\_S_{dense}^{batch\_i} \in R^{1 \times 3 \times NS \times K}$
14. **append**  $Useful\_S_{dense}^{batch\_i}$  **to**  $S_{dense}^{List}$
15. **end**
16.  $Useful\_feature2D \leftarrow \text{Concatenate all elements in } feature2D_{List}$   
 $\triangleright Useful\_feature2D \in R^{1 \times C_{feat} \times N \times K}$
17.  $Useful\_S_{dense} \leftarrow \text{Concatenate all elements in } S_{dense}^{List}$   
 $\triangleright Useful\_S_{dense} \in R^{1 \times 3 \times N \times K}$
18.  $feature2D3D$   
 $\leftarrow \text{FeatureAggregation}(Useful\_S_{dense}, S_{sparse}, Useful\_feature2D)$   
 $\triangleright feature2D3D \in R^{N \times C_{feat}}$
19.  $input\_feature \leftarrow \text{Concatenate}(1, Z, feature2D3D)$   
 $\triangleright input\_feature \in R^{N \times C_{feat\_fused}}$
20.  $Predicton \leftarrow \text{KPCConv}(S_{sparse}, input\_feature)$
21. **return**  $Predicton$

#### 4.4.1 2D Network

The 2D network architecture from the original MVPNet was continued to be used. The backbone of the 2D Encoder network is an ImageNet-pretrained ResNet34 (K. He et al., 2016) with batch normalization and dropout. The decoder network is a lightweight variant of U-Net (Ronneberger et al., 2015). Here, the convolution is used to fuse concatenated features from skip connections (copy), and the transposed convolution is applied for upsampling. Batch normalization and ReLU are attached after each convolution layer. This 2D network was pretrained first on the task of 2D semantic segmentation on our custom ScanNet dataset, and then integrated into the whole pipeline with frozen weights. The feature map output by the 2D network has 64 channels of semantic features. Furthermore, the size of the output feature map  $H \times W$  is equal to that of the input image, and fixed to  $160 \times 120$ . This makes subsequent 2D-3D feature lifting possible (Jaritz et al., 2019).

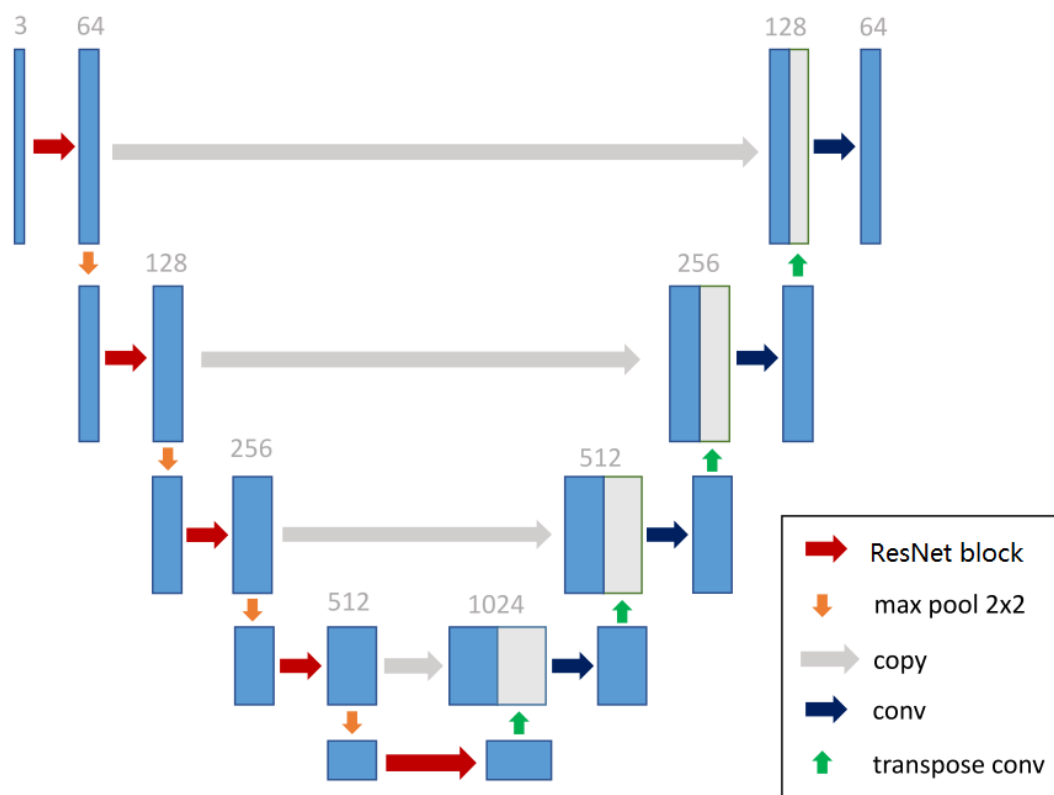


Figure 23 The architecture of the 2D encoder-decoder network (Jaritz et al., 2019)

#### 4.4.2 Feature aggregation module

At the beginning of Section 4.4, it is mentioned that the *FeatureAggregation* module receives groups of neighboring points in the dense point cloud  $S_{dense}$ , which have semantic features and spatial locations. For semantic segmentation, the labels have to be predicted for the input point cloud  $S_{sparse}$ . Thus, the features from the unprojected point cloud  $S_{dense}$  need to be transferred to  $S_{sparse}$ . The *FeatureAggregation* module is used to distill a new feature for each point in  $S_{sparse}$  from its  $K$  nearest neighbors in  $S_{dense}$  (Jaritz et al., 2019):

$$F_i = \sum_{j \in N_K(i)} MLP(\text{concatenate}[f_j, f_{dist}(x_i, x_j)])$$

where  $F_i$  is the distilled feature at point  $x_i$  in  $S_{sparse}$ ,  $f_j$  the semantic feature at one of the  $K$  nearest neighbor points  $x_j$  in  $S_{dense}$ , and  $f_{dist}(x_i, x_j)$  the distance feature between the two points which can be defined as:

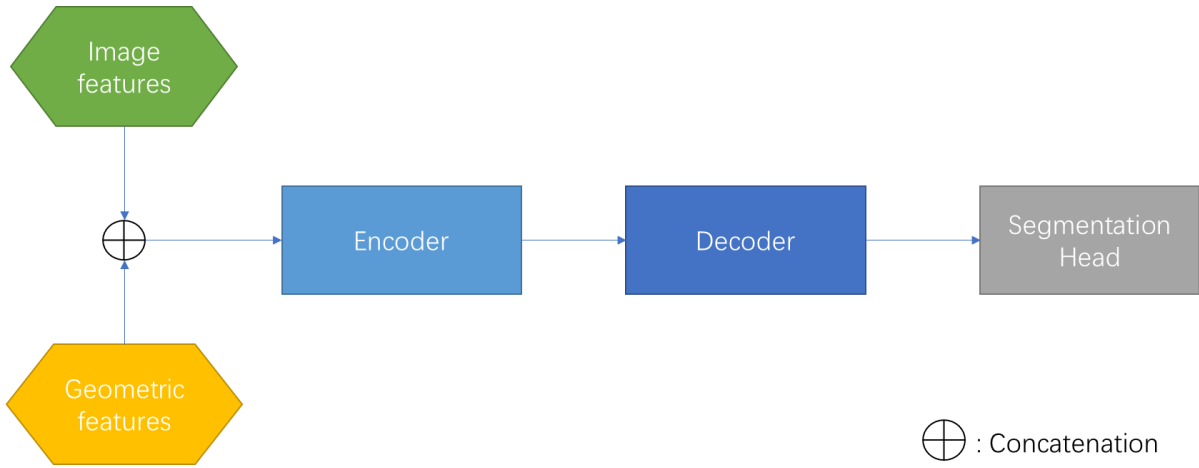
$$f_{dist}(x_i, x_j) = \text{concatenate}[x_i - x_j, \|x_i - x_j\|^2]$$

The author argued that the MLP can transform 2D image features to an embedding space more consistent with the 3D representation (Jaritz et al., 2019). In this study, a three-layer MLP with 64 channels is used. Eventually, the features of the  $K$  nearest neighbor points are fused by the summation operation to become 64-channel semantic features attached to each point of the  $S_{sparse}$ . Note that the *FeatureAggregation* module is differentiable and the weights inside the MLP need to be learned and updated by back propagation of the network (see Section 6.2.1). And since this module has no loss function of its own, its internal weights are adjusted by the loss function of the 3D network, i.e., KP-FCNN.

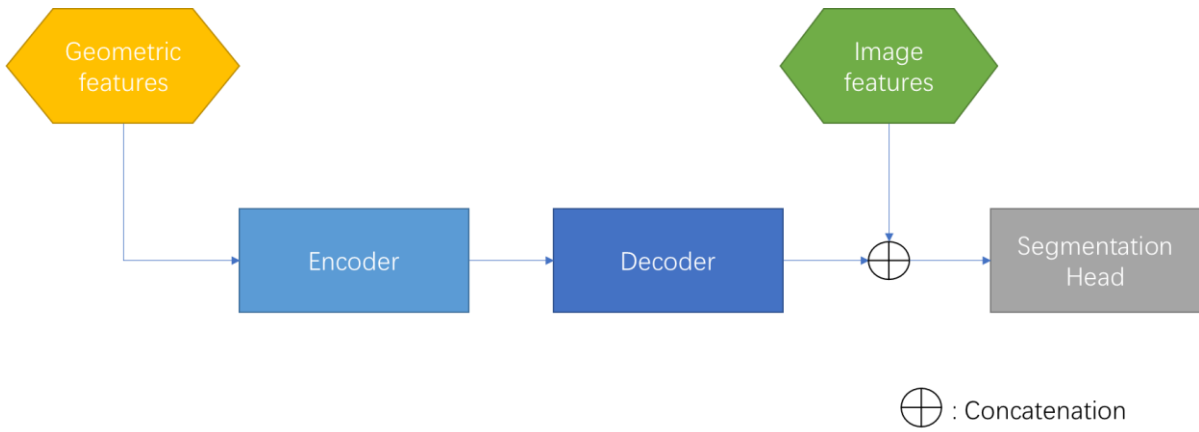
#### 4.4.3 KP-FCNN

The architecture of the KP-FCNN in this study basically follows the standard structure presented in Chapter 2.6. The encoder part still consists of five layers, but except for the initial layer, a standard KPConv block has been added to the remaining layers, making the network structure deeper and thus better able to extract features. Based on this infrastructure, three fusion architectures were designed to investigate the impact of fusion timing on the proposed network.

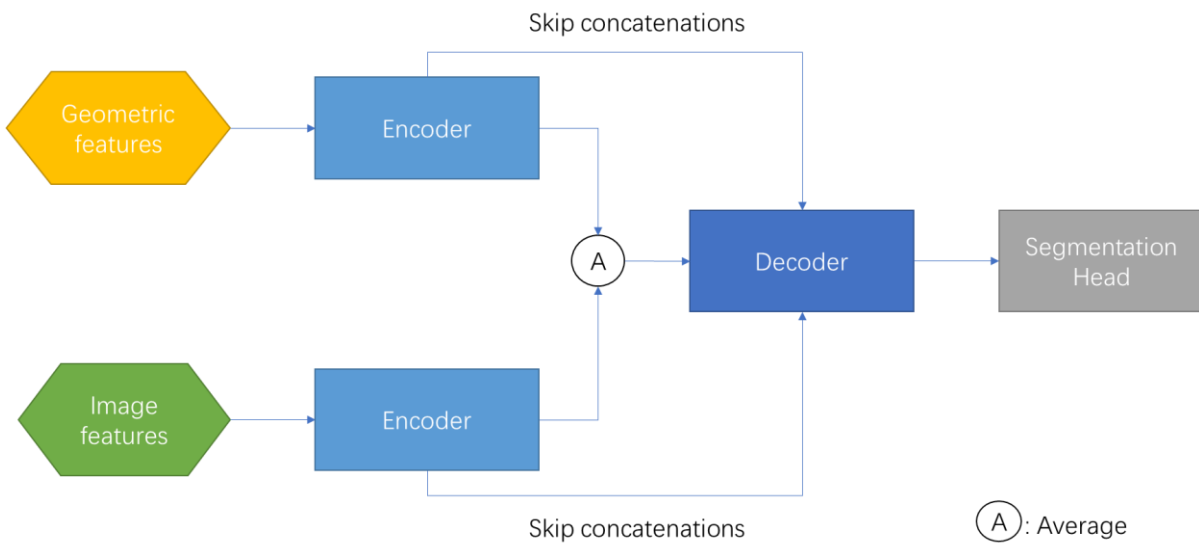




(a) Early fusion



(b) Late fusion



(c) Middle fusion

Figure 24 Three fusion architectures base on KP-FCNN

---

In the early fusion variant, the geometric feature and the 64-channel image features output by the feature aggregation module are concatenated and fed into the KP-FCNN. In the late fusion, geometric feature are passed through the encoder and decoder of KP-FCNN and then connected with image features in front of the segmentation head. In middle fusion, geometric features and image features are passed through sperate encoders and then averaged and fused before the decoder. Consistent with the standard architecture, the skip connection is also used to pass the features from the middle layers of the encoder to the decoder. The only difference is that the upsampled features in the decoder are concatenated with features from both encoders.

## 5 Results and Analysis

This section presents some implementation details, hardware-software information, and quantitative and qualitative results of the network on the validation set. Finally, the time consumption of the network is analyzed and several ablation studies are conducted.

### 5.1 Network parameters

The network implementation is mainly based on Pytorch version of KPConv<sup>1</sup>. Some basic parameter settings are borrowed from the Tensorflow version of KPConv's implementation<sup>2</sup> on ScanNet dataset, since there is no Pytorch implementation for ScanNet. The size of grid voxels *first\_sampling\_dl* in grid subsampling was set to 4 cm as done by (Thomas et al., 2019). The input sphere radius is chosen as 1.2 m. According to the author, the rule of thumb is to have the radius approximately 50 times bigger than *first\_sampling\_dl*, however, the GPU of the computer used in this thesis can only handle a maximum sphere radius of 30 times bigger than *first\_sampling\_dl* - the larger the radius, the more points are input. The batch size is set to 5, which is the largest batch size our GPU can handle. The network is trained with Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.98 and a weight decay of 0.001. The initial learning rate is set to 0.01 and then divided by 10 every 150 epochs. Multiple experiments showed that the network needed roughly 400 - 450 epochs to converge, which is similar to what was reported in the KPConv paper, so the maximum epoch was set to 500. Each epoch contains 500 steps, which means that 2500 spheres are processed per epoch. The remaining parameter settings follow the default settings of KPConv.

As for the 2D part, 5 views are selected for each input sphere during training and validation. The  $K$  in KNN is set to 3. The mean and standard deviations for image normalization are set to [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225], which are the default settings when use ImageNet-pretrained model provided by Pytorch. This 2D model was trained on the task of 2D semantic segmentation on our custom ScanNet dataset.

---

<sup>1</sup> <https://github.com/HuguesTHOMAS/KPConv-PyTorch>

<sup>2</sup> [https://github.com/HuguesTHOMAS/KPConv/blob/master/training\\_Scannet.py](https://github.com/HuguesTHOMAS/KPConv/blob/master/training_Scannet.py)

During training, the SGD optimizer with a momentum 0.9 and a weight decay of 0.0001 was used. The batch size was set to 32 and the network has been iterated 80,000 times.

## 5.2 Hardware and software

The network proposed in this thesis was trained and tested on a laptop with Ubuntu 16.04 operating system, a 5GB Quadro P2000 GPU and 32 GB RAM. CUDA 10.0 and CUDNN 7.6.4 were installed. All experiments were run in a mini conda environment with Python 3.6, Pytorch 1.2.0 and other dependencies. All packages and libraries needed for MVPNet and KPConv are listed in their git repositories<sup>1</sup>.

## 5.3 Validation results

In this study, Intersection over Union (IoU) is used as evaluation metric, which is quite routine for semantic segmentation tasks. As illustrated in the picture below, the IoU is the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. This measure has a scale of 0 to 100%, with 0 indicating no overlap and 100% indicating fully overlapping segmentation. The mean IoU (mIoU) for multi-class segmentation is derived by averaging the IoU of each class (Tiu, 2019).


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Figure 25 IoU, also known as Jaccard index (Wikipedia, n.d.)

It is worth noting, as mentioned before, that the network accepts spherical subclouds as input and makes predictions for the points in them. When testing, the network should be able to test each part of the scene in a regular way, instead of taking random points

---

<sup>1</sup> <https://github.com/maxjaritz/mvpnet/blob/master/environment.yml>  
<https://github.com/HuguesTHOMAS/KPConv-PyTorch/blob/master/INSTALL.md>

in the scene so that high-density regions are selected more often and thus tested more often, which is a waste of time. Therefore, the sphere picking strategy introduced in Chapter 4.3.1 is also used here. In simple terms, this solution is to assign a potential value at each point of the dataset. Whenever the network tests a sphere, the potential value of this sphere is increased so that we know that this region has been tested. The next sphere is chosen in the region of lowest potential so that each part of the dataset will be selected approximately the same number of times. In addition, the potential is increasing as a Gaussian function (the center increases the most and the increasing value decreases with distance). This means that when next time the network has to test a sphere in the same area, the same center will not be selected, so that a given point will be tested by different spheres. Eventually, a voting schema is used to determine the predicted result of a point, i.e. the result probabilities of a point is calculated as the average of the probabilities that this point obtained from different spheres<sup>1</sup>. The higher the number of votes, the less random are the results. In this study, the number of votes was set to 30, which indicates that every location of the dataset has been tested by at least 30 different input test spheres. This number provides relatively stable results while saving time.

The following Table 5.1 summarizes the performance of several fusion structures with rigid or deformable kernel. The proposed network is named as Multi-view-KPConv, i.e. **MV-KPConv**. The original MVPNet and KPConv are applied as the baseline models here. They were both trained and tested on our custom ScanNet dataset. To be fair, the original MVPNet also uses 5 input images. All tests were done on a validation set consisting of 28 scenarios.

---

<sup>1</sup> <https://github.com/HuguesTHOMAS/KPConv/issues/49>

Table 5.1 Semantic segmentation IoU scores on custom ScanNet dataset

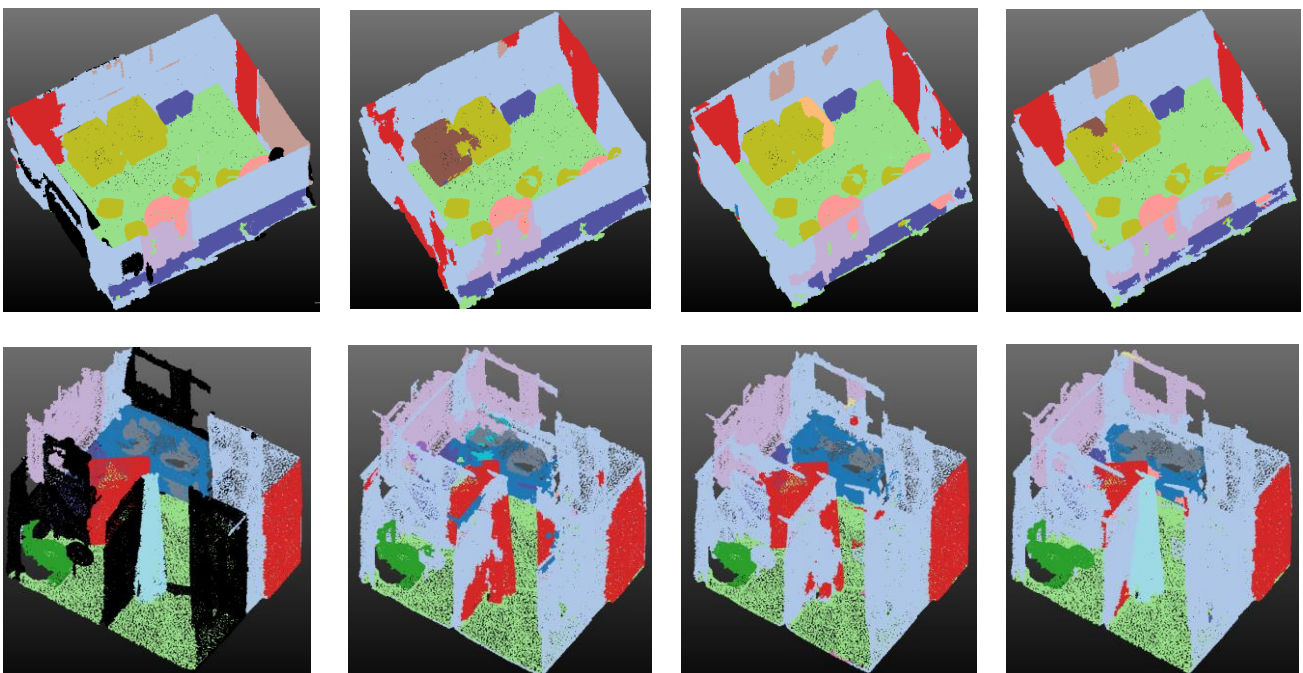
Network	Kernel	mIoU	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	fridge	shower	toilet	sink	bath	other
MVPNet	-	71.21	83.16	92.76	59.78	84.15	87.78	83.95	74.38	81.08	79.59	93.70	11.42	<b>95.46</b>	72.16	40.63	85.58	20.72	86.72	52.03	<b>68.07</b>	81.04
KPConv <sup>1</sup>	rigid <sup>2</sup>	52.58	73.15	92.05	46.05	71.23	81.67	53.22	57.50	37.98	53.85	63.87	3.76	60.82	62.21	15.32	5.51	20.34	88.09	46.55	74.34	44.03
MV-KPConv (Early fusion)	rigid	<b>74.40</b>	<b>86.01</b>	<b>93.51</b>	60.37	<b>91.21</b>	<b>90.16</b>	83.48	74.92	<b>83.09</b>	79.67	95.15	10.90	84.70	74.03	48.97	<b>88.69</b>	<b>44.02</b>	<b>90.50</b>	<b>56.51</b>	66.62	<b>85.56</b>
	deform.	72.86	85.67	93.42	61.48	91.50	89.66	80.59	<b>77.18</b>	81.04	79.48	94.72	11.98	84.40	<b>75.15</b>	45.26	78.46	43.03	83.20	52.56	63.09	85.43
MV-KPConv (Middle fusion)	rigid	73.72	85.56	93.54	60.89	88.88	89.18	83.74	76.15	82.46	<b>81.08</b>	<b>95.16</b>	11.05	86.47	74.50	<b>51.57</b>	87.05	40.09	87.66	55.19	59.79	84.37
	deform.	72.33	85.61	93.39	57.54	89.95	89.77	81.79	73.00	81.61	78.47	94.97	11.43	86.24	71.63	44.44	84.47	39.78	84.32	54.38	61.81	82.06
MV-KPConv (Late fusion)	rigid	72.18	85.25	93.13	<b>62.14</b>	88.31	90.05	<b>84.31</b>	72.43	79.75	80.49	92.91	11.74	85.38	74.56	39.39	84.64	35.33	85.90	55.74	59.69	82.45
	deform.	71.50	84.58	93.48	57.65	89.69	89.11	83.26	75.46	78.62	78.61	93.05	<b>12.36</b>	80.47	72.70	41.43	86.02	34.14	83.12	54.77	60.59	80.82

<sup>1</sup> Only used Z as additional geometric feature<sup>2</sup> In the test, the baseline KPConv with rigid kernel performs better than deformable version, so the best data are recorded here.

All fusion structures in the table above fuse 1-channel geometric features ( $Z$ ) with 64-channel image features. The ablation experiments in Section 5.5.1 explain the reason for such a design choice. Regardless of the fusion structure and kernel types, the proposed MV-KPConv's mIoU scores exceeded both baseline models. This shows that using a more powerful 3D network can indeed improve the performance of MVPNet, and that the original KPConv can benefit a lot from the fused 2D image information. Comparing the different kernel types, we can find that rigid kernel generally performs better than deformable kernel on ScanNet dataset. This is largely consistent with what is reported in the (Thomas et al., 2019). Comparing different fusion structures, we can see that the early fusion using rigid kernel has the best performance. This also confirms the advantage of early fusion, which allows the network to fully exploit the information of the raw data. The MV-KPConv has a significant improvement over the baseline model in terms of detection performance for major categories such as doors, walls, and floors. At the same time, KPConv's strong interpretation capability also allows the network to obtain better scores in minority categories, for example, the score for the shower curtain category is nearly doubled.

Some qualitative results are presented in the following images.

ignore ■ wall ■ floor ■ cabinet ■ bed ■ chair ■ sofa ■ table ■ door ■ window ■ bookshelf ■ picture ■ counter ■ desk ■ curtain ■ refrigerator ■ shower curtain ■ toilet ■ sink ■ bathtub ■ otherfurniture ■



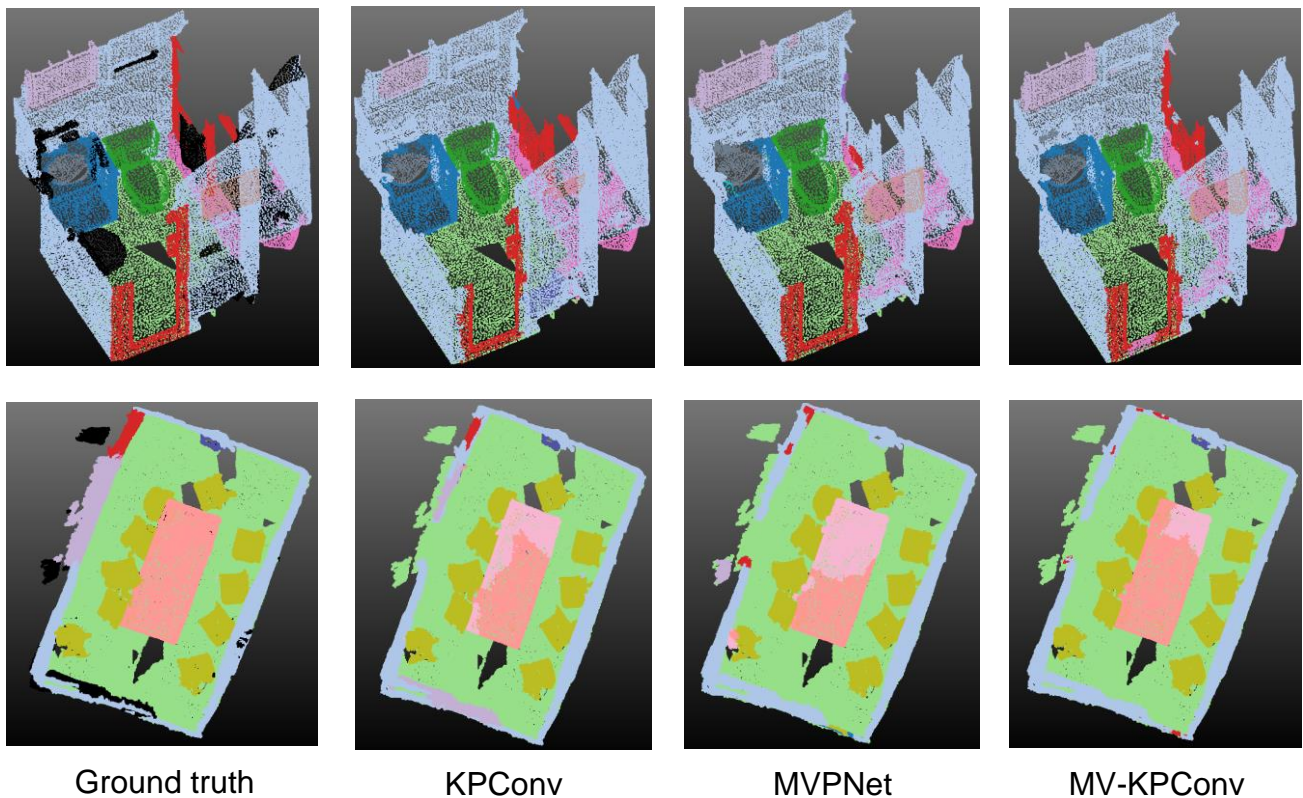


Figure 26 Visualization results of baseline models and MV-KPConv. It can be seen that compared to MVPNet, MV-KPConv can better identify minority categories, such as shower curtain. For the recognition of similarly shaped objects, such as desks and tables, MV-KPConv has a lower error rate.

#### 5.4 Computational time analysis

Figure 27 - Figure 32 show the curve of loss and training time of MV-KPConv when using rigid kernel or deformation kernel under the early fusion structure. It can be seen that the network takes 400 - 450 epochs to converge regardless of which kernel type is used. The deformed kernel version has a higher loss than the rigid kernel version. This is because in addition to the cross-entropy loss, it uses two additional regularized loss functions to enable the network to learn the locations of the kernel points correctly. The deformation kernel version requires significantly more training time than the rigid kernel version, which is related to the fact that the deformation kernel version has more parameters than the rigid kernel, as can be seen from the model size comparison in Table 5.2.



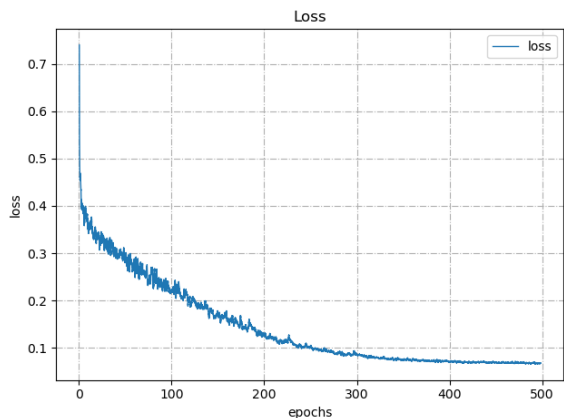


Figure 27 Loss curves of MV-KPConv early fusion versions using rigid kernel.

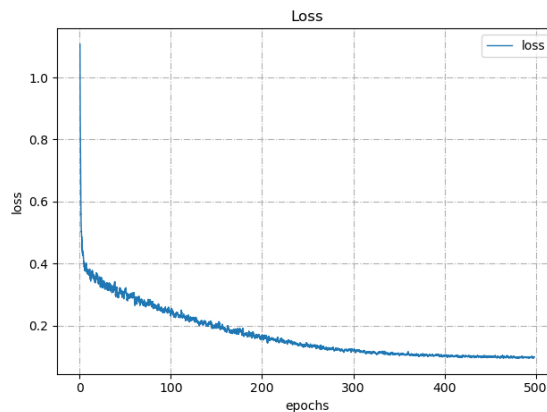


Figure 28. Loss curves of MV-KPConv early fusion versions using deformable kernel.

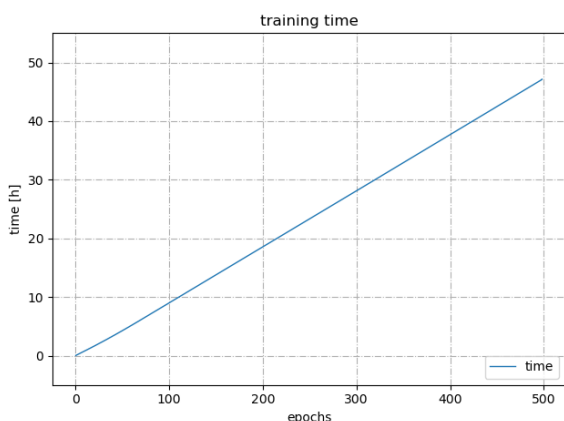


Figure 29 Training time of the early fusion version of MV-KPConv using rigid kernel

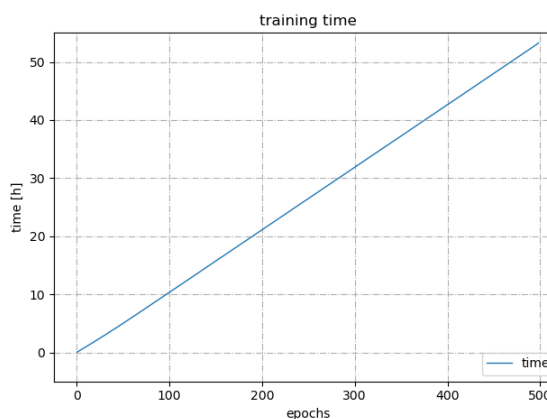


Figure 30 Training time of the early fusion version of MV-KPConv using deformable kernel.



Figure 31 mIoU variation on the validation set during training for early fusion version of MV-KPConv using rigid kernel.

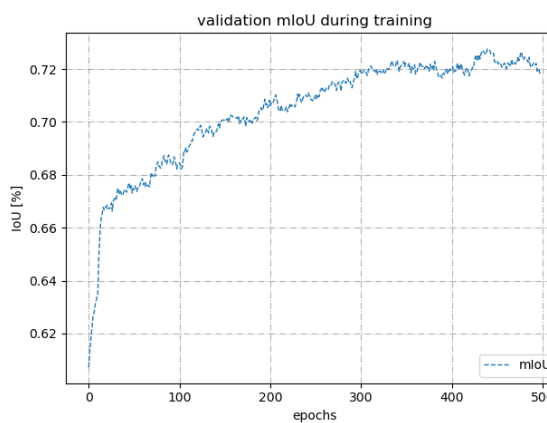


Figure 32 mIoU variation on the validation set during training for early fusion version of MV-KPConv using deformable kernel.

Table 5.2 compares the computation times of different versions of MV-KPConv and MVPNet. Both MVPNet and MV-KPConv spend a lot of time in the preprocessing stage to calculate the overlap of images and point clouds. However, MVPNet uses parallel

processes on the CPU through the multiprocessing package to speed up the computation, i.e., multiple scene data are processed simultaneously by 16 sub-processes. However, the current MV-KPConv implementation does not use multithreading on the CPU in the preprocessing stage. One of the difficulties is that in the preprocessing stage a for loop is used to downsample the point cloud scene by scene. This operation is single-threaded and runs on the GPU. The current implementation is that after downsampling the point cloud of a scene, the overlap between the images and the sampled point cloud is calculated directly. Since the time complexity of this for loop is positively related to the number of scenes to be processed, it is conceivable that the time required for the preprocessing phase will be staggering when processing larger data sets. Therefore, it is necessary to introduce multi-threading for the preprocessing stage of MV-KPConv in the future.

In the training phase, it can be noted that although MV-KPConv has better semantic segmentation capability, the time required and the volume of the model is much larger than that of MVPNet. Because KPConv has a deeper and more complex structure than PointNet, and requires more parameters. It can be noted that the late fusion takes slightly more time than early fusion. This is because an additional small linear transformation layer is added to the end of decoder in order to make the shape of the fused features acceptable to the segmented head. Moreover, due to the use of two encoders, the intermediate fusion structure has a much larger number of parameters than the other two structures, which makes the training time required much higher.

Table 5.2 Runtime and model size comparison

Step	Network	Model size [MB]	Computational time [h]
Preprocessing	MVPNet	-	2.4
	MV-KPConv	-	3.3
Training	2D Network	180	16
	MVPNet <sup>1</sup>	282	5
	KPConv	186	28
	MV-KPConv <i>rigid</i> <sup>1</sup>	457	47

<sup>1</sup> The model size of MVPNet and MV-KPConv include the 2D model size. The training time of MVPNet and MV-KPConv do not include the training time of the 2D model.

	(Early fusion)		
	MV-KPConv <i>deform</i> (Early fusion)	468	54
	MV-KPConv <i>rigid</i> (Middle fusion)	500	56
	MV-KPConv <i>deform</i> (Middle fusion)	513	63
	MV-KPConv <i>rigid</i> (Late fusion)	457	48
	MV-KPConv <i>deform</i> (Late fusion)	468	54.5
<b>Inference</b>	MVPNet	282	0.13
	MV-KPConv <i>rigid</i> <sup>1</sup> (Early fusion)	457	0.25

## 5.5 Ablation studies

To analyze the design choices and to better understand the network characteristics, some ablation experiments were conducted on the validation set. Since the early fusion version of MV-KPConv using rigid kernel performed best, the ablation studies were essentially conducted on this version.

### 5.5.1 Geometric feature

To understand whether fusing 3D geometric features is really beneficial for the network, or whether 2D image features are sufficient, MV-KPConv with, without or with partial geometric features was compared. In fact, KPConv is geometric convolution. Using only the constant 1 as a feature attached to the input points, KPConv can also differentiate the 3D shape implied by the local point cloud through the spatial relationship of neighboring points in the sphere sub-cloud (see Figure 33).

---

<sup>1</sup> Since the difference in inference time between different fusion structures is small and the early fusion version with rigid kernel performs best, only the optimal version is recorded here.

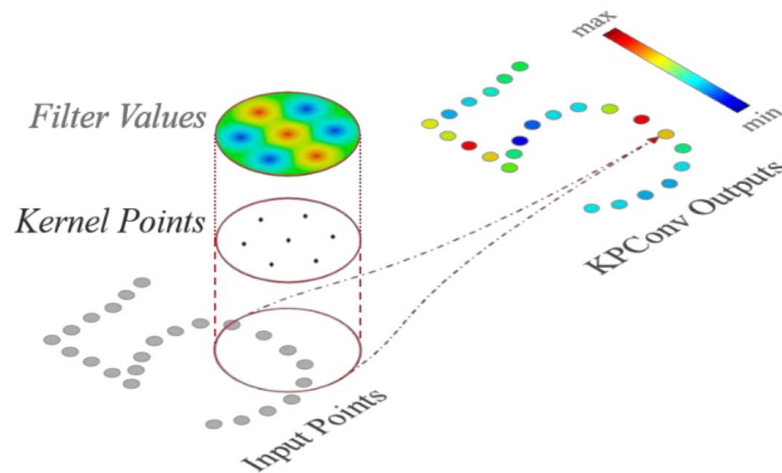


Figure 33 KPCConv illustrated on 2D points. Input points with a constant scalar feature (in grey) are convolved through a KPCConv that is defined by a set of kernel points (in black) with filter weights on each point (Thomas et al., 2019).

However, the results of the experiment showed that the MV-KPCConv does benefit from the complementary information provided by the geometric features (height) compared to the use of image features alone. Interestingly, good result can be obtained using only Z coordinates, and XY coordinates seem to be useless to the network. This can be explained by the fact that X and Y do not make any sense in a dataset where the orientation of the objects in the scene can be in any direction. However, the Z value is the height of the point and actually has a geometric meaning. It will remain the same regardless of the orientation of the dataset.

Table 5.3 Influence of geometric features

Network	mIoU
MV-KPCConv <sup>1</sup> (+ 1 + XYZ + Image features)	73.90
MV-KPCConv (+ 1 + Z + Image features)	74.40
MV-KPCConv (+ 1 + Image features)	74.22

### 5.5.2 Fusion twice

Inspired by the 3D-CVF (Yoo et al., 2020) introduced in Chapter 3.1.4, an attempt was made to perform both early and late fusion in one network. The initial idea was that perhaps the image-geometry joint features used in the early fusion stage did not contain sufficient spatial information, since the dimensionality of the geometric features is

<sup>1</sup> Constant 1 feature is to prevent black/dark points from being ignored, see chapter 4.4.

much smaller compared to the 64-channel image features. Therefore, it was chosen to concatenate the geometric feature again to the output fusion features before the segmentation head. However, the results are not satisfactory. The reason may be that the simple feature concatenation operation does not benefit the network. Because 3D-CVF actually proposes a dedicated fusion module to adaptively fuse the features a second time. Also, the appropriate timing of the second fusion needs to be further investigated, because 3D-CVF does not directly and brutally feed the second fused features into the network head, but uses a refinement network to further process the fused information. The inspiration given by this experiment is that complex fusion structures require refined network design and special fusion methods, which are determined only through extensive experiments and studies.

*Table 5.4 Fusion twice compared to early fusion*

Network	mIoU
MV-KPConv (Early fusion)	74.40
MV-KPConv (Early + Late fusion)	73.67

### 5.5.3 3D point color

To investigate the effect of point cloud color, an additional 3-channel RGB feature was fused in an early stage. It is noticed that the use of point cloud colors has instead a negative effect on the network. This may be because ScanNet's point cloud is obtained by 3D reconstruction of images and the point cloud color is also derived from the images. However, the 2D network has already processed the color information and it is confusing for the network if we pass the unprocessed color information to the 3D network as well at this point.

*Table 5.5 Influence of point cloud color*

Network	mIoU
MV-KPConv (+ 1 + Z + Image features)	74.40
MV-KPConv (+ 1 + Z + RGB + Image features)	73.36

### 5.5.4 Number of views

In the proposed network, 5 images are selected for each input sphere. This number was chosen based on the results of the ablation study in the MVPNet paper (Jaritz et al., 2019). Since MVPNet is based on PointNet implementation, the input to the 3D

network is a point cloud chunk of  $1.5 m \times 1.5 m \times scene\ height$ . They studied the coverage of an input point cloud chunk with different number of images and found that 5 views could already cover 97.45% of the chunk area. In our experiments, the radius of the input sphere was chosen to be  $1.2 m$ . Because the volume of the input sphere is similar to the volume of the chunk in the MVPNet, and more views means more computations and more memory budget. Therefore, 5 images are selected by default to balance performance and effect. Since the coverage of 5 images is already large, increasing to 6 images hardly changes score. This was also verified by the experimental results. However, it is conceivable that using a larger number of images such as 10 can result in a slightly better score, however this comes at the cost of more calculations. It is worth noting that testing with 6 images is the maximum number that the hardware used can handle. The maximum number of images that our GPU can handle when training the network is 5. This is another reason why 5 was chosen as the default number.

Table 5.6 Impact of the number of images

Network	mIoU
MV-KPConv (5 Images)	74.40
MV-KPConv (6 Images)	74.40

## 6 Conclusions and Future Works

This study is devoted to improving the performance of 3D point cloud semantic segmentation by using multimodal fusion with 2D images. To this end, the performance of an open-source multimodal algorithm is enhanced by introducing a more powerful 3D network. Based on the experimental results in the previous chapter, conclusions and possibilities for future optimization of the network will be discussed in this chapter.

### 6.1 Conclusions

In this section, conclusions will be drawn by answering the research questions defined in Chapter 1.2.

#### ***Is it possible to improve MVPNet by changing its 3D network from PointNet to KPConv?***

The experimental results demonstrate that using more powerful 3D networks can improve the semantic segmentation capability of MVPNet. This is made possible by the clear hierarchical structure of MVPNet. First, a separate pre-trained 2D network is used to extract image features. Since the 2D network is not involved in 3D training, this allows it to be flexibly integrated into the workflow of a different 3D network. Furthermore, the 2D network is designed to adapt the decoder so that the output feature map is the same size as the input image, which allows each pixel to have its own features, thus making 2D - 3D feature lifting possible through the established pixel-dot mapping.

Secondly, the whole 2D – 3D feature lifting method provided by MVPNet (including image selection, feature aggregation) is related to the 2D network but independent of the 3D network. Because as mentioned earlier, the feature transfer presupposes that the 2D network outputs a feature map of the same size as the input image. At the same time, the output of the whole 2D-3D lifting module is a point cloud augmented by features, which is not fundamentally different from a color point cloud: except that each point has 64 channels of image semantic features instead of 3 channels of RGB features. Such feature-augmented point clouds are a widely accepted format by 3D networks such as KPConv.

Finally, the core feature aggregation module of the 2D-3D lifting method proposed by MVPNet is differentiable, which means that when it is integrated into KPConv, its weights can be learned by backpropagation according to the loss function of KPConv, and thus adaptively aggregates the 2D features best suited for the 3D network.

From the above points we can draw an inference that the feature lifting method of MVPNet is not only applicable to KPConv, but also to other 3D networks. In terms of implementation, some main works we may need to do are, for example, paying attention to some details of point cloud preprocessing by different networks, the format of the input and output feature tensor, and rewriting the data loader of the 3D network so that it can load 2D images, etc.

***How much better does the proposed MV-KPConv perform compare to the baseline model in terms of mIoU and IoU on class doors and windows?***

On the custom ScanNet validation set, MV-KPConv achieves the best score of 74.40 mIoU. This score is 3.19 mIoU higher than MVPNet, and 21.82 mIoU higher than the original KPConv. In the recognition of door categories, MV-KPConv is 2.01 IoU higher than MVPNet. However, the difference between two networks is not significant for the recognition of windows. In addition, MV-KPConv has a greater improvement in the recognition of large objects such as walls, floors and beds. For minority categories, such as shower curtain, the proposed network also achieves good scores.

***How do early, middle, and late fusions perform on MV-KPConv respectively? Does using two types of fusion in the network at the same time improve network performance?***

The experimental data in Table 5.1 demonstrate that the mIoU scores of all three MV-KPConv fusion structures surpass the baseline model. Comparing the three fusion structures, it can be found that early fusion is more advantageous than intermediate and late fusion. Firstly, in the result of semantic segmentation, the early fusion version has the highest score, followed by the middle fusion version and finally the late fusion. This demonstrates that learning the joint features of multiple modalities at an early stage can make full utilization of the information from the raw data. Secondly, in terms of computation time, the early fusion version jointly handles the features of both modalities, making it the least demanding in terms of computation and low memory budget. The ablation experiments in Chapter 5.5.2 demonstrate that it is not beneficial to simply mimic other literature and use two fusions in the proposed network. This is related to



the different designs, structures, and principles of the different networks, and requires in-depth studies and experiments to determine the optimal fusion time point.

***To what extent do 3D point cloud colors affect the proposed network that already fuse 2D image?***

Fusing point cloud colors while using 2D image semantic features has only negative effects for the proposed network. The reason for this is explained in Section 5.5.3. In simple terms, the sparse RGB features attached to the point cloud are derived from the pictures. This is because the point cloud of ScanNet is obtained from the 3D reconstruction of RGB images. The feature maps output from the 2D network already contain rich color and texture information, and adding the unprocessed color information again at this point actually pollutes the input features and causes confusion.

***Is the 2D – 3D lifting method provided by MVPNet feasible for different 2D input format?***

As mentioned earlier, the 2D-3D lifting method provided by MVPNet requires image formats that are related to the image formats accepted by the 2D network. The current 2D network only accepts the normal field of view (NFoV) images. If the input image format is a panorama or fisheye view, then it needs to be mapped to an NFoV image first, a related conversion method is provided in (Sun, 2020). Once we have the RGB images in normal view and their corresponding depth maps, we can project them to 3D space to form dense point cloud. Subsequently, we can normally follow the lifting methods provided by MVPNet for overlap calculation, image selection, feature aggregation, etc. If the panorama or fisheye image does not have the corresponding depth information, then we may need a deep learning network to predict the depth of the image so that it can be mapped to 3D space. The keyword here is monocular depth estimation, such as (Yang et al., 2018).

## 6.2 Discussion

### 6.2.1 Other possible workflows

The workflow of MV-KPConv presented in this study actually underwent three iterations. The initial network design was informed by the recommendations<sup>1</sup> of the author of MVPNet. First, 30 images were selected for a whole scene. After that, these images were processed using the pre-trained 2D network. Using the feature aggregation module, a feature augmented point cloud of this scene was obtained, which was then stored on the hard disk. Finally, this scene point cloud with 64-channels image features attached is used as the input to KPConv. This workflow looks very intuitive, but there are problems with the implementation. First, the feature aggregation module contains MLPs need to learn weights. If the untrained feature aggregation module is used directly, the resulting semantic features will be meaningless. Second, as mentioned in Section 4.4.2, the weights inside the feature aggregation module are updated according to the loss function of the 3D network. Taking MVPNet as an example, a backpropagation starts from the bottom of PointNet and updates the weights all the way up through the feature aggregation module, stopping in front of the frozen 2D network. If we follow the workflow suggested by the author of MVPNet and use the feature aggregation module trained by PointNet to generate feature augmented point clouds, it can be found that such a MV-KPConv can obtain a score of 70.1 mIoU with rigid kernel, but a very poor score with deformation kernel. The reason for this is that the PointNet and the rigid version of KPConv use the same cross-entropy loss function, while the deformation kernel version uses two additional loss functions to control the shift of the kernel. Thus, the aggregated 2D features make no sense for KPConv which uses a deformation kernel. This is clearly a bad implementation.

To solve this problem, in the second version of MV-KPConv, the feature aggregation module is added to the forward function of the network so that it can learn the weights correctly according to the loss function of KPConv. The optimal result obtained in this version is 70.39 mIoU. However, the 2D network and other modules of the feature lifting method, such as the *GatherNeighborPointsByIndices* function (see Section 4.4), etc., are not combined into the forward function. These modules are placed in a sepa-

---

<sup>1</sup> <https://github.com/maxjaritz/mvpnet/issues/3>

rate script and their outputs are stored on disk as sequence files, which are then available to the KPConv data loader. This implementation makes the whole workflow inflexible and wastes a lot of disk space. Meanwhile, the score we obtained is slightly higher than the previous version, but the core idea remains the same, which is to select 30 images for the whole scene as additional information for the 3D semantic segmentation. And this score is still lower than MVPNet's 71.21 mIoU, so we need a new implementation idea.

Eventually, the workflow proposed in this thesis was adopted. The pre-trained 2D network and the 2D-3D lifting module are integrated into the MV-KPConv network in its entirety. Also, the data loader was rewritten to make it possible to select 5 images for each input sphere sub-cloud in real time. This implementation allows for a larger number of images to be used per scene compared to 30 images per scene. Moreover, the coverage of each image is more accurate, bringing more useful texture information. Such an implementation allows the proposed network to exceed the baseline model, reaching 74.40 mIoU on our validation set.

### 6.2.2 Better fusion methods and better 3D networks

At the time of writing this thesis, a new adaptive cross-modal learning network (Jaritz et al., 2021) was found to be proposed by the author of MVPNet. The architecture of the first half of this network is very similar to MVPNet, however, in the second half of the network, he designed a mechanism to make the output features of the 2D and 3D networks learn from each other by mimicking each other (see Figure 34).

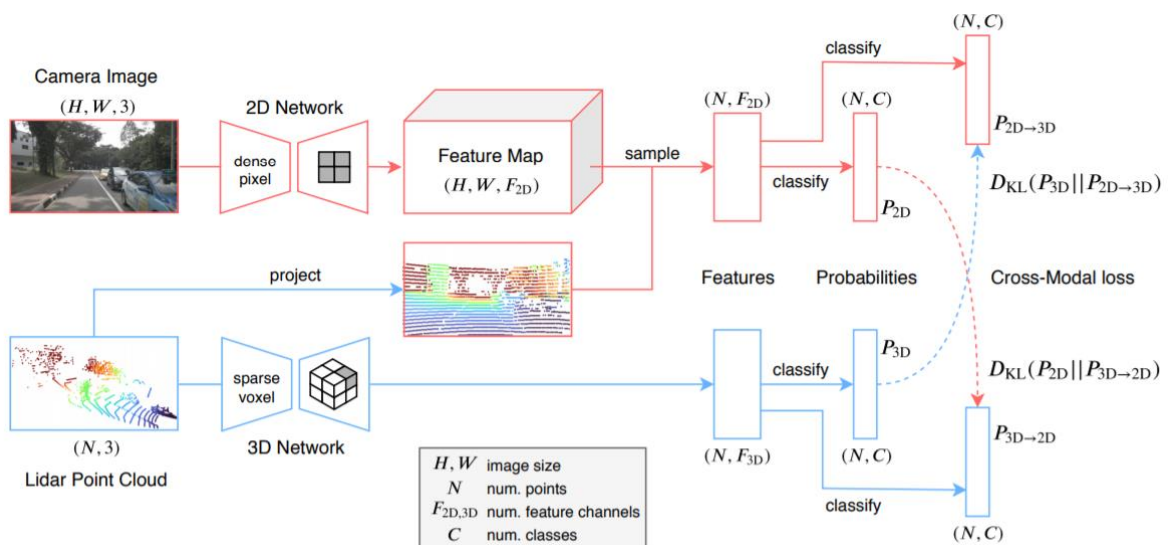


Figure 34 Pipeline of xMUDA network (Jaritz et al., 2021)

In the final stage of the network, before the detection head, he does not simply concatenate and fuse the features of the two modalities, but makes the individual unimodal features obtain complementary information from the cascaded features through the mimicry mechanism (see Figure 35).

The inspiration from this recent work is that adaptive fusion methods are more efficient and provide more effective complementary information than simple concatenation operations. This is also a trend in multimodal fusion research. At the same time, it can be noted that he used SparseConvNet (Graham et al., 2018) as a 3D backbone in this new study. In fact, SparseConvNet is ranked higher than KPConv on the ScanNet benchmark. Therefore, in the future, it would make sense to use this better 3D network for fusion features.

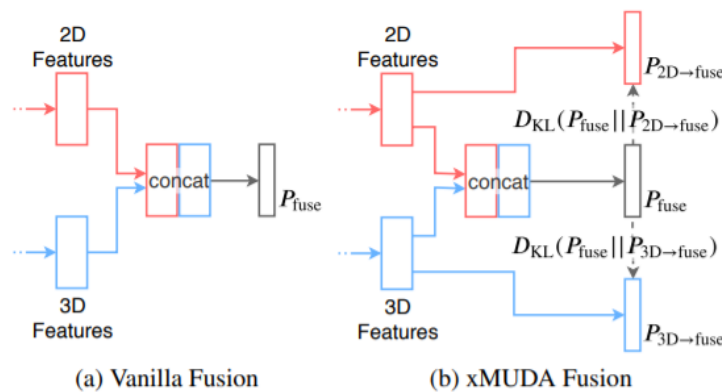


Figure 35 Instead of simply joining and fusing the features of two modalities, xMUDA enables each unimodal feature to obtain complementary information from the cascaded features through an imitation mechanism (Jaritz et al., 2021).

### 6.2.3 When to fuse

It is not simple to answer this question - no conclusive evidence can be found in the literature review in Chapter 3 that one fusion method is better than the other. Performance depends heavily on the data and network structure. In addition, the fusion structure designed in this paper is more based on intuitive and empirical results and does not delve into the way the network works and how the layers in it behave in the face of different fusion approaches. As suggested in the (Feng et al., 2021), the network structure design can be optimized in the future by visual analysis techniques, similar methods such as (Liu et al., 2017), for understanding image classification in CNNs. Such visualization tools can help to understand and analyze how the network operates, diagnose problems, and ultimately improve the network structure.

### 6.3 Contributions

This thesis introduces the following contributions:

- The proposed fusion network meaningfully combines the modules of MVPNet and KPConv, and its 3D semantic segmentation performance outperforms these two original networks.
- It is demonstrated that the 2D-3D feature lifting method provided by MVPNet is applicable to different 3D networks and has the potential to accept different 2D image input formats.
- A detailed literature review summarizes the STOA multimodal fusion methods and gives a table of their respective characteristics.
- The impact of different fusion structure designs on the network is investigated, and the performance of the network is further improved by selecting a suitable fusion structure.
- Implemented a Pytorch version of KPConv's ScanNet data loader, filling the gap in the public git repository.

### 6.4 Limitations and future works

One of the limitations of the proposed method comes from the hardware. The Quadro P2000 used for training and testing is a good GPU, but its 5GB of memory makes it not an optimal solution for deep learning. This prevents the recommended values of the KPConv hyperparameters from being set. For example, the input sphere radius of the network is set to 1.2 m when using a rigid kernel because of GPU memory limitations. With the deformation kernel, the sphere radius can only be set to a maximum of 1 m, while the recommended radius is 2 m. This may also be the reason why the deformation kernel version generally performs less well than the rigid kernel. Smaller input spheres may result in some objects not being included more completely. The mistake of misidentifying a part of the cabinet edge as a table in Figure 34 may be due to this reason. Therefore, using a GPU with larger memory to test the network may be able to further improve the performance of the network in the future.

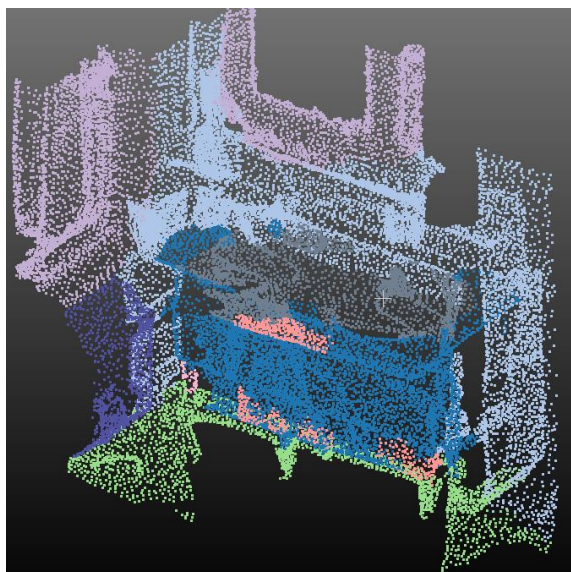


Figure 36 Misidentification of part of the edge of a cabinet as a table



Figure 37 Original Mesh

Another of the more common drawbacks of MV-KPConv is its lower accuracy for counter recognition comparing to MVPNet. This can be seen in the following pictures and in Table 5.1. In the future, it will be interesting to study improvements for this phenomenon.

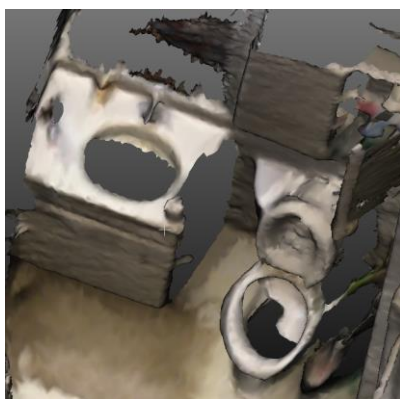


Figure 38 Original mesh

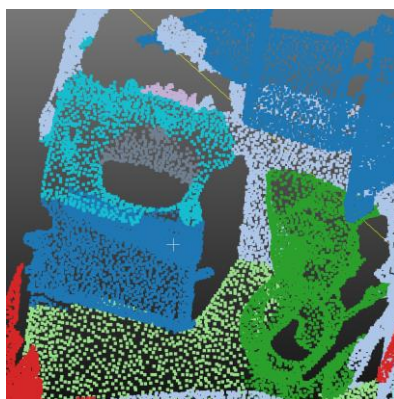


Figure 39 MVPNet (counter in light blue)

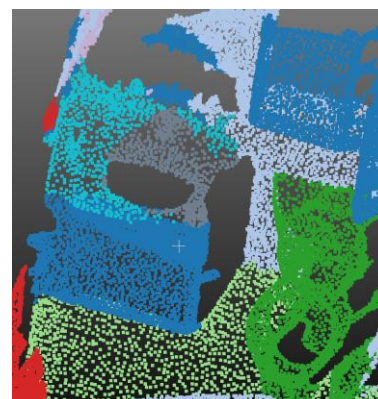


Figure 40 MV-KPConv (counter in light blue)

Furthermore, as mentioned in Section 5.4, the algorithm implementation of MV-KPConv in the preprocessing stage is not good and requires a lot of time to calculate the overlap between the photos and the point cloud. A possible optimization would be to introduce multithreading here, allowing the program to process multiple scene data at the same time.

Finally, the proposed method in this thesis has only been trained and tested on the indoor small-scale scene datasets. However, the performance of the network in the

---

face of large-scale scenes is still unknown. Therefore, in the future, it would be interesting to study the behavior of the proposed method in datasets with such characteristics, such as the Stanford 2D-3D-Semantics dataset (Armeni et al., 2017).

---

## References

- Armeni, I., Sax, S., Zamir, A. R., & Savarese, S. (2017). *Joint 2D-3D-Semantic Data for Indoor Scene Understanding*. <http://arxiv.org/abs/1702.01105>
- Atrey, P. K., Hossain, M. A., El Saddik, A., & Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis: A survey. In *Multimedia Systems* (Vol. 16, Issue 6). <https://doi.org/10.1007/s00530-010-0182-0>
- Bello, S. A., Yu, S., Wang, C., Adam, J. M., & Li, J. (2020). Review: Deep learning on 3D point clouds. *Remote Sensing*, 12(11), 1–34. <https://doi.org/10.3390/rs12111729>
- Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9), 509–517. <https://doi.org/10.1145/361002.361007>
- Braun, A. (2020). *Automated BIM-based construction progress monitoring by processing and matching semantic and geometric data*. November, 150.
- Cai, Z., & Vasconcelos, N. (2018). Cascade R-CNN: Delving into High Quality Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6154–6162. <https://doi.org/10.1109/CVPR.2018.00644>
- Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). *Rethinking Atrous Convolution for Semantic Image Segmentation*. <http://arxiv.org/abs/1706.05587>
- Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua*, 6526–6534. <https://doi.org/10.1109/CVPR.2017.691>
- Chiang, H. Y., Lin, Y. L., Liu, Y. C., & Hsu, W. H. (2019). A Unified Point-Based Framework for 3D Segmentation. *Proceedings - 2019 International Conference on 3D Vision, 3DV 2019, Table 2*, 155–163. <https://doi.org/10.1109/3DV.2019.00026>



- Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., Li, Y., & Cao, D. (2021). Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review. *IEEE Transactions on Intelligent Transportation Systems*, 1–19. <https://doi.org/10.1109/TITS.2020.3023541>
- Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). ScanNet: Richly-annotated 3D reconstructions of indoor scenes. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua*, 2432–2443. <https://doi.org/10.1109/CVPR.2017.261>
- Dai, Y., Gieseke, F., Oehmcke, S., Wu, Y., & Barnard, K. (2021). *Attentional Feature Fusion*. 3559–3568. <https://doi.org/10.1109/wacv48630.2021.00360>
- Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W., & Dietmayer, K. (2021). Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3), 1341–1360. <https://doi.org/10.1109/TITS.2020.2972974>
- Google, C. (2014). *Xception: Deep Learning with Depthwise Separable Convolutions*.
- Graham, B., Engelcke, M., & Van Der Maaten, L. (2018). 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 9224–9232. <https://doi.org/10.1109/CVPR.2018.00961>
- Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2020). Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. <https://doi.org/10.1109/tpami.2020.3005434>
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2), 386–397. <https://doi.org/10.1109/TPAMI.2018.2844175>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-Decem*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- He, L. (2019). *Improving 3D Point Cloud Segmentation Using Multimodal Fusion of Projected 2D Imagery Data*.

- Hermosilla, P., Ritschel, T., Vázquez, P. P., Vinacua, À., & Ropinski, T. (2018). Monte Carlo convolution for learning on non-uniformly sampled point clouds. *SIGGRAPH Asia 2018 Technical Papers, SIGGRAPH Asia 2018*, 37(6). <https://doi.org/10.1145/3272127.3275110>
- Hou, J., Dai, A., & Niebner, M. (2019). 3D-SIS: 3D semantic instance segmentation of RGB-D scans. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June*, 4416–4425. <https://doi.org/10.1109/CVPR.2019.00455>
- Huang, T., Liu, Z., Chen, X., & Bai, X. (2020). EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12360 LNCS, 35–52. [https://doi.org/10.1007/978-3-030-58555-6\\_3](https://doi.org/10.1007/978-3-030-58555-6_3)
- Jaritz, M., Gu, J., & Su, H. (2019). Multi-view pointnet for 3D scene understanding. *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, 3995–4003. <https://doi.org/10.1109/ICCVW.2019.00494>
- Jaritz, M., Vu, T., de Charette, R., Wirbel, É., & Pérez, P. (2021). *Cross-modal Learning for Domain Adaptation in 3D Semantic Segmentation*. 1–15. <http://arxiv.org/abs/2101.07253>
- Kundu, A., Yin, X., Fathi, A., Ross, D., Brewington, B., Funkhouser, T., & Pantofaru, C. (2020). Virtual Multi-view Fusion for 3D Semantic Segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12369 LNCS, 518–535. [https://doi.org/10.1007/978-3-030-58586-0\\_31](https://doi.org/10.1007/978-3-030-58586-0_31)
- Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June*, 12689–12697. <https://doi.org/10.1109/CVPR.2019.01298>
- Liang, M., Yang, B., Wang, S., & Urtasun, R. (2018). Deep Continuous Fusion for Multi-sensor 3D Object Detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*

- Bioinformatics*), 11220 LNCS, 663–678. [https://doi.org/10.1007/978-3-030-01270-0\\_39](https://doi.org/10.1007/978-3-030-01270-0_39)
- Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., & Liu, S. (2017). Towards Better Analysis of Deep Convolutional Neural Networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1), 91–100. <https://doi.org/10.1109/TVCG.2016.2598831>
- Lu, H., & Shi, H. (2020). *Deep Learning for 3D Point Cloud Understanding: A Survey*. <http://arxiv.org/abs/2009.08920>
- Madali, N. (2021). *Point Cloud Sampling*. <https://medium.com/@nabil.madali/point-cloud-sampling-8e74a6e48743>
- Pang, S., Morris, D., & Radha, H. (2020). CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. *IEEE International Conference on Intelligent Robots and Systems*, 10386–10393. <https://doi.org/10.1109/IROS45743.2020.9341791>
- Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). *ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation*. 1–10. <http://arxiv.org/abs/1606.02147>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85), 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>
- Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum PointNets for 3D Object Detection from RGB-D Data. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 918–927. <https://doi.org/10.1109/CVPR.2018.00102>
- Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua*, 77–85. <https://doi.org/10.1109/CVPR.2017.16>
- Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems, 2017-Decem*, 5100–5109.

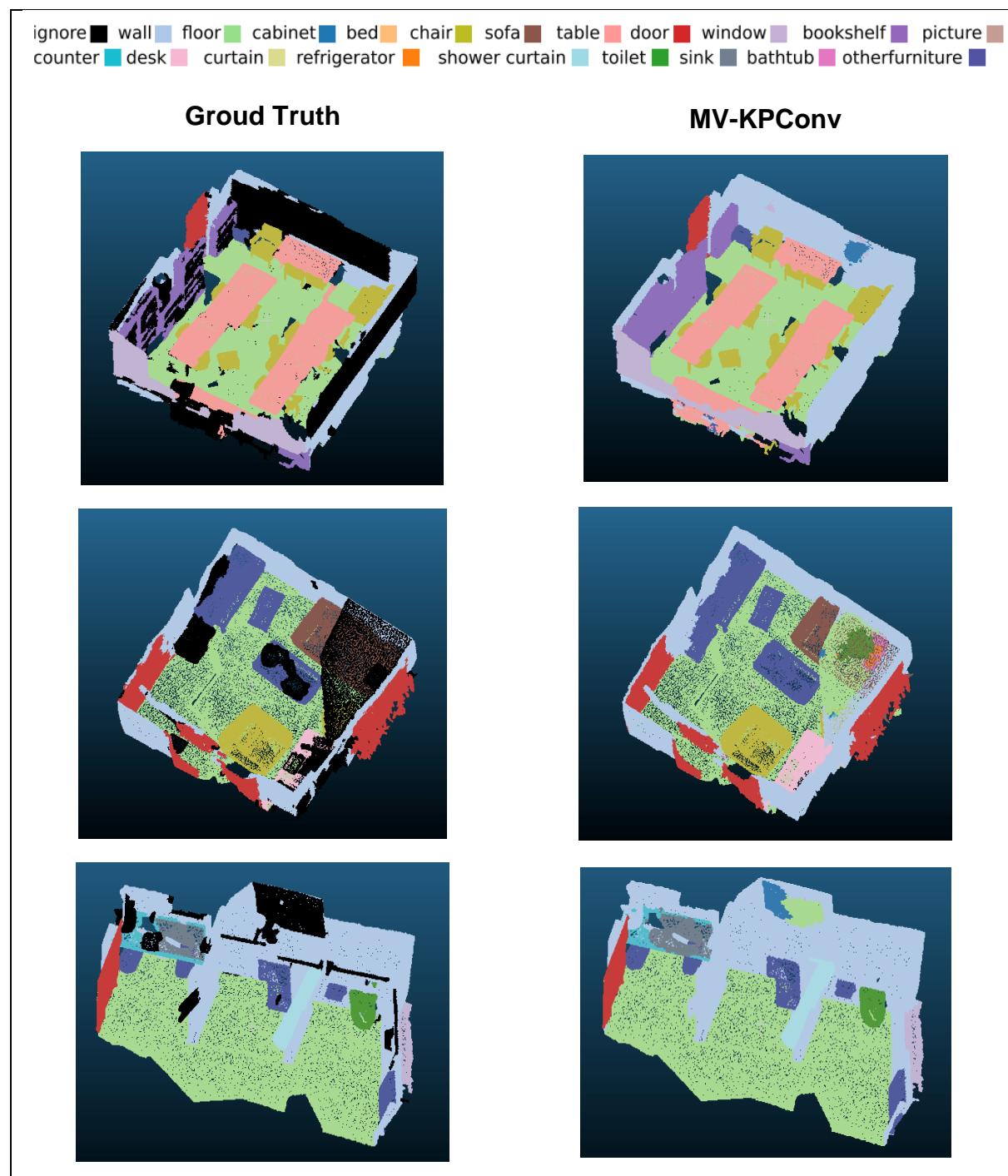
- Redmon, J., & Farhadi, A. (2018). YOLO v.3. *Tech Report*, 1–6.  
<https://pjreddie.com/media/files/papers/YOLOv3.pdf>
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149.  
<https://doi.org/10.1109/TPAMI.2016.2577031>
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention -- MICCAI 2015* (pp. 234–241). Springer International Publishing.
- Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). PV-RCNN: Point-voxel feature set abstraction for 3D object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 10526–10535. <https://doi.org/10.1109/CVPR42600.2020.01054>
- Shi, S., Wang, X., & Li, H. (2019). PointRCNN: 3D object proposal generation and detection from point cloud. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019-June*, 770–779.  
<https://doi.org/10.1109/CVPR.2019.00086>
- Snoek, C. G. M., Worring, M., & Smeulders, A. W. M. (2005). Early versus late fusion in semantic video analysis. *Proceedings of the 13th ACM International Conference on Multimedia, MM 2005, January*, 399–402.  
<https://doi.org/10.1145/1101149.1101236>
- Stojanovic, V., Trapp, M., Richter, R., Hagedorn, B., & Döllner, J. (2020). *Semantic Enrichment of Indoor Point Clouds An Overview of Progress towards Digital Twinning*. 2(Grieves 2014), 809–818. [https://doi.org/10.5151/proceedings-ecaadesigradi2019\\_051](https://doi.org/10.5151/proceedings-ecaadesigradi2019_051)
- Sun, J. (2020). *3D Point Cloud Object Detection Based on 2D ObjectDetection*. 1619418.
- Tan Qu, & Wei Sun. (2015). Usage of 3D Point Cloud Data in BIM (Building Information Modelling): Current Applications and Challenges. *Journal of Civil Engineering and Architecture*, 9(11), 1269–1278. <https://doi.org/10.17265/1934-7359/2015.11.001>

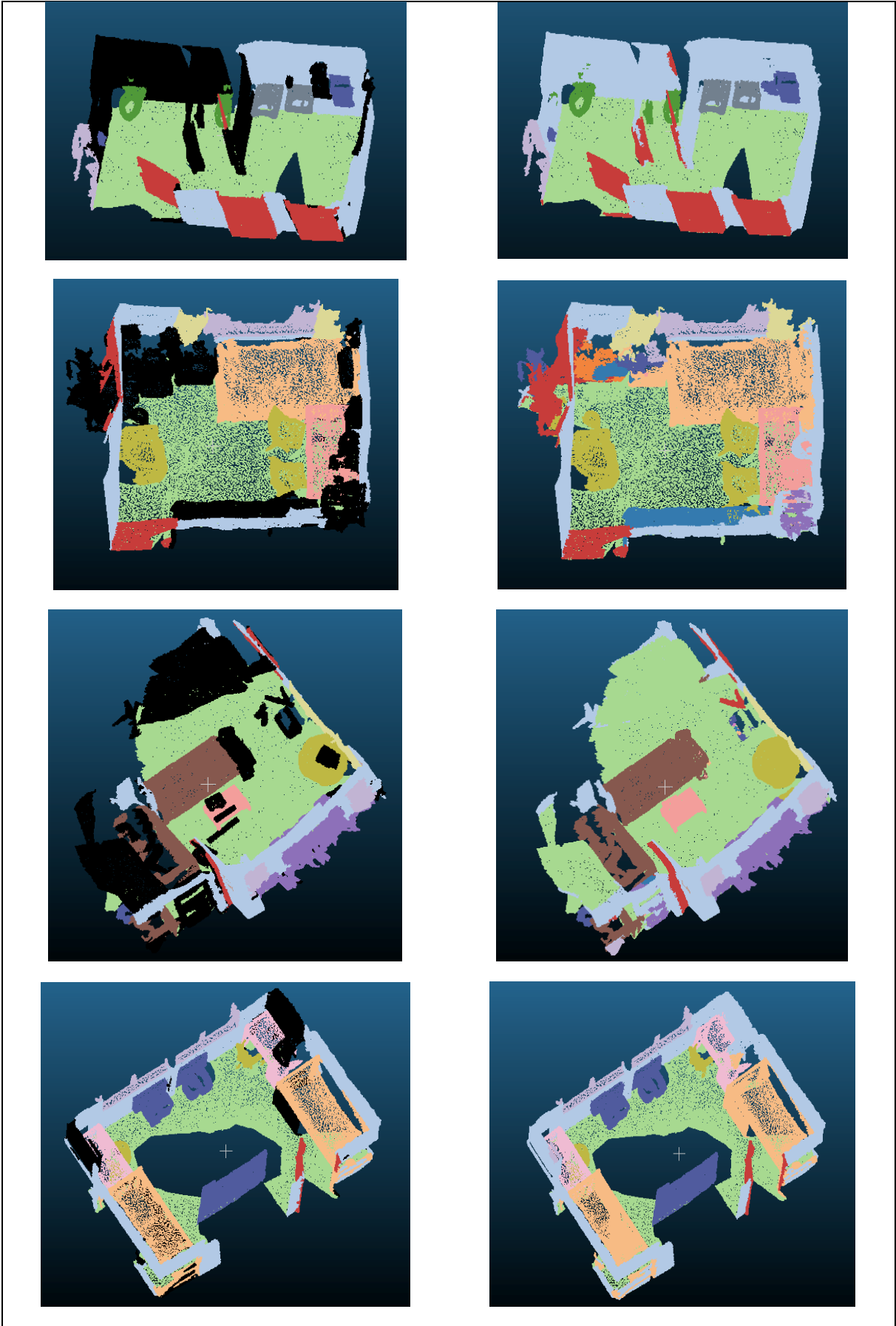
- Thomas, H. (2020). *Learning new representations for 3D point cloud semantic segmentation* To cite this version : HAL Id : tel-02458455 Apprentissage de nouvelles représentations pour la sémantisation de nuages de points 3D *Learning new representations for 3D point cloud sema.*
- Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., & Guibas, L. (2019). KPConv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision, 2019-October*, 6410–6419. <https://doi.org/10.1109/ICCV.2019.00651>
- Tiu, E. (2019). *Metrics to Evaluate your Semantic Segmentation Model*. Towards Data Science. <https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>
- Vega Torres, M. A., Borrmann, A., Braun, A., Bauer, H., & Noichl, F. (2020). *Efficient Vertical Object Detection in Large High-Quality Point Clouds of Construction Sites*. October.
- Vora, S., Lang, A. H., Helou, B., & Beijbom, O. (2020). Pointpainting: Sequential fusion for 3D object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4603–4611. <https://doi.org/10.1109/CVPR42600.2020.00466>
- Wahbeh, W., Kunz, D., Hofmann, J., & Bereuter, P. (2020). Digital twinning of the built environment-an interdisciplinary topic for innovation in didactics. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 5(4), 231–237. <https://doi.org/10.5194/isprs-Annals-V-4-2020-231-2020>
- Wang, Z., & Jia, K. (2019). Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal. *IEEE International Conference on Intelligent Robots and Systems*, 1742–1749. <https://doi.org/10.1109/IROS40897.2019.8968513>
- Wikipedia, F. (n.d.). *Jaccard index*. Sciences-New York. [https://en.wikipedia.org/wiki/Jaccard\\_index](https://en.wikipedia.org/wiki/Jaccard_index)
- Xie, Y., Tian, J., & Zhu, X. X. (2020). Linking Points with Labels in 3D: A Review of Point Cloud Semantic Segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4), 38–59. <https://doi.org/10.1109/MGRS.2019.2937630>

- Xu, S., Zhou, D., Fang, J., Yin, J., Bin, Z., & Zhang, L. (2021). *FusionPainting: Multimodal Fusion with Adaptive Attention for 3D Object Detection*.  
<http://arxiv.org/abs/2106.12449>
- Yang, Y., Jin, S., Liu, R., Kang, S. B., & Yu, J. (2018). Automatic 3D Indoor Scene Modeling from Single Panorama. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3926–3934.  
<https://doi.org/10.1109/CVPR.2018.00413>
- Yoo, J. H., Kim, Y., Kim, J., & Choi, J. W. (2020). 3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12372 LNCS, 720–736. [https://doi.org/10.1007/978-3-030-58583-9\\_43](https://doi.org/10.1007/978-3-030-58583-9_43)
- Zhang, D. (2011). Camera calibration with a near-parallel imaging system based on geometric moments. *Optical Engineering*, 50(2), 023601.  
<https://doi.org/10.1117/1.3533032>
- Zhang, Y., Sidibé, D., Morel, O., Mériaudeau, F., Zhang, Y., Sidibé, D., Morel, O., & Mériaudeau, F. (2020). *Deep multimodal fusion for semantic image segmentation : A survey To cite this version : HAL Id : hal-02963619 Deep Multimodal Fusion for Semantic Image Segmentation : A Survey*.
- Zhao, X., Liu, Z., Hu, R., & Huang, K. (2019). 3D object detection using scale invariant and feature reweighting networks. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 9267–9274.  
<https://doi.org/10.1609/aaai.v33i01.33019267>
- Zhou, Y., & Tuzel, O. (2018). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4490–4499.  
<https://doi.org/10.1109/CVPR.2018.00472>

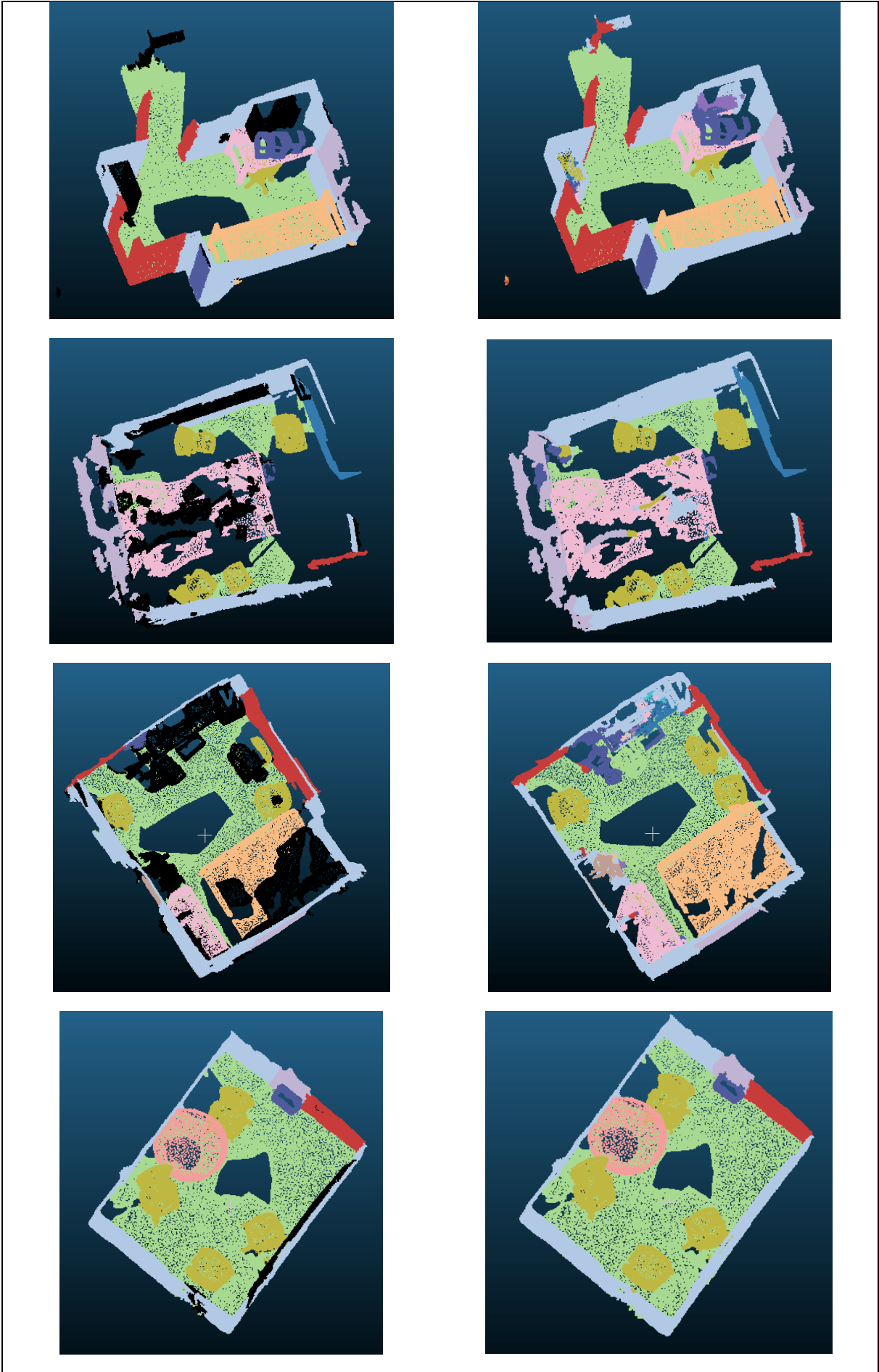
## Appendix A

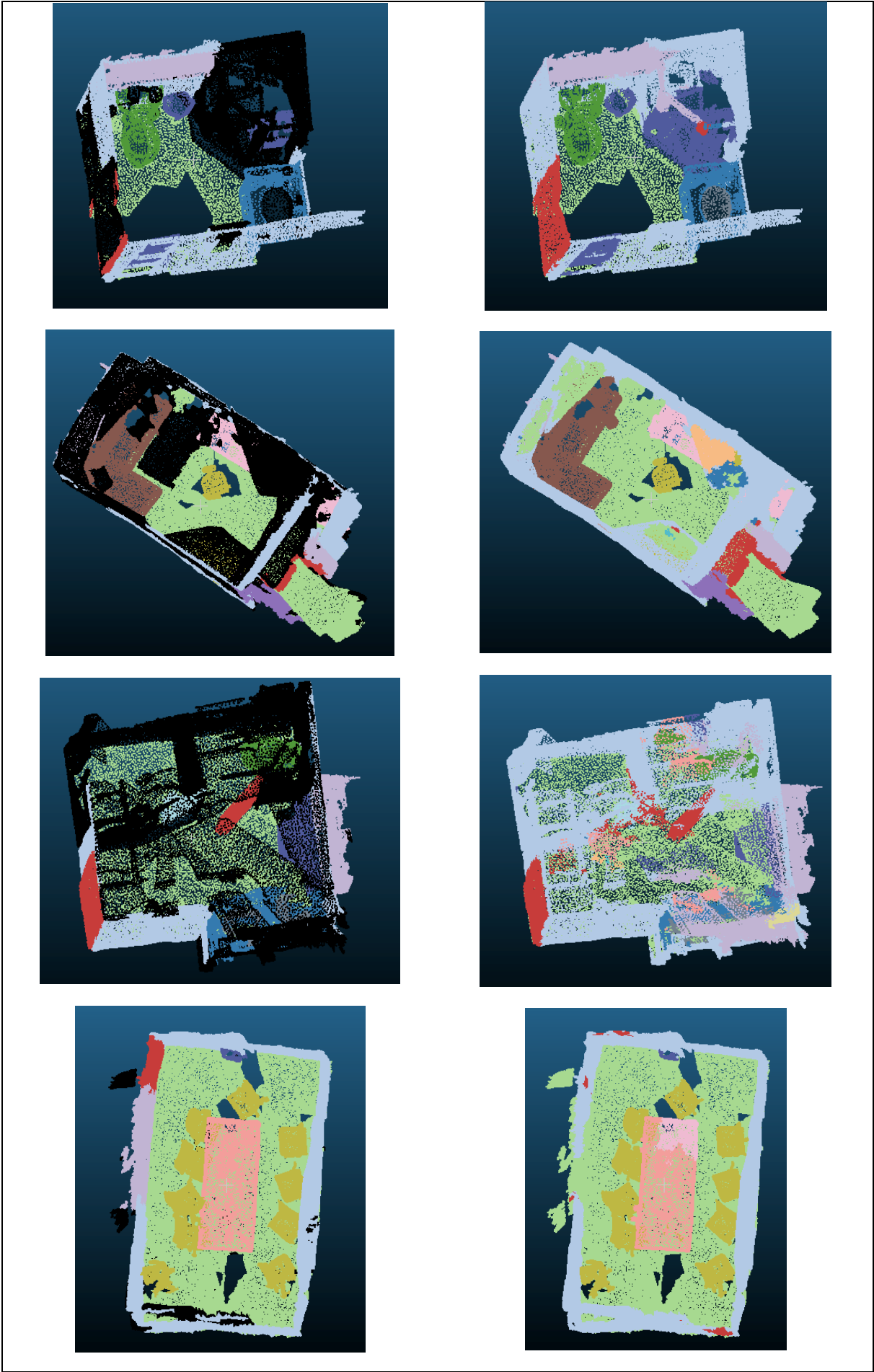
Table 6.1 Visualization results of 3D semantic segmentation by MV-KPConv on 28 validation scenarios

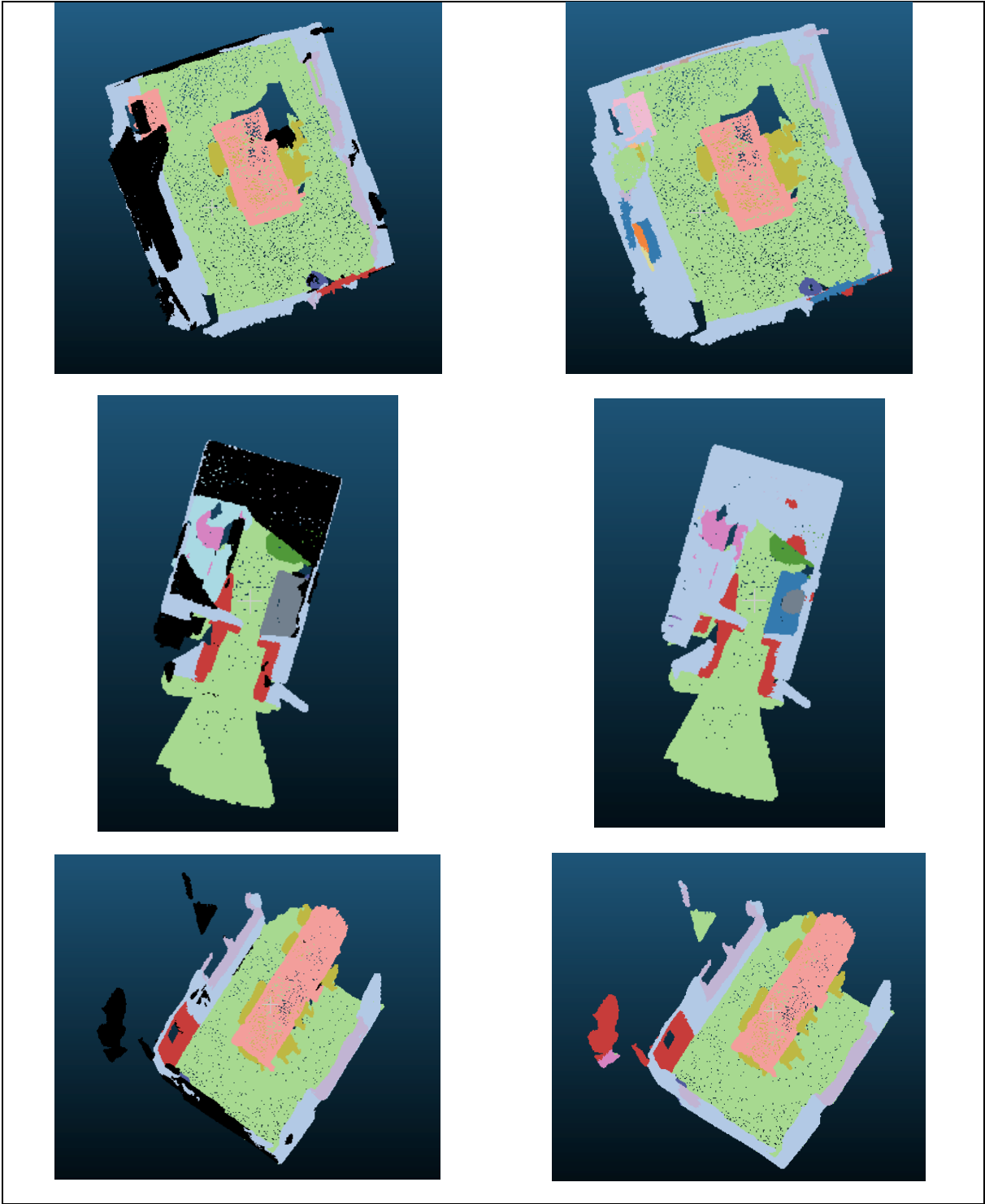


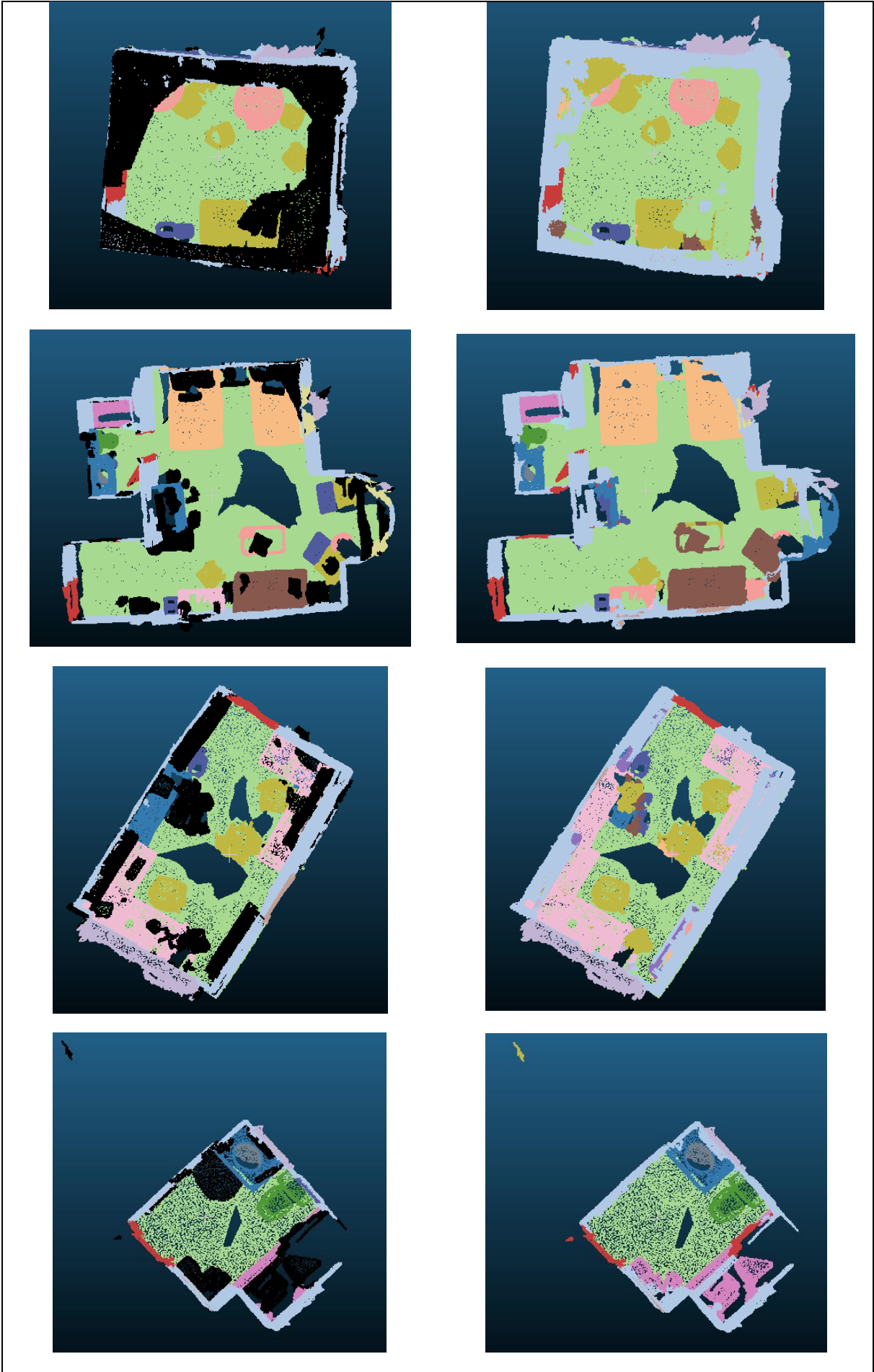


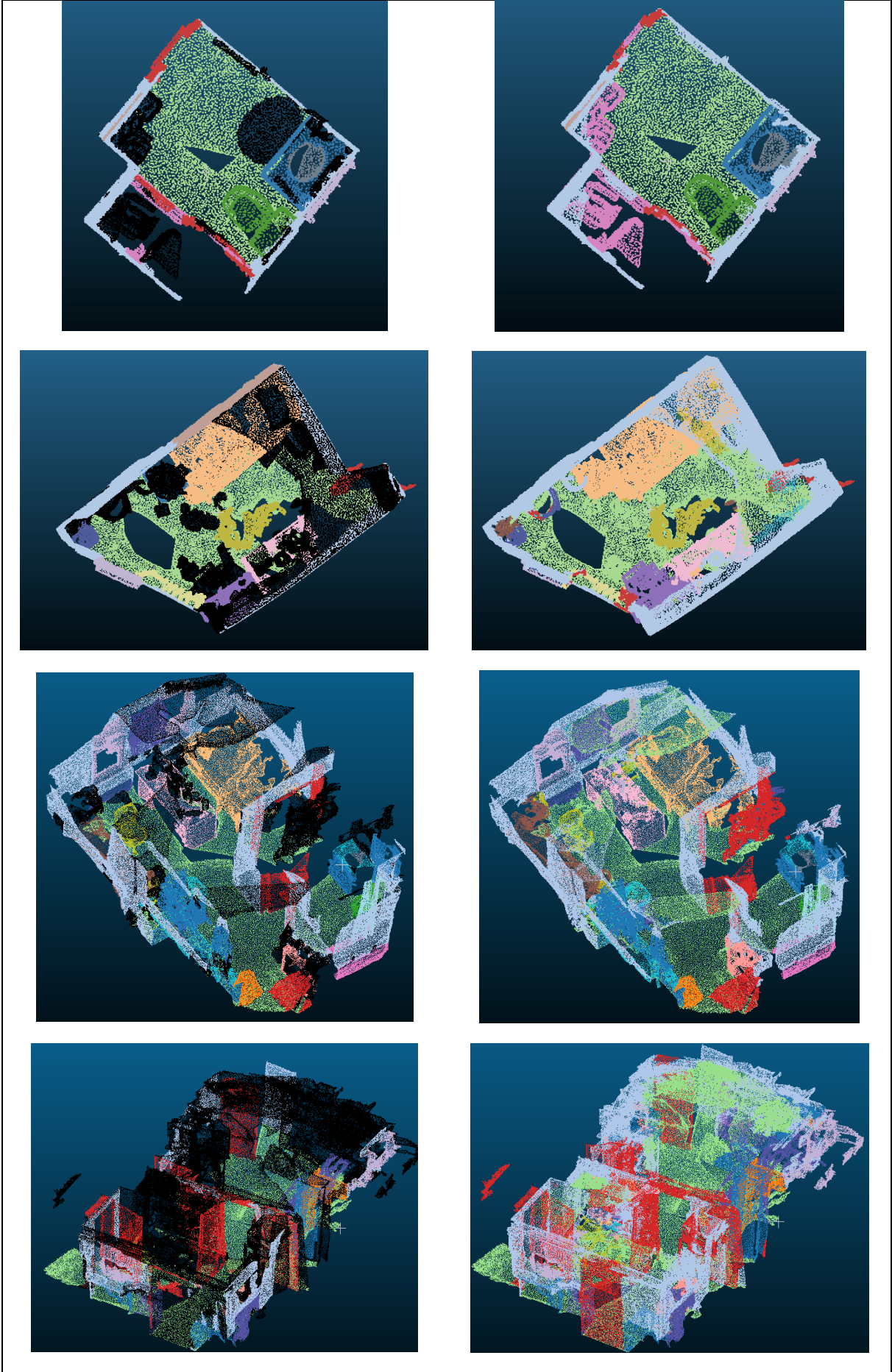


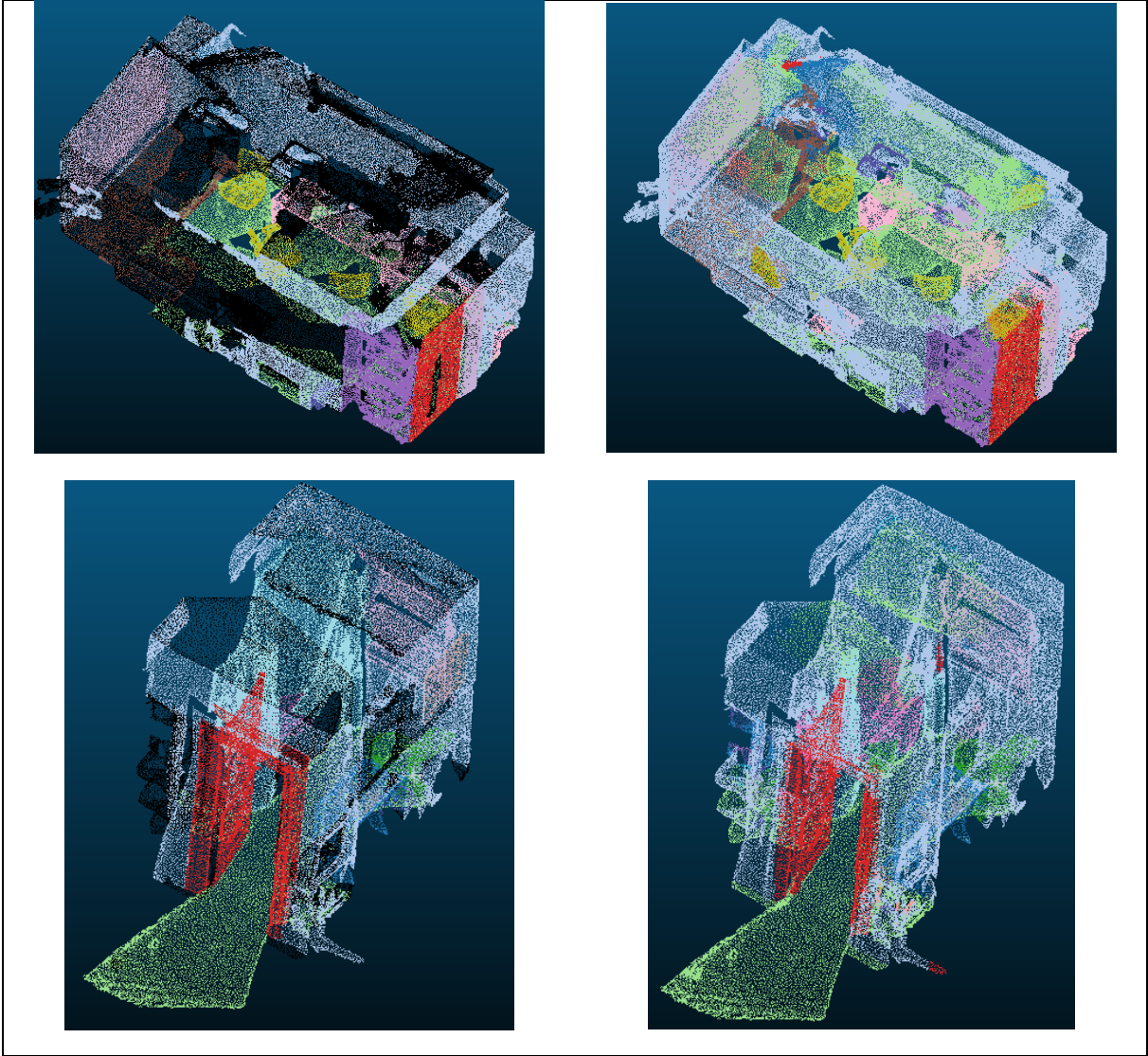












## Declaration

I hereby affirm that I have independently written the thesis submitted by me and have not used any sources or aids other than those indicated.

München, 30. August 2021

---

Changyu Du