Cross-Modal Data Fusion for 3D Instance Segmentation of Indoor Scenes

by

Edwin Gawing Ng

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Graduate Department of Aerospace Science and Engineering
University of Toronto

Cross-Modal Data Fusion for
3D Instance Segmentation of Indoor Scenes

Edwin Gawing Ng
Master of Applied Science

Graduate Department of Aerospace Science and Engineering
University of Toronto
2023

Abstract

Tremendous strides have been made in image-based scene understanding over the past decade, thanks to larger datasets and enhanced model capacity. However, 3D-based understanding still struggles, in part because 3D data are so costly to annotate. Top 3D instance segmentation models, trained exclusively on 3D data, outperform models using both 2D and 3D data, suggesting untapped potential in merging 2D data to enrich 3D pipelines. Interestingly, while instance segmentation has not yet benefited from 2D-3D data fusion, the sequential fusion of outputs from 2D and 3D models that are trained separately does improve object detection. This thesis applies sequential fusion to instance segmentation and investigates what and where to fuse. We demonstrate that current 2D models do not perform well enough compared to 3D models to enhance instance segmentation results, but that future, higher-performing 2D models should show performance gains using the sequential fusion method.

Acknowledgements

As I approach the end of my Master's program at the University of Toronto, I cannot help but feel the utmost gratitude to have been a part of this program, encompassing the friendships I've made and the valuable experience I've acquired along this journey. I would like to thank my advisor, Jonathan Kelly, for providing me with this opportunity. Throughout the program, your positive attitude and technical guidance helped me overcome obstacles in both my research and personal life, and I am truly grateful to have had you as my mentor. I wish to express my gratitude to my second reader, Steven Waslander, for conducting a comprehensive review and providing valuable insights through your comments and inquiries. I would also like to thank my friends and colleagues at the STARS laboratory and TRAILab for your support, advice, and constructive feedback. I would like to thank Andrej Janda for having been my partner on the ARIBIC project and for our conversations that helped culminate my initial research direction. I would like to particularly thank Anas Mahmoud for meeting with me regularly to discuss the progression of my research results and providing me with new perspectives to consider other approaches. I would like to thank Lee Slinger for her meticulous review of my thesis and valuable editorial recommendations. I would also like to thank those at aUToronto and the ARIBIC project for the experience of applying theory to a real-world product. Lastly, I would like to thank my parents, Victor Ng and Lily Ng, and sister, Olivia Ng, for your unconditional love throughout my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

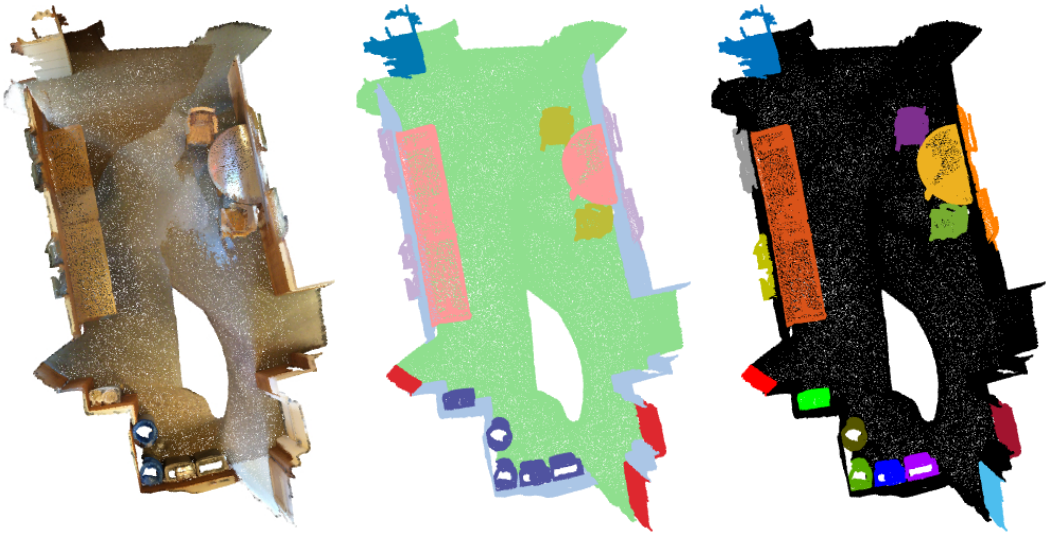
This thesis focuses on improving the 3D instance segmentation of indoor environments by combining 2D image data and 3D point cloud data. We begin by providing an overview of the segmentation process, current trends in the field of 3D instance segmentation, opportunities for improvement, and our contributions.

1.1 About Segmentation

Segmentation is a popular method for processing images or point clouds that facilitates interpretation by downstream decision-making modules. The goal of segmentation is to divide an image or point cloud into distinct regions, which can then be useful in a variety of applications. When used by autonomous vehicles, for example, segmentation can help identify pedestrians, other vehicles, and obstacles on the road. In warehouse environments, segmentation can assist with automatic inventory tracking by recognizing and identifying objects.

There are three primary types of segmentation: semantic, instance and panoptic. We focus on semantic and instance segmentation in this thesis. Figure 1.1 compares the application of semantic and instance segmentation to a point cloud that represents an indoor scene. Semantic segmentation provides a high-level understanding of the scene by identifying object categories, without differentiating between individual instances of those objects. Instance segmentation goes a step further by also distinguishing between different instances of the same class. Panoptic segmentation combines both of the above types of segmentation by distinguishing different instances of the same class, as well as identifying classes of amorphous background regions, such as the floor and walls.

Segmentation algorithms can also be categorized based on the type of data being



(a) Original Point Cloud (b) Semantic Segmentation (c) Instance Segmentation

Figure 1.1: Comparison between semantic segmentation and instance segmentation labels obtained from the ScanNet dataset [16].

segmented. Segmentation of images is referred to as 2D segmentation, while segmentation of point clouds is referred to as 3D segmentation. Notably, segmentation methods are not limited to using data exclusively from the modality of the data being segmented. For example, 2D techniques can incorporate point-cloud data to aid in image segmentation, which is known as fusion-based 2D segmentation. Similarly, 3D methods can integrate images to assist in point cloud segmentation, which is referred to as fusion-based 3D segmentation.

1.2 Motivation

The current landscape of leading 3D instance segmentation models is dominated by those that rely solely on point cloud data. Surprisingly, fusion-based models that leverage both image and point cloud data perform poorly in comparison. Point clouds, although providing accurate depth information, are relatively sparse, making segmentation inherently challenging. Images offer dense colour and texture information, and so intuitively, fusing dense image data with sparse point cloud data should aid in disambiguating the segmentation of objects. In particular, fusing multiple images that capture different angles of an object should further aid segmentation. Also, whereas 3D point-cloud datasets are typically limited in size, 2D image datasets tend to be

significantly larger. This substantial volume of labelled image data can be harnessed effectively for training purposes.

One possible explanation for why current fusion-based methods underperform 3D-only methods is that point clouds are less susceptible to degradation than images. The photometric information captured in images is sensitive to lighting conditions, such as intense shadows, reflections, and over- and underexposure. In general, extreme changes in lighting are not present in segmentation datasets, but even minor changes in lighting conditions can still cause relatively large variations in the data captured in an image. Consequently, fusion-based methods that incorporate image features without also incorporating confidence scores for the features may underperform compared to 3D-only methods [24]. Moreover, most fusion-based segmentation networks employ less mature architectures, and the alignment between the two modalities may be imprecise [47]. Previous work on cross-modal 3D instance segmentation jointly fused 2D and 3D data, which creates a complex optimization problem.

Other branches of 3D scene understanding have successfully enhanced 3D-only models by fusing 2D and 3D data sequentially in a process called sequential fusion [42, 44, 33, 47, 32]. A 2D model is trained separately on image data and then the image-based outputs of this model are fused into the 3D model. The advantage of sequential fusion over joint fusion is that the models can be trained separately, which simplifies the optimization problem and allows the method to be model-agnostic. This flexibility enables the pairing of the best-performing 2D segmentation model with the best-performing 3D segmentation model. Additionally, confidence scores from the 2D model can be easily incorporated within the sequential fusion approach.

Sequential fusion has been successfully applied to 3D object detection [42] and 3D semantic segmentation [47], however, to the best of our knowledge, it has not been successfully applied to improve 3D instance segmentation models. Furthermore, sequential fusion has primarily been used with outdoor datasets, where each point in the point cloud typically corresponds to only one or two image views. In contrast, indoor datasets often have many image views for each point cloud point. The availability of multiple views in indoor scenes presents further potential to fully exploit the benefits of 2D information. Consequently, this thesis aims to bridge the gap in the literature by exploring sequential, cross-modal data fusion for 3D instance segmentation, specifically tailored for indoor scenes.

1.3 Contributions

Our sequential fusion approach trains a 2D semantic segmentation model on images from an indoor dataset and extracts image-based features from this model. These image-based features are then fused into the 3D instance segmentation model during training. The key contributions of this thesis are summarized as follows:

- Previous work has focused on 3D-only instance segmentation or fusion-based 3D object detection for outdoor scenes. This thesis fills a gap in the literature by proposing a fusion-based instance segmentation method evaluated on indoor scenes.

- Whereas outdoor datasets tend to have only one or two image views corresponding to each point, indoor scenes tend to have a large number. We present an algorithm for establishing associations between points and corresponding pixels in situations involving multiple images. The goals of this research required considering a larger number of images than previous studies.

- We present experiments to determine where to fuse the image-based features by comparing the effect of early-fusion against mid-fusion.

- We present experiments to determine what to fuse, by comparing the effect of varying the content of the image-based fusion vector. The four different representations that we explore are features, logits, softmax scores, and one-hot encoded predictions extracted from the 2D semantic segmentation model.

- We provide an analysis of the effect that cross-modal data fusion has on semantic and instance segmentation performance.

- We provide an analysis to determine the main contributing factors that impact post-fusion performance.

- We provide a method of determining an estimated maximum performance boost using the mid-fusion approach.

- We perform an ablation study to determine the effectiveness of geometric information, compared to photometric information, on 3D instance segmentation performance.

Chapter 2

Background

In this chapter, we provide a review of the mathematical concepts needed to understand the thesis. We begin with an overview of images and point clouds, highlighting the strengths and weaknesses of each modality for scene understanding. Next, we discuss standard representations of 2D and 3D data and the types of data structures used in our work. We then examine three-dimensional and projective geometry in the context of point cloud segmentation. Lastly, we describe various scene-understanding tasks and provide an overview of available learning-based methods to solve such tasks.

2.1 Contrasting Images with Point Clouds

Images contain densely encoded photometric information, including the colour and intensity details of the environment. However, this photometric information is susceptible to variations in lighting conditions, such as low lighting, intense shadows, and bright reflections. Additionally, occlusions can obscure key elements—for instance, a pedestrian can be obscured by a moving vehicle. Images obtained from a monocular camera also lack depth information. Although stereo cameras (i.e., two cameras with known relative positions, designed to capture two images of the same subject) can be utilized for depth estimation, their range and accuracy remain limited.

Point clouds complement images by capturing geometric data from the environment, which often provide more precise depth measurements and which are less susceptible to environmental disturbances, such as changes in lighting conditions. Point clouds can also be converted to various other 3D representations, such as voxel grids and polygon meshes (as depicted in Figure 2.1), to suit different applications. If multiple point clouds captured from different positions in the environment are merged, they can form a single reconstructed cloud that encompasses object surfaces from

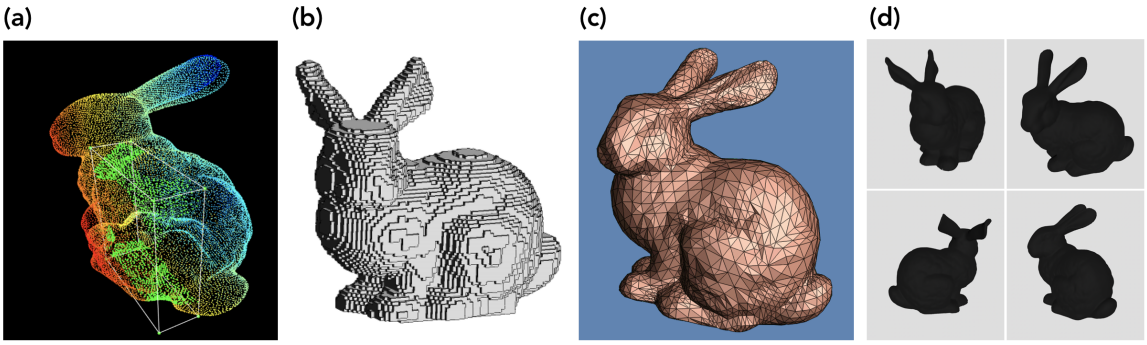


Figure 2.1: 3D data representations [17]: (a) a point cloud, (b) a voxel grid, (c) a polygon mesh, and (d) a multi-view representation.

various angles, thereby solving the occlusion problem. Point clouds, however, are typically sparse, whereas images consist of densely packed pixels. When using both data modalities simultaneously, images can help overcome potential issues with point clouds, such as sparsity.

Image datasets are orders of magnitude larger than point cloud datasets. Large 2D datasets, such as Conceptual Captions [40, 34], Segment Anything 1 Billion (SA-1B) [27], Conceptual 12M [8], and ImageNet [39] contain millions of images, whereas the largest 3D datasets—such as S3DIS [1], 2D-3D-S [2], ScanNet [16], and SemanticKITTI [4]—only contain thousands or tens of thousands of 3D scans. This difference in scale is also a result of the relative ease in labelling image data compared with point cloud data. Annotating point clouds typically requires human annotators to pan, zoom, and rotate the point cloud to label each point, whereas image data labelling requires fewer transformations. Contemporary point cloud datasets employ novel approaches to accelerate point cloud labelling. For instance, ScanNet [16] (discussed in more detail in Section 5.1.1) uses an over-segmentation algorithm, which separates the point cloud into small segments, to enable annotators to label each segment instead of each point. However, even with this novel labelling approach, it still takes on average 22 minutes to label one 3D scan of a 243 square foot room with 24 objects [16]. With the availability of such large image datasets, image encoders generalize well to novel data and continue to rapidly improve while 3D encoders are limited by data. Hence, there is untapped potential for improving 3D models by effectively fusing them with powerful image encoders.

2.2 Representations for 2D and 3D Data

Data can be represented in various ways and its format dictates the available processing methods. In this thesis, 2D images are represented using the RGB colour space, as captured by a standard camera with a rectangular photosensor. While other colour spaces, such as HSV (for hue, saturation, and value) exist, we opt to employ the RGB colour space due to its widespread use in learning-based frameworks. An image is represented as a 3D matrix, denoted as $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{H \times W \times C}$, where H represents the height of the image, W represents the width of the image and C represents the number of channels in the image. Conventionally, each image *pixel* location is defined by its coordinates: $u \in [0, \cdots, W-1]$ along the x-axis and $v \in [0, \cdots, H-1]$ along the y-axis. The origin is positioned at the upper left-hand corner of the image. RGB images contain three channels, a red, a green and a blue one; each channel measures the intensity of light for the corresponding colour. The values of each channel in RGB images are typically represented as integers ranging from 0 to 255 or floating point values ranging from 0 to 1, where larger values indicate a greater intensity. Greyscale images also utilize the same value ranges but contain a single channel.

In contrast, the raw 3D data obtained from depth sensors are typically in the form of an unordered set of point coordinates. The coordinates are obtained by measuring the distances of object surfaces from the depth sensor. Each point cloud can represent a single measured frame from the depth sensor or multiple frames can be concatenated to create a reconstructed map of the environment. This concatenation process is commonly used in indoor settings to generate a dense map of the surroundings (as discussed in Section 2.3). Point clouds are represented by a 2D matrix denoted as $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{N \times C}$, where N represents the number of points and C represents the number of channels. Note that the channels described here are different from the channels described for images. Point clouds contain three channels that represent the x, y and z coordinates of each 3D point. Colourized point clouds contain six channels consisting of an x, y, z and a red, green and blue channel. Additional information, such as the reflectivity of objects, can be encoded as separate channels.

Machine learning models such as PointNet [9] can process point clouds directly, but a more common approach is to transform the unordered point cloud into an ordered voxel grid using a process called *voxelization*. During the voxelization process, a resolution (voxel size) is chosen, which determines the side length of each voxel. Then, the RGB value of each voxel is computed as the average RGB value of all points that fall within the bounds of the voxel. If no points fall within a voxel, then

it is empty and its values are set to zero. Voxel grids are represented by a 4D tensor denoted as $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{H \times W \times D \times C}$, where H represents the height, W represents the width, D represents the depth, and C represents the number of channels. Voxel grids, which are obtained from the voxelization of colourized point clouds, contain three channels consisting of a red, green and blue channel.

2.3 Three-Dimensional Geometry

A key advantage of indoor point cloud datasets over those for outdoor autonomous vehicles is that they tend to include point cloud measurements from a larger variety of viewing angles, allowing dense reconstructed maps of the indoor environment to be produced. This section reviews the 3D geometry required to merge multiple point clouds measured from a moving depth sensor into a dense reconstructed map. We follow the notation of [3], where unbolded characters denote scalars, bolded lowercase characters denote vectors, and bolded uppercase characters denote matrices.

Objects in the environment are typically able to translate and rotate, giving them six degrees of freedom, three for translation and three for rotation. The position and orientation of an object define its *pose*. The position of a point on the surface of an object measured by a depth sensor can be described as a vector $\underrightarrow{r}^{ps}$. We follow the convention where the leftmost superscript describes the tip of the vector (point p) while the rightmost superscript describes the tail of the vector (sensor s).

Each set of points measured by the moving depth sensor must be represented in the same reference frame to generate the reconstructed map. By convention, the inertial (static or world) frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_i$, coinciding with the first pose of the depth sensor, is chosen as the reference. Therefore, the goal is to transform all point coordinates from the moving sensor frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_s$ to the inertial frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_i$. To obtain the relationship between the two reference frames, we start by expressing the vector $\underrightarrow{r}^{ps}$ in the sensor frame and in the inertial frame, which has the following relationship:

$$\underrightarrow{r}^{ps} = \underrightarrow{\boldsymbol{\mathcal{F}}}_i{}^T \mathbf{r}_i^{ps} = \underrightarrow{\boldsymbol{\mathcal{F}}}_s{}^T \mathbf{r}_s^{ps}. \tag{2.1}$$

The components of the vector $\underrightarrow{r}^{ps}$ with respect to the inertial reference frame are represented by $\mathbf{r}_i^{ps}$, while the components of the vector $\underrightarrow{r}^{ps}$ with respect to the sensor frame are represented by $\mathbf{r}_s^{ps}$. Following the convention in [3], the vector components are with respect to the reference frame denoted by its subscript. By rearranging the

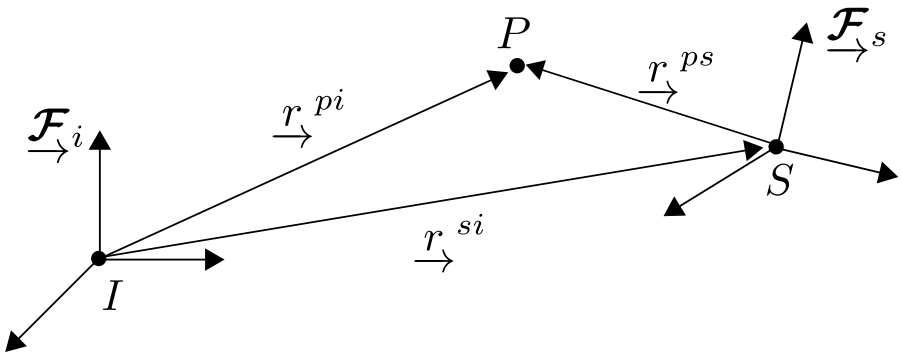


Figure 2.2: Visualization of a point P relative to an inertial reference frame $\underrightarrow{\mathcal{F}}_i$ and a moving sensor frame $\underrightarrow{\mathcal{F}}_s$.

equation, we obtain the relationship between the two components as

$$\mathbf{r}_i^{ps} = \underrightarrow{\mathcal{F}}_i \underrightarrow{\mathcal{F}}_s{}^T \mathbf{r}_s^{ps} = \mathbf{C}_{is}\mathbf{r}_s^{ps}. \tag{2.2}$$

The matrix $\mathbf{C}_{is}$ is called a rotation matrix; it converts the components in the sensor frame to the components in the inertial frame and is defined as

$$\mathbf{C}_{is} = \underrightarrow{\mathcal{F}}_i \underrightarrow{\mathcal{F}}_s{}^T. \tag{2.3}$$

Rotation matrices have the property that

$$\mathbf{C}_{is} = \mathbf{C}_{si}^T = \mathbf{C}_{si}^{-1}. \tag{2.4}$$

The set of all valid 3×3 rotation matrices in $\mathbb{R}^3$ form the *special orthogonal group*, defined as

$$\mathrm{SO}(3) = \left\{ \mathbf{C} \in \mathbb{R}^{3\times 3} \;\middle|\; \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det(\mathbf{C}) = 1 \right\}, \tag{2.5}$$

where $\mathbf{1}$ is the identity matrix and $\det(\mathbf{C}) = 1$ ensures that $\mathbf{C}$ is a *proper rotation* (i.e., a rotation only and not a rotation followed by a reflection).

In addition to rotating, the sensor can translate within the environment. The relationship between the position of the inertial frame and the sensor frame can be derived as:

$$\underrightarrow{r}^{pi} = \underrightarrow{r}^{si} + \underrightarrow{r}^{ps}. \tag{2.6}$$

Writing the relationship in the inertial frame $\underrightarrow{\mathcal{F}}_i$, we have

$$\mathbf{r}_i^{pi} = \mathbf{r}_i^{si} + \mathbf{r}_i^{ps}. \tag{2.7}$$

Substituting in Equation (2.2), we obtain

$$\mathbf{r}_i^{pi} = \mathbf{r}_i^{si} + \mathbf{C}_{is}\mathbf{r}_s^{ps}, \tag{2.8}$$

where $\{\mathbf{r}_i^{si}, \mathbf{C}_{is}\}$ represents the *pose* of the depth sensor. The relationship in Equation (2.8) can be rewritten in the form given by

$$\begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_{is} & \mathbf{r}_i^{si} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{T}_{is}} \begin{bmatrix} \mathbf{r}_s^{ps} \\ 1 \end{bmatrix}, \tag{2.9}$$

where $\mathbf{T}_{is}$ is referred to as a 4×4 *transformation matrix*. To use this transformation matrix, the 3×1 point coordinates must be augmented with a one. The augmented point coordinates are referred to as the *homogeneous* representation of the point. Using Equation (2.4), we can derive the property of the inverse of transformation matrices such that

$$\mathbf{T}_{is}^{-1} = \begin{bmatrix} \mathbf{C}_{is} & \mathbf{r}_i^{si} \\ \mathbf{0}^T & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{C}_{is}{}^T & -\mathbf{C}_{is}^T\mathbf{r}_i^{si} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{si} & -\mathbf{r}_s^{si} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{si} & \mathbf{r}_s^{is} \\ \mathbf{0}^T & 1 \end{bmatrix} = \mathbf{T}_{si}. \tag{2.10}$$

The set of all transformation matrices form the *special Euclidean group*,

$$\mathrm{SE}(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \;\middle|\; \mathbf{C} \in \mathrm{SO}(3), \mathbf{r} \in \mathbb{R}^3 \right\}. \tag{2.11}$$

Finally, the reconstructed map of the environment is obtained by transforming the vector components of all points measured with respect to the moving sensor frame into the inertial frame. This reconstruction process is performed by applying the corresponding transformation matrix to the points measured by the depth sensor in each sensor frame.

2.4 Projective Geometry

This section reviews the projective geometry necessary to project a 3D point into a 2D image. We consider, in particular, a depth camera, a type of sensor that captures both colour and depth information in the form of an RGB-D image. The captured colour and depth data are expressed with respect to the same sensor reference frame. We use projective geometry to determine pixel correspondences for each point, allowing

our fusion algorithm (discussed in Chapter 4) to append the associated 2D fusion information to each point.

The reconstructed map of the environment is represented as a point cloud $\boldsymbol{\mathcal{P}}$ with respect to the inertial reference frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_i$. To project each point into an image, we must first convert the point cloud coordinates from the inertial frame to the sensor frame. The transformation matrix $\mathbf{T}_{si}$ (equivalent to the inverse of the sensor pose shown in Equation (2.10)) is used to convert the coordinates from the inertial frame to the sensor frame. This transformation matrix is called the *extrinsic matrix* $\mathbf{E}$ as denoted by

$$\begin{bmatrix} \mathbf{r}_s^{ps} \\ 1 \end{bmatrix} = \mathbf{T}_{si} \begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix} = \mathbf{E} \begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix}. \tag{2.12}$$

Once the point coordinates are represented with respect to the sensor frame, a *projective mapping* based on the *ideal perspective model*, also known as the *pinhole camera model*, is applied to project the 3D coordinates onto a 2D image plane. The pinhole camera model describes the imaging characteristics when the aperture of the camera is reduced to a single point. In this scenario, all light rays converge at the *optical centre*, which serves as the origin of the sensor frame. The optical axis intersects with the *principal point*, which represents the origin of the image-plane coordinate system. The projective map from $\mathbb{R}^3 \to \mathbb{R}^2$ for a point $\mathbf{r}_s^{ps} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ in the sensor frame is given by

$$\mathbf{p} = \frac{\mathbf{r}_s^{ps}}{z} = \begin{bmatrix} x/z \\ y/z \\ z/z \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}, \tag{2.13}$$

where $\{x_n, y_n\}$ are called the normalized image coordinates.

The normalized image coordinates are associated with a hypothetical camera that has a unit *focal length* along both the horizontal and vertical axes, and has the origin set at the principal point. We can map the normalized image coordinates to the actual pixel coordinates through the relation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}}_{\mathbf{p}}, \tag{2.14}$$

where $\mathbf{K}$ is called the *intrinsic matrix*. The intrinsic matrix contains the focal length

along the horizontal axis f_u and the focal length along the vertical axis f_v, as well as the offset from the principal point (c_u, c_v). These parameters can be determined through camera calibration.

The complete projection model to convert a point with respect to the inertial frame to image coordinates is defined as

$$\mathbf{P}_1\mathbf{K}\frac{1}{z}\mathbf{P}_2\mathbf{E}\begin{bmatrix}\mathbf{r}_i^{pi}\\1\end{bmatrix} = \mathbf{P}_1\mathbf{K}\frac{1}{z}\mathbf{P}_2\begin{bmatrix}\mathbf{r}_s^{ps}\\1\end{bmatrix} = \mathbf{P}_1\mathbf{K}\frac{1}{z}\mathbf{r}_s^{ps} = \mathbf{P}_1\mathbf{K}\mathbf{p} = \mathbf{P}_1\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}u\\v\end{bmatrix}, \quad (2.15)$$

where

$$\begin{gathered}\mathbf{P}_1 = \begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix}f_u&0&c_u\\0&f_v&c_v\\0&0&1\end{bmatrix},\\ \mathbf{P}_2 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\end{bmatrix}, \quad \mathbf{E} = \mathbf{T}_{si} = \begin{bmatrix}\mathbf{C}_{si}&\mathbf{r}_s^{is}\\\mathbf{0}^T&1\end{bmatrix}, \quad \mathbf{r}_i^{pi} = \begin{bmatrix}x_i^{pi}\\y_i^{pi}\\z_i^{pi}\end{bmatrix}.\end{gathered} \quad (2.16)$$

The depth d measured by the depth sensor is computed as

$$d = zk, \quad (2.17)$$

where the constant k is used to convert between different units of measurement, for example from metres to millimetres.

2.5 Tasks and Metrics for Scene Understanding

Scene understanding encompasses various tasks involving both the 2D and 3D modalities. This section describes the tasks and metrics associated with scene understanding for the 2D modality, but the concepts discussed here can be extended to the 3D modality also. We will outline the key tasks of scene understanding in increasing levels of detail: image classification, object detection, semantic segmentation, and instance segmentation.

The simplest scene understanding task is *image classification*, where the goal is to categorize the entire image. Performance is generally measured in terms of accuracy. *Object detection* takes a step further by predicting both a class and bounding box around each object in the image. The metric used to measure the performance of an object detector is the mean average precision (mAP). To compute mAP, a *confusion*

matrix is computed for each class, which consists of four values:

- True Positive (TP): the number of correct detections made by the model.
- False Positive (FP): the number of incorrect detections made by the model.
- False Negative (FN): the number of times a ground-truth object is not detected by the model.
- True Negative (TN): this represents the background region that should not be detected by the model and is not detected by the model. However, this value is not used for object detection because these regions are not explicitly annotated in object detection datasets.

To determine whether a detection is considered correct or not, a metric called *intersection over union* (IoU) is used to measure how closely the predicted bounding box matches the ground-truth bounding box. Specifically, the IoU metric computes the overlap between the two bounding boxes as given by

$$\text{IoU} = \frac{\text{area}(\text{gt} \cap \text{pd})}{\text{area}(\text{gt} \cup \text{pd})}, \tag{2.18}$$

where gt is the ground-truth bounding box and pd is the predicted bounding box. IoU ranges from zero and one, where a value of zero means that there is no overlap between the bounding boxes and a value of one means that there is a full overlap. An IoU threshold of 0.5 (i.e., 50% overlap) is usually used to determine whether a correct detection is made by the model. Once the confusion matrix is computed for each class, the precision P and recall R metrics for each class are computed using

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.19}$$

and

$$R = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.20}$$

Precision is a metric that evaluates the accuracy of predicted bounding boxes for a specific class, while *recall* measures the ability of the model to detect all ground-truth bounding boxes for that class. The computation of these two metrics relies on an additional threshold called the confidence score threshold, which ranges from zero to one. Learning-based object detection models provide a confidence score for each predicted bounding box, and the confidence score threshold is used to remove predicted bounding boxes with low confidence scores. Increasing the confidence score

threshold results in fewer detected objects, leading to fewer false positives but more false negatives. Consequently, precision increases while recall decreases. In contrast, decreasing the confidence score threshold leads to more detected objects, resulting in more false positives but fewer false negatives. This causes precision to decrease while recall increases. Comparing models using two separate metrics can be cumbersome. To overcome this issue, precision and recall are combined into a *precision-recall curve*, as a function of the confidence score threshold. The *area under the curve* (AUC) is calculated to generate a metric called the *average precision*. The average precision summarizes the model performance on a specific class into a single metric. Finally, the *mean average precision* (mAP) is computed by taking the mean of the average precisions across all classes. In the literature, mAP may be followed by a number such as mAP_{25} or mAP_{50}, which indicates the mean average precision at different IoU thresholds, such as 25% or 50%, respectively.

Compared to classification and object detection, segmentation is a significantly more challenging task, because a prediction for each image pixel is required. Segmentation can be divided into two main categories: semantic segmentation and instance segmentation. Semantic segmentation involves classifying each pixel into an object class. It does not distinguish between different instances of objects in the same category. For example, given an image of a classroom, the task of semantic segmentation would be to label each pixel as a chair, desk, window, chalkboard, etc. All instances of the same category (e.g., all chairs) will be given the same label. The performance of semantic segmentation is a commonly evaluated using the mean intersection over union (mIoU) metric, which calculates the intersection over union (IoU) for each class and then computes the average. Instance segmentation takes a step further by not only classifying each pixel by its category but also distinguishing between individual instances of the same category. This is done by predicting a unique label called an instance ID for each individual object in the image. Similar to object detection, instance segmentation employs the mean average precision (mAP) as a metric. However, instead of computing the overlap between predicted and ground truth bounding boxes, the IoU metric calculated for the mAP metric computes the overlap between predicted and ground truth object segmentation masks. Specifically, in the case of ScanNet [16], 3D instance segmentation performance is measured using mAP_{25}, mAP_{50}, and mAP. These metrics represent the mean average precision using IoU thresholds of 25%, 50%, and an average over the range of 50% to 95% incrementing every 5%, respectively. In summary, the key difference between semantic and instance segmentation is in the granularity of the output.

2.6 Learning Algorithms

Although classical techniques—thresholding, K-means clustering [31], edge-based approaches, region-based techniques, the watershed transform, and others [5]—have proven effective for simple image segmentation tasks, they are often outperformed by learning-based methods in more complex domains. We begin this section with a brief introduction to machine learning fundamentals and then delve into the key components of learning algorithms that have state-of-the-art performance on 3D instance segmentation.

2.6.1 Machine Learning Fundamentals

There are two primary types of machine learning models: regression models, which are used to predict real-valued numbers from input data, and classification models, which are employed to predict categories based on input values. This subsection gives an introductory overview of fundamental regression and classification problems and explores their relationship with fully connected neural networks, which serve as the basis for many contemporary machine learning approaches. We also discuss the optimization and generalization issues of neural networks and potential solutions to these issues.

Primer on Regression

One of the simplest approaches to model the relationship between a scalar target and one or more input variables is linear regression. Formally, given a set of N examples $(\mathbf{x}^i, t^i)_{i=1}^N$ where the input values of example i are stored in vector $\mathbf{x}$ and the target value is stored in scalar $t \in \mathbb{R}$, the goal is to predict t^i from $\mathbf{x}^i$. In linear regression, as indicated by the name, a linear model predicts the target value from the input values and has the form

$$y = \mathbf{w}^T \mathbf{x}, \tag{2.21}$$

where y is the predicted value and column vector $\mathbf{w}$ contains the model parameters (also called model *weights*). From this model, an optimization problem is defined to solve for the weights that minimize the difference between the predicted value y from the target value t. Linear regression uses the L2 loss function also known as *squared error loss* defined as

$$\mathcal{L}_{L2}(y, t) = \frac{1}{2}(y - t)^2 \tag{2.22}$$

to optimize the weights. The loss is zero when the predicted and target values match and the loss error is quadratic. An alternative loss function, which is popular in regression-type problems is the L1 loss function, which is

$$\mathcal{L}_{L1}(y,t) = |(y-t)|. \tag{2.23}$$

The loss function is defined for a single example, whereas the corresponding *cost function* is defined as the average loss across the entire *training* set of examples. The relationship between the cost function $\mathcal{C}$ and loss function $\mathcal{L}$ is given by

$$\mathcal{C}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y^i, t^i). \tag{2.24}$$

The cost function is minimized with respect to the weights $\mathbf{w}$ to determine the optimal weights that minimize the difference between the predicted and target values.

There are two primary ways to minimize the cost function. The first method is to set the derivative of the cost function to zero,

$$\frac{\partial \mathcal{C}}{\partial \mathbf{w}} = 0, \tag{2.25}$$

and solve for the weights. The second method is to use an iterative algorithm called *gradient descent* to update the weight values in each iteration of the algorithm according to the iteration rule

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\frac{\partial \mathcal{C}}{\partial \mathbf{w}} \tag{2.26}$$

until convergence, where α is a *hyperparameter* called the *step size* or *learning rate*.

A hyperparameter is a parameter that is not learned from the data during the training of a machine-learning model but is set prior to the training process to control various aspects of the learning process. The learning rate is a hyperparameter that dictates how fast and stable the convergence characteristics are. A learning rate that is too small requires many iterations before convergence, while a learning rate that is too large would result in oscillations or instability during the optimization process.

The relationship between the input values and the target value is not always linear. A non-linear model such as

$$\begin{aligned} y &= w_3x^3 + w_2x^2 + w_1x + w_0 \\ &= \mathbf{w}^T\boldsymbol{\phi}, \quad \boldsymbol{\phi}(x) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \end{aligned} \tag{2.27}$$

can be used, where the choice of $\boldsymbol{\phi}$ is another example of a hyperparameter. $\boldsymbol{\phi}$ is also called a feature map [19], which involves selecting or designing a function that maps the original input values to a new set of features that better represents the relationship between the inputs and target value.

Careful consideration of hyperparameters is essential when developing a machine-learning model to avoid the issues of *underfitting* or *overfitting*. If the underlying relationship between the input data and the target output is quadratic, a linear model will underfit this relationship, whereas a cubic model has the potential to overfit. Hence, it is crucial to select suitable hyperparameters.

A separate dataset called the *validation set* is used to evaluate different hyperparameter choices. By testing various hyperparameters on the validation set, in a process called *hyperparameter tuning*, we can discover the values for the hyperparameters that yield optimal model performance. Furthermore, to obtain an unbiased evaluation of the performance of the model, we utilize a third dataset known as the *test set*. This dataset is independent from both the training set and the validation set. By using the test set, we can report the performance of the model without any biases introduced by hyperparameter tuning, ensuring a fair and accurate assessment.

Primer on Classification

The simplest classification problem is binary classification. In this problem, we are given a dataset consisting of N examples with input values and binary class targets denoted as $(\mathbf{x}^i, t^i)_{i=1}^N$, where t can take values of zero or one. The objective is to predict the target t^i based on the input values $\mathbf{x}^i$. For instance, we may want to determine whether an image depicts a dog or a cat. As mentioned in Section 2.2, an image $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{H \times W \times C}$ can be represented as a 2D grid of RGB values, which can then be flattened into a single input vector $\mathbf{x}$. In this case, we can assign a target value of zero to dogs and a target value of one to cats.

One of the core machine learning algorithms for solving binary classification problems is logistic regression. In logistic regression, we choose a log-linear model to represent the relationship between the input values and the target value. The model is formulated by the following two equations:

$$\begin{aligned} z &= \mathbf{w}^T \mathbf{x}, \\ y &= \sigma(z) = \frac{1}{1 + e^{-z}} \in [0, 1]. \end{aligned} \tag{2.28}$$

The *sigmoid function* $\sigma(z)$ maps a real-valued number to a range between zero and

one. The *binary cross-entropy loss* function is used for optimizing the weights and is defined as

$$\mathcal{L}_{BCE}(y, t) = -t \log y - (1 - t) \log(1 - y). \tag{2.29}$$

The loss is low when the predicted value is close to the target value and the loss is high when the predicted value is far from the target value. As with linear regression, the cost function for logistic regression is computed as the average of the loss across the training set, as shown in Equation (2.24). This cost function can be optimized by finding the direct solution using Equation (2.25) or by using gradient descent as shown in Equation (2.26).

Finding a direct solution may not be feasible in cases where the cost function is non-convex. Additionally, when dealing with large training sets, it may be computationally intractable to calculate the derivative over all examples required for gradient descent (a form of *batch training*). To address these challenges, stochastic gradient descent (SGD) is commonly employed. It is a form of *mini-batch training* where the derivative is computed for a subset of training examples during each iteration of the algorithm. The mini-batch size is another example of a hyperparameter. Choosing a mini-batch size that is too large can result in increased memory usage and longer computation time per iteration, while selecting a size that is too small may reduce the effectiveness of vectorized operations.

The process of determining the values for the weights in a model is known as the *training* process. Once the model has been trained, it can be used for *inference* to make predictions on new data. During inference, the predicted value y falls within the range of zero and one, indicating the probability of belonging to a particular category. To assign a specific category, a threshold is set, such that values equal or above the threshold are assigned a category of one and values below the threshold are assigned a category of zero. Typically, a default threshold of 0.5 is used, but it can be adjusted based on performance and specific requirements.

The binary classification task described earlier can be extended to handle multiple classes. For instance, given a flattened input vector representing an image, we may want to predict whether it belongs to the class of dog, cat, or pig. Formally, in a multi-class classification problem, we are given a dataset, which is comprised of N examples with input values and multi-class targets denoted as $(\mathbf{x}^i, \mathbf{t}^i)_{i=1}^N$. Here, $\mathbf{x} \in \mathbb{R}^{D \times 1}$ represents the input values with D dimensions, and $\mathbf{t} \in \mathbb{R}^{K \times 1}$ is a one-hot encoded vector representing the target, where K denotes the number of classes. The goal is to predict the target $\mathbf{t}^i$ based on the input values $\mathbf{x}^i$.

A *one-hot encoded vector* is a vector with zeros in all positions except for the index

corresponding to the labelled class, which is assigned a value of one. In our example, we can assign the one-hot encoded vectors $[1, 0, 0]^T$, $[0, 1, 0]^T$, and $[0, 0, 1]^T$ to represent the classes dog, cat, and pig, respectively. Alternatively, a simpler representation—known as *label encoding*—assigns the values zero, one, and two to represent dog, cat, and pig, respectively. However, label encoding may not work well in practice since the categorical classes of the animals do not form a continuous spectrum. For example, two cats (with a label encoding of one) do not form a pig (with a label encoding of two).

One approach to solving the multi-class classification task is through multi-class logistic regression. This method employs a model with the following two equations:

$$\begin{aligned} \mathbf{z} &= \mathbf{W}\mathbf{x} + \mathbf{b}, \\ y_k &= \text{softmax}(\mathbf{z})_k = \frac{\exp z_k}{\sum_{k'=1}^{K} \exp z_{k'}}. \end{aligned} \tag{2.30}$$

The parameters of the model, $\mathbf{W} \in \mathbb{R}^{K \times D}$ and $\mathbf{b} \in \mathbb{R}^{K \times 1}$, are called weights and biases, respectively. Together, the weights and biases form the parameters $\boldsymbol{\theta}$ of the model. The softmax function is used to map from the real-valued vector $\mathbf{z} \in \mathbb{R}^{K \times 1}$ called *logits* to a vector $\mathbf{y} \in \mathbb{R}^{K \times 1}$ called *softmax scores* where each element is in the range between zero and one. Each element y_k is the softmax score of the class index it corresponds to. The cross-entropy loss (multi-class version) is used in multiclass logistic regression and is defined as

$$\mathcal{L}_{CE}(\mathbf{y}, \mathbf{t}) = -\sum_{k=1}^{K} t_k \log(y_k) = -\mathbf{t}^T (\log \mathbf{y}). \tag{2.31}$$

The loss is high when the predicted softmax score of the correct class is near zero and the loss is low when the predicted score of the correct class is near one. The cost function can be the average loss over all training examples as shown in Equation (2.24), which is then optimized using gradient descent. However, for large datasets, it is more common for the cost function to be the average loss over a subset of training examples, which is optimized using stochastic gradient descent. During inference, the model in Equation (2.30) is used to determine the softmax scores. The element with the highest score is used as the predicted class as given by

$$\text{prediction\_index} = \underset{k}{\text{argmax}} \left(\text{index}(\mathbf{z}, k) \right), \tag{2.32}$$

where the $\text{index}(\mathbf{v}, i)$ function outputs the value of the vector at index i.

2.6.2 Neural Networks and Deep Learning

Artificial neural networks have become extremely successful in a variety of regression and classification tasks. Equation (2.30) is an example of a model with a single *fully connected* layer. A fully connected layer is a function defined as

$$\mathbf{h}^l(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{W}^l\mathbf{x} + \mathbf{b}^l) \tag{2.33}$$

with layer index l where $\boldsymbol{\phi}(\cdot)$ is called the activation function and each input or output element of the fully connected layer is called a *neuron*. The input neurons are the elements x_d in vector $\mathbf{x} \in \mathbb{R}^{D\times 1}$ and the output neurons are the elements h_k^l from the vector $\mathbf{h}^l \in \mathbb{R}^{K\times 1}$. The equation for an output neuron is

$$h_k^l = \phi({\mathbf{w}_k^l}^T\mathbf{x} + b_k^l), \tag{2.34}$$

where ${\mathbf{w}_k^l}^T \in \mathbb{R}^{1\times D}$ is the k-th row of matrix $\mathbf{W}^l \in \mathbb{R}^{K\times D}$. A fully connected neural network defined by function $\mathrm{FC}(\mathbf{x})$ is the composition of M fully connected layers as given by

$$\mathbf{FC}(\mathbf{x}) = (\mathbf{h}^M \circ \cdots \circ \mathbf{h}^1)(\mathbf{x}). \tag{2.35}$$

The purpose of the activation function is to add non-linearity to the otherwise linear model. By having a non-linearity in the activation function and an exponentially large network, a fully connected network can universally approximate any function. Many different types of non-linear activation functions can be used. The following activation functions will be defined for a scalar value. A linear function

$$\phi(x) = x \tag{2.36}$$

represents no activation function, such as in Equation (2.30). The most common activation function for fully connected networks is the rectified linear unit (ReLU) defined by

$$\phi(x) = \max(0, x). \tag{2.37}$$

Backpropagation popularized by [38] is the most common algorithm for computing the gradients necessary for updating the parameters of a neural network by using the chain rule to compute derivatives efficiently. However, for very deep neural networks, using the ReLU activation function can cause vanishing gradient problem. This problem arises when the gradients of the loss function with respect to the parameters of the network become very small as they are back-propagated through the layers of the

network during the training process. When gradients become extremely small, they effectively "vanish", making it difficult for the network to update the weights in a meaningful way. This can significantly slow down or hinder the convergence of the training process. One method to reduce this issue is to use a soft ReLU activation function, which produces small values rather than a zero value for negative input values. The soft ReLU is defined as

$$\phi(x) = \log(1 + \exp x). \tag{2.38}$$

Optimization Issues and Potential Solutions

As alluded to above, the primary challenges associated with neural networks are around optimization. The cost function is generally non-convex, so stochastic gradient descent may have problems getting stuck in local minima, encountering saddle points, or reaching plateau areas. Any of these optimization problems can lead to difficulties in converging to the global minimum. However, there are a few techniques to reduce these problems, such as initializing the parameters of the network with small random values instead of initializing all weights to zero and using *momentum* in the stochastic gradient descent algorithm to reduce the potential of getting stuck in local minima. The update rule for stochastic gradient descent with momentum is defined by the following two iteration rules:

$$\begin{aligned} \mathbf{p} &\leftarrow \mu\mathbf{p} - \alpha\frac{\partial C}{\partial \boldsymbol{\theta}}, \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \mathbf{p}. \end{aligned} \tag{2.39}$$

The momentum factor μ is usually set to a value slightly less than one, such as 0.99 or 0.9. In stochastic gradient descent, when the algorithm reaches a local minimum, the magnitude of the gradient will reduce and the algorithm may converge to the local minimum instead of the global minimum. The addition of the momentum factor helps to overshoot small local minima and increases the chance of converging to the global minimum. Another extension of the stochastic gradient descent algorithm that uses momentum is the Adam optimizer [26]. It adjusts the learning rate adaptively for each model parameter based on the exponential moving average of the gradients and the square of the gradients calculated for that parameter; this leads to faster convergence. Other techniques to reduce the chance of getting stuck in local minima, such as ravines, involve centring inputs to zero mean and unit variance, as done with *batch normalization* or performing learning rate decay. Learning rate decay refers to the method of reducing the learning rate over the course of training. This is useful

because the model parameters are far from optimal in the early stages of training so a larger learning rate is necessary. In contrast, the model parameters would be closer to a minimum in the later stages of training, so a small learning rate to fine-tune the parameters is desired. Other learning rate strategies, also referred to as learning rate schedules, exist. For instance, a cosine annealing learning rate schedule decreases the initial learning rate to a value near zero before increasing the learning rate back to its original value in a cyclical process. By introducing periodic fluctuations in the learning rate, the model can potentially escape local minima in the loss landscape.

Generalization Issues and Potential Solutions

In addition to optimization issues, the performance on the training set does not always generalize to the test set. To improve generalization from the training set to the test set, a variety of techniques can be used, such as data augmentation, dropout, capacity reduction, weight decay and early stopping. Data augmentation is the process of adding more data to the training set by transforming the existing training examples. The transformations used depend on the task, but the common transformations for images are: translation, horizontal/vertical flip, rotation, smoothing and adding noise. For point clouds, common transformations include: randomly dropping out points, random rotation and axis flip. Another technique called *dropout*, randomly 'turns off' a number of neurons, which prevents overfitting by encouraging the network to learn redundant representations of the data. This redundancy makes the network more robust because it can not rely too heavily on the output of a single neuron. Dropout can also be thought of as training an ensemble of multiple models within a single model. Each dropout configuration (i.e., which neurons are dropped out) corresponds to a different subnetwork. When making predictions, dropout is typically turned off, but during training, different subnetworks are sampled. This ensemble effect helps improve the ability of the model to generalize. Reducing the *model capacity*, which is the ability of the model to fit a variety of functions, is yet another method to reduce overfitting. There are many ways of reducing model capacity, such as introducing a smaller layer between two larger layers called a *bottleneck layer* or simply reducing the number of layers or parameters of the model. *Weight decay* is a method of regularizing the neural network by penalizing large weight values, which encourages the network to only use small weight values. Small weight values are desirable because they are less susceptible to changes in the data distribution between training and test set.

Weight decay augments the cost function into the form of

$$C_{\text{reg}} = C + \lambda R, \tag{2.40}$$

where λ is the regularization rate that describes how heavily to penalize large weight values and R is the regularize. The two most common regularizers are L2 regularization in the form of

$$R_{L2} = \sum_i w_i^2 \tag{2.41}$$

and L1 regularization in the form of

$$R_{L1} = \sum_i |w_i|, \tag{2.42}$$

where w_i is an individual weight value. L2 regularization encourages weights to be close to zero while L1 regularization encourages weights to be zero, which is a form of feature selection. Lastly, early stopping of the optimization process is a method to reduce the model capacity, which in turn reduces the chance of overfitting onto the training set.

2.6.3 Convolutional Neural Networks

A convolutional neural network (CNN) named AlexNet [28] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [39] in 2012, which popularized the method of applying convolutional neural networks to image-based tasks. ImageNet [39] is a dataset of over 14 million annotated images with more than 20,000 categories for the image classification task; ILSVRC trims the list of categories to a non-overlapping set of 1,000.

A convolutional neural network uses convolutional layers. Unlike fully connected layers that require many parameters, convolutional layers share parameters within a layer making them much more efficient. A convolutional layer uses a square matrix called a convolutional *kernel* or *filter*, to extract features from square patches of the image by performing an element-wise multiplication of the kernel with the extracted patch and summing the results. The *receptive field* of the kernel is defined by its size, and the output of a convolutional layer is called a set of *features*. Convolutional layers use the kernel to process consecutive (possibly overlapping) patches of the image in a "sliding window" fashion to extract features such as edges. The number of pixels to skip between consecutive patches when performing the windowing is called the

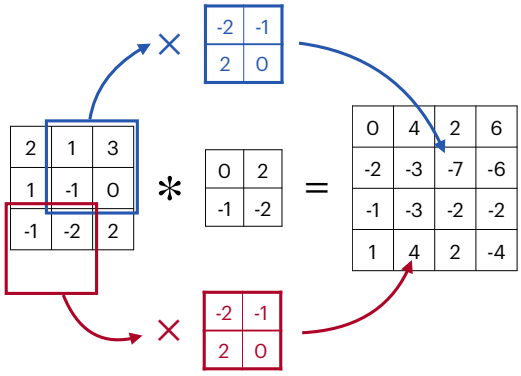


Figure 2.3: Illustration of the convolution operation. The illustration depicts a 3×3 input matrix being convolved with a 2×2 convolutional kernel to generate a 4×4 output matrix. The kernel is applied to each extracted patch of the input matrix in a "sliding window" fashion. For each extracted patch, the kernel performs a "flip-and-filter" operation. The flip operation refers to a double diagonal reflection of the square matrix, which involves reflecting the matrix first over its main diagonal (from the top-left to the bottom-right corner) and then over its secondary diagonal (from the top-right to the bottom-left corner). The filter operation refers to an element-wise multiplication of the flipped kernel with the extracted patch from the input matrix and summing the results. Note that this is the mathematical definition of the convolution operation. The convolution operation in convolutional layers of CNNs only performs the filtering step.

stride of the kernel. Convolutional layers found in the earlier parts of the network extract fine-grained features, while convolutional layers found in the later parts of the network extract higher-level features (by aggregating lower-level features).

The element-wise multiplication and summation of the kernel with the extracted patch of the image is referred to as the convolution operation. This operation, used in mathematics, is shown in Figure 2.3, where there is an additional step of flipping the kernel before performing the element-wise multiplication and summation.

However, it is important to note that convolutional layers in neural networks do not perform convolutions in the strict mathematical sense. Instead, they only perform the element-wise multiplication and summation step [19].

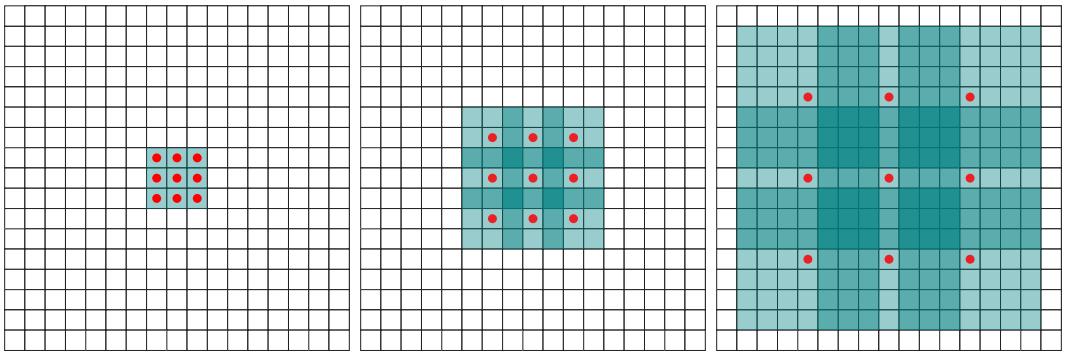
Convolutions are commutative and associative, so when applying two convolution operations on an image, the order of the two operations does not matter. They are also *equivariant* to translation, meaning that if the input is translated, then the output of the convolution layer is shifted by the same amount. This is a useful property because CNNs are used to recognize patterns in data and translational equivariance implies that if the input pattern shifts, the response or prediction of the network also shifts by the same amount.

A desirable property of the convolutional neural network would be its ability to remain unchanged or consistent when exposed to minor image transformations. To obtain this property, a pooling layer is used to summarize the features extracted from convolutions. The most common pooling layer summarizes the features by aggregating the features and extracting the maximum value within the extracted patch called *max pooling*.

The authors of AlexNet emphasized that the depth of the model was essential in its performance. A higher-performing model called VGG [41] was introduced in 2014 with 19 layers, compared to the 8 layers used in AlexNet. However, deeper neural networks were more difficult to train due to the vanishing gradient problem. To address this problem, a CNN called ResNet [20] introduced the use of *residual connections*, also known as *skip connections*, that allow previous layer outputs to be propagated to later parts of the model. Directly propagating the outputs from previous layers to subsequent layers reduces the chance of encountering zero gradients, which allowed deeper neural networks to be trained. Residual connections have become a common part of contemporary segmentation models.

Types of Convolutions

In current state-of-the-art computer vision methods, numerous improved variants of the convolution operation are utilized. This section highlights the most relevant variants employed in the domain of segmentation.



(a) Dilation Rate = 1 (b) Dilation Rate = 2 (c) Dilation Rate = 4

Figure 2.4: Convolutions with different dilation rates [45]. The number of parameters is identical, but the receptive field grows exponentially as the number of parameters increase linearly. (a) A dilation rate of one is equivalent to a normal convolution. (b)(c) Display dilated convolutions with rates of two and four, respectively.

Dilated convolutions, also known as *atrous convolutions*, introduce an additional

parameter called the *dilation rate* shown in Figure 2.4, which determines the spacing between values in the kernel. By adjusting the dilation rate, the receptive field can be increased while using fewer parameters, thereby improving the efficiency of the network.

Regular and dilated convolutions both generate output maps smaller than the input. In some cases, generating an output map larger than the input is necessary. *Transposed convolutions* enable the creation of output maps larger than the input by inserting additional rows and columns of zeros between each pixel of the input layer, padding the image border with zero values, and then performing a regular convolution with a stride of one over the modified input.

Another category of convolution is the separable convolution, of which there are two main types. *Spatially separable convolutions*, also known as *asymmetric convolutions*, decompose a 2D convolution into two 1D convolutions. In the first step, a 1D convolution is performed on the x-axis of the input. The second step performs a 1D convolution on the y-axis of the input. The second type, *depthwise separable convolutions*, are also performed in two steps. In the first step, a 2D convolution is performed for each channel of the input. In the second step, a 1x1 convolution is applied to the output of the previous operation. Separable convolutions significantly reduce the number of multiplication operations, resulting in accelerated computation time.

In extending convolutions from two-dimensional data to three-dimensional data, point clouds (discussed in Section 2.2) are voxelized, as convolutions can only be applied to ordered data. This allows us to apply *3D convolutions* directly on the voxel grid, which—when obtained from point clouds—also typically contain numerous empty voxels. To leverage this sparsity, *sparse convolutions* are used, which allow empty voxels to be discarded during computations to improve efficiency.

2.7 Summary

We began this chapter by exploring the merits and drawbacks of images and point cloud data. Images capture dense photometric information, but can be degraded by unfavourable lighting conditions; they also do not necessarily represent occluded portions of a scene. In contrast, point clouds contain depth and geometric data and exhibit greater resilience to lighting and occlusion issues. However, they are sparser and prove more challenging to annotate.

Subsequently, we laid the groundwork for our fusion-based research by explaining

various representations of image and point cloud data, alongside providing an introduction to the essentials of 3D geometry. The section on projective geometry provided details on the mathematics of the mapping between 3D points and 2D pixels.

Following the mathematical foundations, we shifted our focus to the problem of scene understanding and described associated task metrics, with an emphasis on semantic segmentation and instance segmentation. Finally, we provided a comprehensive review of the fundamentals of machine learning, that serves as a prerequisite for understanding how recent related work has implemented cross-modal data fusion in 3D scene understanding tasks.

Chapter 3

Related Work

This thesis explores the potential of fusing 2D semantic segmentation with 3D instance segmentation models. In this chapter, we provide an overview of prior work on 2D-semantic and 3D-instance segmentation, including their development, foundational models and those used in the current study. We also review previous efforts to combine models for each modality in the context of object detection, semantic segmentation, and instance segmentation.

3.1 Semantic Segmentation in 2D

Early methods for semantic segmentation performed image classification on consecutive sub-regions of the image using a "sliding window:" a small patch or window of pixels was extracted from the image and then processed by a fully connected layer to classify the centre pixel of the patch. The sliding window was then shifted and another fully connected layer was used to classify the centre pixel of the newly extracted patch. This process was repeated for all pixels in the image. This approach, however, was very computationally inefficient as each fully connected layer contained its own set of parameters that needed to be learned. Additionally, fully connected layers could only accept fixed-size inputs, so all images in the dataset were required to be of the same size.

The fully convolutional network (FCN) [30] solves both the inefficiency problem and the fixed-sized input problem by replacing the fully connected layers with convolutional layers. Convolutional layers do not restrict the input size and they use shared parameters across all image patches, improving computational efficiency. The performance of the FCN, however, is limited by the type of upsampling layer used in the network. An upsampling layer is one that produces an output with a size

greater than its input; a downsampling layer is one that produces an output that has a smaller size than its input. The convolutional layer is a downsampling layer: the FCN architecture uses multiple convolutional layers to extract useful image features, downsampling the size of the input in the process. To produce a semantic prediction for each pixel, the internal network representation must be upsampled back to the same size as the input image, which the FCN architecture does through bilinear interpolation in the upsampling layer to generate a semantic prediction map. Notably, unlike convolutional layers, bilinear interpolation layers have no tunable parameters, so the FCN architecture is unable to adjust any of the class predictions made by the final layers of the network.

One of the foundational components of most modern semantic segmentation models is the UNet [37] architecture, a network based on the FCN design. It replaces bilinear interpolation with transposed convolutions (described in Section 2.6.3) in the upsampling step. This enables the model to adjust class predictions in the final stage of the network. Multiple transposed convolutions are applied to increase the number of tunable parameters in the upsampling stage, which boosts performance.

The ENet [35] model is based on the UNet architecture and is one of the two semantic segmentation models selected for further investigation in this thesis. Its main advantage is the emphasis on efficiency, which leads to fast training and high inference speed. It does this by replacing regular convolutions with dilated convolutions (described in Section 2.6.3) and by downsampling the input image more heavily, using a smaller decoder, and performing pooling in parallel with the convolutional layers.

Other important models used for semantic segmentation include ResUNet [46] and the DeepLab [13, 10, 12, 11] family of models. ResUNet combines the strengths of ResNet [20] and UNet [37] by introducing residual connections (discussed in Section 2.6.3) within the downsampling layers of the network and between the downsampling and upsampling layers, enabling even deeper networks to be trained. The DeepLab models are based on FCN and use dilated convolutions and depth-wise separable convolutions (discussed in Section 2.6.3) to improve speed and accuracy.

Pri3D [23], which can use the ResUNet or DeepLabv3+ architecture as a feature extractor, is the second semantic segmentation model chosen for investigation in this thesis, primarily because it achieves state-of-the-art performance on the ScanNet [16] dataset. It also has slightly higher performance and unlike the previously discussed models, which only use 2D image data, it includes 3D priors in a pre-training stage before fine-tuning on image data. In the pre-training stage, Pri3D trains the network

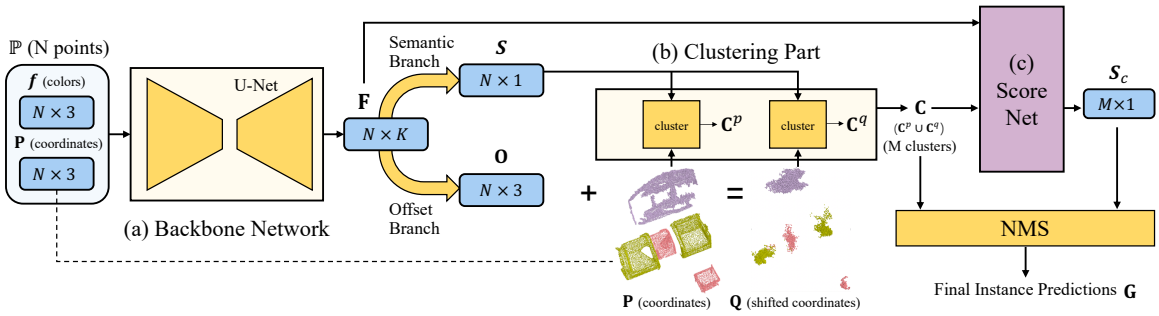
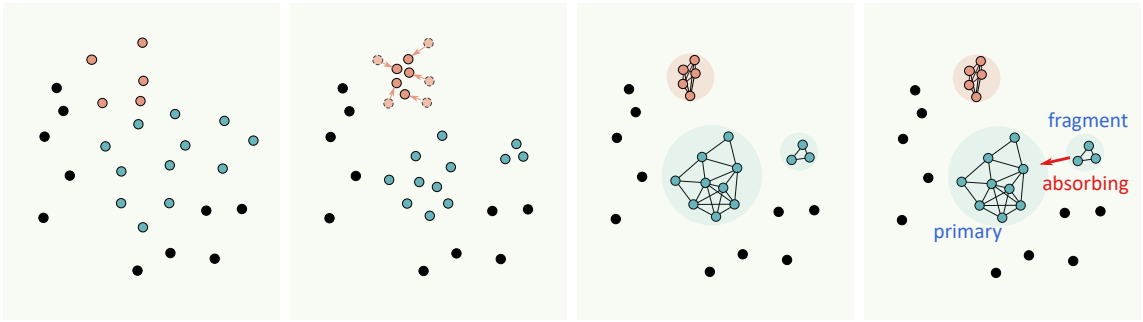


Figure 3.1: Illustration of the PointGroup architecture [25]. The backbone network extracts point features and predicts a semantic class and offset vector for each point. The offset vectors are used to shift each point to its respective instance centroid. The original point coordinates and shifted point coordinates are each separately grouped into clusters. Each cluster is assigned a score measuring how confident the network is that the cluster forms an object instance. Finally, low confidence clusters are removed and the remaining clusters are output as the set of final instance predictions.

to output similar features for all pixels and voxels that correspond to the same point in 3D space. Correspondences between pixels and voxels are computed using perspective geometry, a topic described in Section 2.4. After pre-training, the network is fine-tuned on images only to perform semantic segmentation.

3.2 Instance Segmentation in 3D

Early 3D instance segmentation models were based on related object detectors and operated by predicting bounding boxes and then predicting point-level masks for instance segmentation. Modern approaches instead predict point-level instances directly through a 'point grouping' technique that leverages the void space between objects to improve performance. PointGroup [25], whose architecture is shown in Figure 3.1, originally introduced this grouping procedure. The architecture incorporates three main processing stages. The first stage uses a two-branch network based on the 3D UNet [15] to extract point features and to predict an object class and offset vector for each point. An offset vector is a three-dimensional vector with x, y and z components. When the offset vector is added to the point coordinates, the point is shifted towards its respective instance centroid. In the second stage, candidate object instances are predicted by clustering the original point cloud and the offset-shifted point cloud. The reason for applying clustering to both point clouds is that clustering on the original cloud (only) tends to produce incorrect predictions: adjacent same-class instances are often identified as a single instance. For example, if two cabinets



(a) Original coordinates (b) Shifted coordinates (c) Point aggregation (d) Set aggregation

Figure 3.2: Illustration of hierarchical aggregation [14], used to group points into predicted object instances. (a) Original points in the point cloud. (b) Points are shifted towards their predicted instance centroid. (c) Points in close proximity to each other are joined into a single set. (d) Small sets of points are merged into larger sets that have the same semantic class. Points with different colours belong to different semantic classes. Black points belong to the background.

are next to each other, clustering on the original point cloud will often group both cabinets together as a single instance. Clustering the offset-shifted point cloud improves the chances of grouping each object separately because the offsets increase the void space between adjacent objects (as shown in Figure 3.1). Conversely, the authors of PointGroup choose to not only cluster on the offset-shifted point cloud because of potential inaccuracies in the offset vectors. Any errors in the offset vectors may shift points towards an incorrect instance centroid; points near object boundaries, which are furthest from the instance centroid, are more susceptible to being shifted to an incorrect instance. In the last stage, a UNet-based network called ScoreNet is used to predict a confidence score for each candidate instance found in the previous stage. Finally, the network determines the final instance predictions by removing duplicate candidates that have low confidence scores.

HAIS [14] is a newer model that improves upon the PointGroup architecture by modifying the clustering step. HAIS has two main stages. Like PointGroup, first HAIS predicts an object class and offset vector for each point. In the second stage, the authors introduce "hierarchical aggregation", a novel method to cluster points into final instance predictions. First, as shown in Figure 3.2(b), the points (referred to as nodes by the authors) are shifted towards their predicted object centroids using the offset vectors from the first stage. Distances between every pair of nodes are computed and, as shown in Figure 3.2(c), an edge is formed between two nodes when the distance between them is less than a threshold value. After performing this operation on all

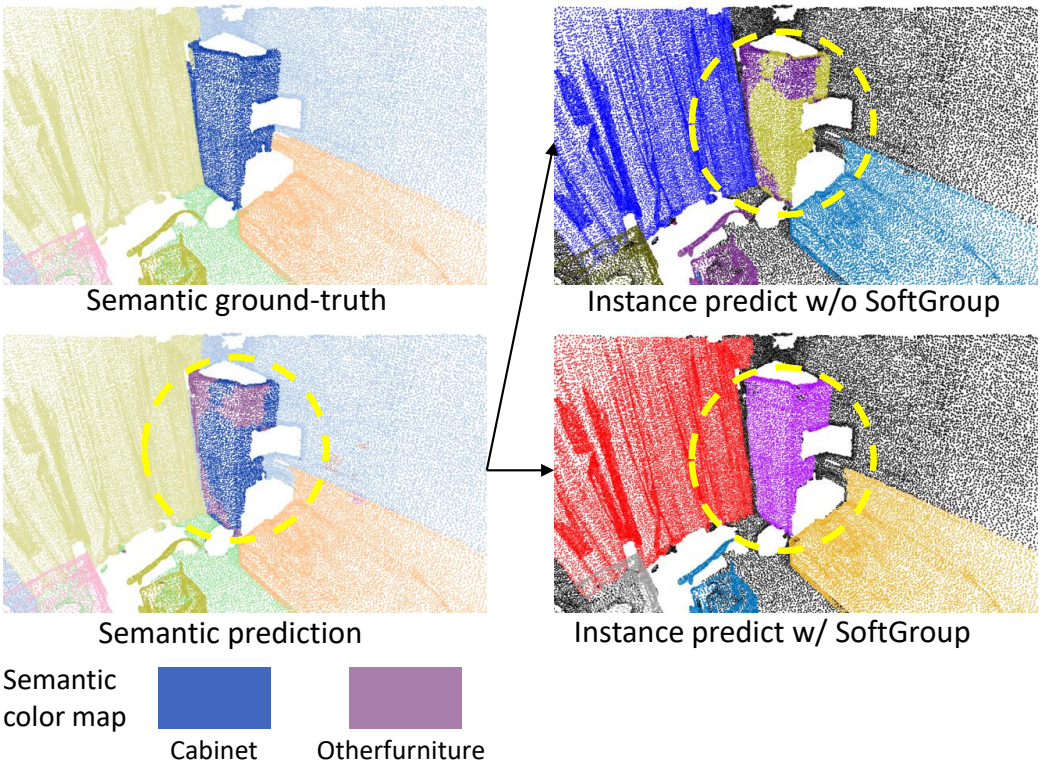


Figure 3.3: Illustration of how soft grouping [43] of points (bottom right image) can improve instance predictions compared to hard grouping (top right image). Hard grouping refers to the clustering of points into instances such that each point is only associated with a *single* semantic class. Soft grouping refers to clustering points into instances such that each point is associated with *multiple* possible semantic classes.

pairs of nodes, the point cloud is separated into independent groups of connected nodes. These groups can be interpreted as intermediate instance predictions. In the final step of hierarchical aggregation, smaller groups of points are merged into the closest large group with the same semantic class prediction, forming the final instance predictions. This hierarchical approach produces more accurate segments and non-overlapping instances. Hence, unlike the clustering approach of PointGroup, HAIS does not require a third stage to remove duplicate candidate instances.

PointGroup and HAIS make 'hard' semantic segmentation predictions that force each point to be associated with a single class after the first stage of processing. However, any errors in the hard semantic predictions propagate to the clustering stage and may lead to false positives. SoftGroup [43], whose architecture is shown in Figure 3.4, improves performance by introducing a 'soft' grouping module that associates each point with multiple high-confidence classes instead of a single class. We choose SoftGroup as the 3D instance segmentation model to investigate in this thesis because

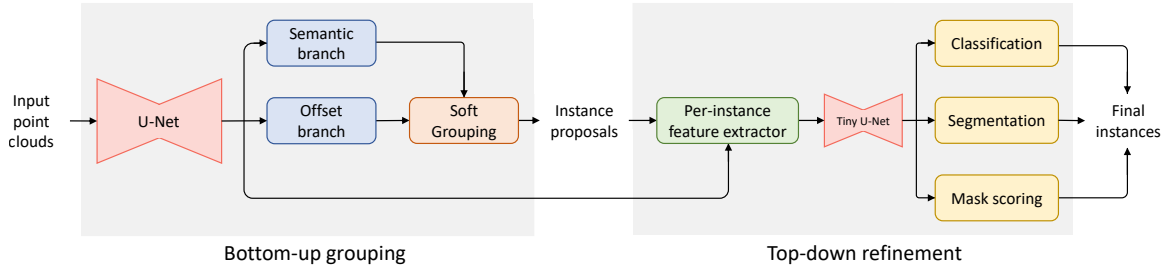


Figure 3.4: Illustration of the SoftGroup architecture [43]. A UNet extracts point features and predicts semantic confidence scores and an offset vector for each point. Then, for each semantic class, the soft grouping module generates candidate instances by performing hierarchical aggregation on the subsets of points with confidence scores higher than a threshold. Each candidate instance is a group of points predicted to form an instance. Lastly, a tiny UNet is used to predict the semantic class, segmentation mask, and mask confidence score for each candidate instance, forming the final set of output instances. The segmentation mask is used to differentiate between foreground (part of the instance) and background points in each candidate instance.

it achieves state-of-the-art performance on the ScanNet dataset [16]. SoftGroup has three main stages, similar to HAIS. The first stage predicts semantic scores and offset vectors for each point. In the second stage, the soft grouping module predicts candidate object instances based on these scores and vectors. For each semantic class, the subsets of points with a semantic score greater than a specified threshold are grouped using hierarchical aggregation. The overall set of candidate object instances is the union of the candidate instances obtained from each semantic class. In the final stage, a UNet-based module is used to refine the candidate instances by extracting features and predicting the semantic class, segmentation mask (to differentiate between foreground and background points) and mask confidence score for each candidate instance. Compared to PointGroup and HAIS, this soft grouping module, combined with the UNet-based refinement module, provides a significant boost in accuracy.

3.3 Cross-Modal Data Fusion for 3D Scene Understanding

Prior work has focused on two main types of cross-modal data fusion: sequential and joint. Sequential fusion involves training a model with 2D data in a separate optimization step before using the 2D model in a 3D fusion pipeline. In contrast, joint fusion entails training a model with 2D data and 3D data simultaneously within

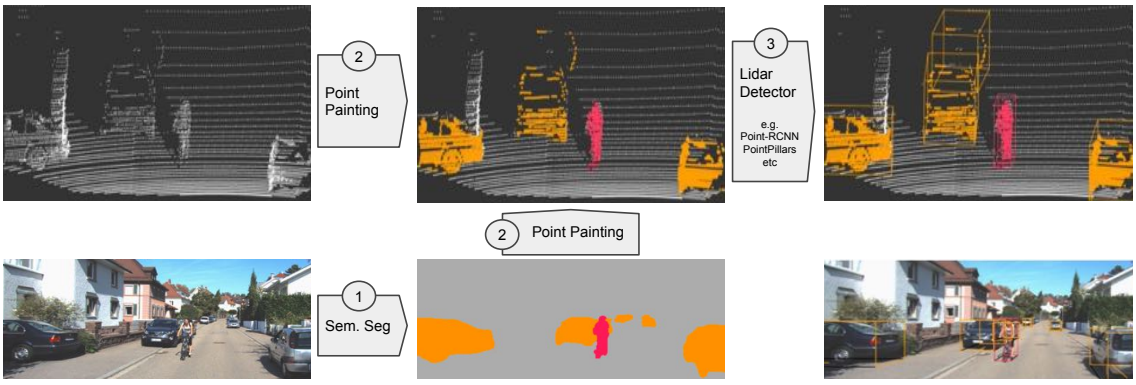


Figure 3.5: Illustration of the PointPainting method [42]. A 2D semantic segmentation model first generates semantic predictions from images. Then, each point in the input point cloud is projected into one image and the associated pixel-level semantic prediction is appended to the point. If the field of view of two cameras overlaps, some points will project onto two images simultaneously. In this case, one of the two semantic prediction vectors is randomly chosen. This 'decorated' point cloud can be processed by any object detector.

a single optimization procedure.

For 3D object detection, sequential fusion has yielded improvements compared to 3D-only baselines. An example of sequential fusion is the PointPainting [42] method. As illustrated in Figure 3.5, PointPainting [42] is a cross-modal data fusion method that successfully improves the performance of 3D object detectors by fusing 2D data at the input level of the detectors. Specifically, each point in the point cloud is projected into an image that contains the point using Equation (2.15). Then, the associated semantic-class prediction obtained from the output of a 2D semantic segmentation network is appended to the point. The augmented (painted) point cloud can be processed by any 3D object detector. PointPainting has successfully been applied to the KITTI [18] and nuScenes [6] outdoor autonomous driving datasets. In general, the worst-performing classes benefited the most from PointPainting augmentation, with a few notable exceptions. The nuScenes traffic-cone class from the nuScenes dataset showed the largest improvement, most likely because traffic cones often have very few points associated with them, and so the additional information from semantic segmentation is particularly useful. Trailers and construction vehicles showed smaller gains, despite starting from a lower baseline because the segmentation network had worse recall on these classes. These observations suggest that most gains with this fusion technique are for classes where the 2D semantic segmentation network does well compared to the 3D object detection network, and where points on the object

are generally sparse.

Other variants of PointPainting, such as bounding box PointPainting [33] and PointAugmenting [44] have also shown improvements compared to 3D-only object detector baselines. In bounding box PointPainting, bounding box-level semantics are concatenated with the point cloud, as opposed to the point-level semantics used in regular PointPainting. PointAugmenting [44] further improves PointPainting results by appending CNN features instead of 2D semantic predictions to the point cloud.

For 3D semantic segmentation, sequential fusion via a modified version of PointPainting has also improved the performance of 3D-only semantic segmentation. One example is LIF-Seg [47], which not only appends 2D semantic predictions to the input of its 3D semantic segmentation model, but also appends either a 1x1, 3x3 or 5x5 window of RGB values. These windows of RGB values are centred on the image pixel that corresponds to the point in 3D space; they provide additional 2D contextual information to the 3D semantic segmentation model. The LIF-Seg authors also explore incorporating fused 2D information at different locations within the 3D pipeline. Specifically, they experiment with early-fusion—where the 2D data is added at the input level of the 3D encoder,—and mid-fusion—where the 2D data is added at the output level of the 3D encoder. Experimentally, using the nuScenes dataset, they show that appending 2D semantic predictions with a 3x3 window of RGB values and using mid-fusion provides the largest boost in performance relative to their 3D-only baseline.

In 3D instance segmentation, one of the best-performing models that uses both 2D and 3D data is 3D-SIS [22]. Unlike the sequential fusion strategy used by PointPainting, which trains a model on 2D data independently from 3D data, 3D-SIS jointly trains with and fuses 2D and 3D data. Specifically, 2D features extracted from multi-view images are back-projected into a voxel grid and concatenated with a voxel grid of 3D features. Then, the fused 2D-3D features are used to predict object instances. The training procedure optimizes the 2D and 3D feature networks simultaneously. However, the performance of the overall model is inferior to that of leading 3D instance segmentation models that solely rely on 3D data. The inferior performance could be attributed to the less mature 2D feature extractor used by the model or that joint fusion creates a more difficult optimization problem.

3.4 Limitations

The predominant focus of prior research on 3D instance segmentation has been on the use of point clouds, with only a minority of work incorporating both image and point cloud data. Approaches solely reliant on point cloud data are constrained by smaller 3D datasets than those that are able to integrate with larger scale 2D datasets. Moreover, the sparse nature of point clouds makes segmentation more difficult—incorporating dense image features should help to improve performance. Results from experiments with PointPainting support this hypothesis. As described above, when 2D data was included, classes of objects with smaller numbers of points, such as traffic cones, experienced more significant performance boost than those with larger numbers of points, such as trailers and construction vehicles.

Currently, 3D-SIS is the best-performing fusion-based 3D instance segmentation model on the ScanNet dataset, although its performance is far below leading 3D-only instance segmentation approaches. Unlike successful fusion methods in the fields of object detection and semantic segmentation, which apply sequential fusion, 3D-SIS jointly fuses 2D and 3D data simultaneously. This increases the complexity of the optimization process, which may be the cause of the reduced performance. Sequential fusion, as is done by PointPainting, facilitates the training of a model on 2D data independently from 3D data. Subsequently, the outputs of the 2D model are integrated into the training pipeline of the 3D model, simplifying each optimization procedure. This separation allows for the use of the highest-performing 2D network with the highest-performing 3D network (as separate modules).

3.5 Summary

This chapter provided an overview of the state-of-the-art 2D semantic and 3D instance segmentation models and summarized literature describing approaches that integrate both modalities. Table 3.1 categorizes these methods.

Early 2D semantic segmentation models were significantly enhanced by the introduction of convolutional neural networks. With further improvements, the influential UNet architecture now serves as a foundational element of many contemporary segmentation models. To integrate 2D information with 3D models, our work leverages ENet, an efficient variant of UNet, and Pri3D, a network with state-of-the-art performance on the ScanNet dataset.

For 3D data, there are various models for instance segmentation of point clouds,

Image-based Methods	Point Cloud-based Methods	Fusion-based Methods
2D Semantic Segmentation	3D Instance Segmentation	3D Object Detection
FCN [30]	PointGroup [25]	PointPainting [42]
UNet [37]	HAIS [14]	Bounding Box PointPainting [33]
ENet [35]	SoftGroup [43]	PointAugmenting [44]
DeepLab [13, 10, 12, 11]		3D Semantic Segmentation
Pri3D [23]		LIF-Seg [47]
		3D Instance Segmentation
		3D-SIS [22]
		Ours

Table 3.1: Categorization of related work. Image-based methods use only 2D image data. Point Cloud-based methods use only 3D point cloud data. Fusion-based methods use both 2D image data and 3D point cloud data.

but most of the leading models use a point grouping method. SoftGroup has emerged as one of the best-performing models in this class, and we use SoftGroup as the baseline 3D instance segmentation model in our research. Lastly, we noted that prior work on cross-modal 3D object detection and semantic segmentation has yielded evidence suggesting that sequential fusion can improve 3D instance segmentation performance. In the next chapter, we describe our own sequential fusion method that incorporates 2D predictions from the ENet and Pri3D models into the SoftGroup model.

Chapter 4

Methodology

This chapter describes our approach to fusing 2D semantic segmentation outputs from ENet or Pri3D with the SoftGroup 3D instance segmentation model. We begin by discussing ENet and Pri3D in more detail, including their loss functions and their outputs. Then, we review several important aspects of SoftGroup, with a focus on the loss function of the model. Lastly, we explain our cross-modal data fusion method.

4.1 Models for Semantic Segmentation in 2D

The usual goal of 2D semantic segmentation is to assign a class label to every image pixel. However, in our fusion architecture, the 2D semantic segmentation network extracts a feature vector for each pixel, which is then used by the 3D instance segmentation model at a later stage. We call this vector an *image-based fusion vector* and explore four different representations: features, logits, softmax scores or one-hot encoded predictions obtained from the 2D semantic segmentation network. One fusion vector is required per pixel, and while logits, softmax scores, and one-hot encoded predictions have a one-to-one correspondence with image pixels, features do not. The height and width of the extracted feature map are smaller than the height and width of the image, so we use bilinear interpolation to generate an interpolated feature map that has the same size as the image.

To determine the effects of 2D model accuracy on fusion results, we experiment with two different models: a 'lighter' fusion architecture with moderate performance and a 'heavier' architecture with state-of-the-art performance. For the lighter network, we use ENet [35], which is a UNet-based architecture (discussed in Section 3.1) that achieves 36.1 mIoU on the validation set of the ScanNet Frames 25k dataset [16] (datasets are described in more detail in Section 5.1.1). The encoder of ENet uses

max-pooling, and regular, dilated and asymmetric convolutional layers, while the decoder uses regular and transposed convolutional layers. For the heavier network, we use Pri3D [23], which is a ResUNet-based [46] architecture (discussed in Section 3.1) that achieves 62.3 mIoU on the validation set of the ScanNet Frames 25k dataset. The encoder of Pri3D is a standard ResNet and the decoder contains convolutional layers and bilinear interpolation layers.

The loss function used to train both networks is the multi-class cross-entropy loss, given by Equation (2.31), averaged across all pixels in the image. Other variants of the cross-entropy loss function, such as the focal loss [29] function could also be used. However, the multi-class cross-entropy loss is the most widely used loss function for 2D semantic segmentation and is also used by both original models, so we adopt it for our training process. The loss function is

$$\mathcal{L}_{\text{semantic}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{\text{CE}}(\mathbf{y}_i, \mathbf{t}_i), \tag{4.1}$$

where N is the total number of pixels per image, $\mathbf{y}_i$ is the one-hot encoded prediction at pixel i, and $\mathbf{t}_i$ is the one-hot encoded target class of pixel i.

4.2 A Model for Instance Segmentation in 3D

SoftGroup, our baseline 3D instance segmentation network, is one of the top-performing models released to date, achieving 67.3 mAP_{50} (reproduced) on the ScanNet 3D instance segmentation benchmark. The architecture is discussed in Section 3.2; here, we describe the loss functions for multiple branches of the network: semantic, offset, instance classification, instance segmentation, and instance mask scoring (shown in Figure 3.4).

The combined loss function consists of five separate terms, one for each branch of the network. The semantic branch computes semantic scores $\mathbf{S} = \{\mathbf{s}_1, \cdots, \mathbf{s}_N\} \in \mathbb{R}^{N \times N_{class}}$ over all N_{class} classes for each of the N points in the point cloud. The offset branch computes offset vectors $\mathbf{O} = \{\mathbf{o}_1, \cdots, \mathbf{o}_N\} \in \mathbb{R}^{N \times 3}$, where each vector has its tail at a point and its tip at the predicted instance centroid. Cross-entropy loss from Equation (2.31) and L1 loss from Equation (2.23) are used for the semantic and offset branches, respectively. They are defined as

$$\mathcal{L}_{\text{semantic}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{CE}(\mathbf{s}_i, \mathbf{s}_i^*) \tag{4.2}$$

and

$$\mathcal{L}_{\text{offset}} = \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{\{\mathbf{p}_i\}}} \sum_{i=1}^{N} \mathbb{1}_{\{\mathbf{p}_i\}} \mathcal{L}_{L1}(\mathbf{o}_i, \mathbf{o}_i^*), \tag{4.3}$$

where $\mathbf{s}^*$ and $\mathbf{o}^*$ are the semantic and offset labels respectively, and $\mathbb{1}_{\{\mathbf{p}_i\}}$ is an indicator function that determines whether the point $\mathbf{p}_i$ belongs to a specific instance.

The remaining three network branches (instance classification, instance segmentation and instance mask scoring) generate the final, predicted instances from candidate ones. The instance classification branch computes semantic scores $\mathbf{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_K\} \in \mathbb{R}^{K \times N_{\text{class}+1}}$ for each instance, where K is the number of instances. In addition to the N_{class} semantic classes, there is one additional 'background' class that is used to account for unlabelled points in the dataset. The instance segmentation branch predicts a mask within each candidate instance to determine whether each point in the candidate is a foreground point (that belongs to the instance) or a background point (that does not belong). The mask scoring branch computes a score $\mathbf{E} = \{\mathbf{e}_1, \cdots, \mathbf{e}_K\} \in \mathbb{R}^{K \times N_{\text{class}}}$ that is an estimate of the IoU of each predicted mask, compared against its ground-truth mask. The final confidence score for each instance is computed by multiplying the mask score by the classification score. The classification loss, mask loss and mask score loss use the cross entropy (from Equation (2.31)), binary cross entropy (from Equation (2.29)) and L2 loss (from Equation (2.22)) functions, which are

$$\mathcal{L}_{\text{class}} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{CE}(\mathbf{c}_k, \mathbf{c}_k^*), \tag{4.4}$$

$$\mathcal{L}_{\text{mask}} = \frac{1}{\sum_{k=1}^{K} \mathbb{1}_{\{\mathbf{m}_k\}}} \sum_{k=1}^{K} \mathbb{1}_{\{\mathbf{m}_k\}} \mathcal{L}_{BCE}(\mathbf{m}_k, \mathbf{m}_k^*), \tag{4.5}$$

and

$$\mathcal{L}_{\text{mask\_score}} = \frac{1}{\sum_{k=1}^{K} \mathbb{1}_{\{\mathbf{r}_k\}}} \sum_{k=1}^{K} \mathbb{1}_{\{\mathbf{r}_k\}} \mathcal{L}_{L2}(\mathbf{r}_k, \mathbf{r}_k^*). \tag{4.6}$$

The values c^*, m^*, and r^* are the classification, segmentation and mask score labels, respectively. Here, K is the total number of candidates and $\mathbb{1}_{\{\cdot\}}$ indicates whether a specific candidate belongs to a ground-truth instance.

4.3 Cross-Modal Data Fusion

We now describe the two fusion architectures that we explore in this thesis. Each architecture inserts the image-based fusion vectors at a different location in the 3D in-

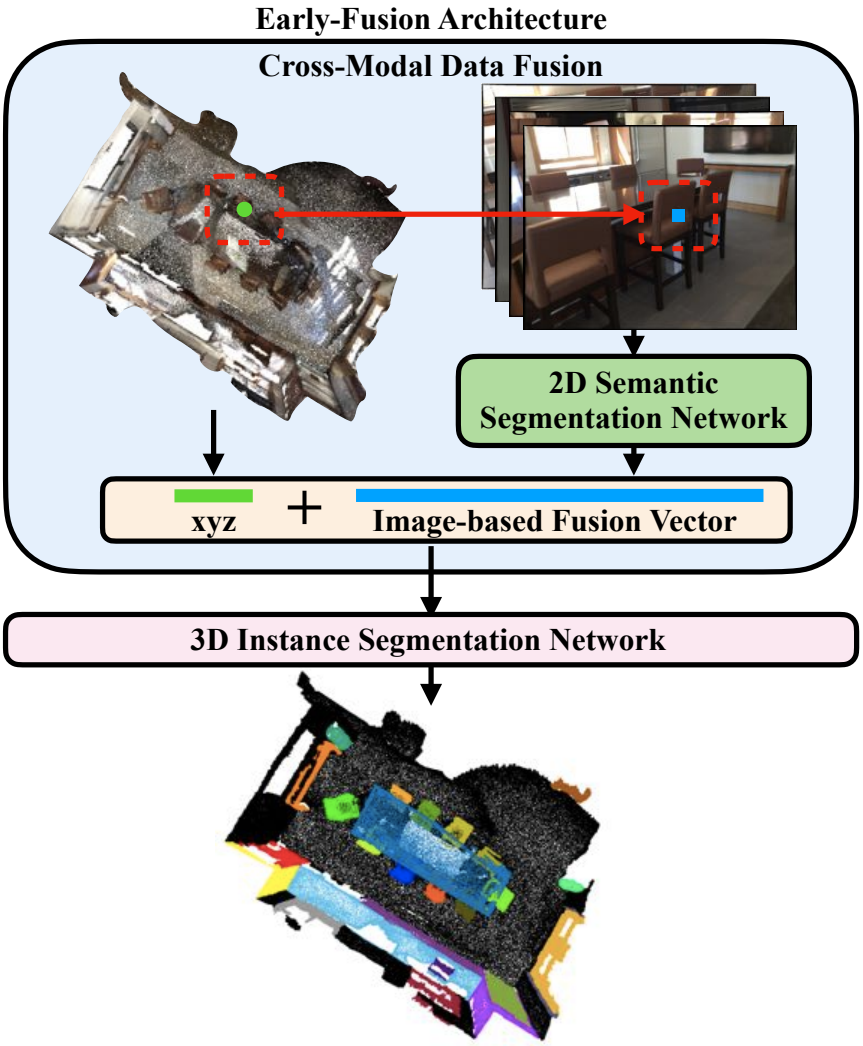


Figure 4.1: Our early-fusion architecture. The architecture consists of three stages: (1) 2D semantic segmentation to obtain an image-based fusion vector for each pixel. This vector can either be an encoded feature or the segmentation prediction in the form of class logits, softmax scores, or one-hot encodings. (2) Cross-modal data fusion, appending the image-based fusion vector to each point. (3) 3D instance segmentation on the decorated point cloud to produce the instance-segmented point cloud. The segmented cloud is shown at the bottom of the figure.

stance segmentation pipeline. Additionally, we discuss our two matching algorithms—an initial version and an improved version—that are used to match each point with an image-based fusion vector.

We run our experiments with an early-fusion architecture (shown in Figure 4.1) and a mid-fusion architecture (shown in Figure 4.2) to determine which fusion location maximizes performance. In the early-fusion design, the image-based fusion vectors

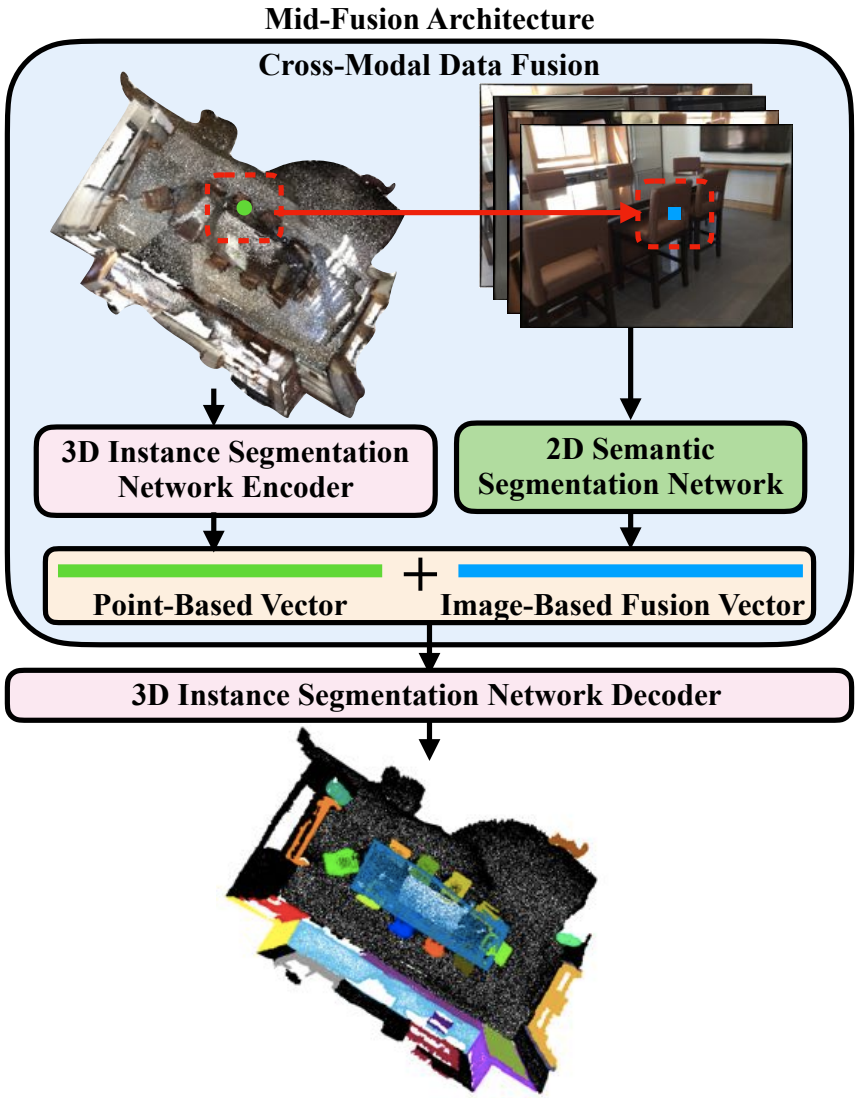


Figure 4.2: Our mid-fusion architecture. Similar to the early-fusion design, the mid-fusion architecture consists of the same three stages: 2D semantic segmentation, cross-modal data fusion, and 3D instance segmentation. However, in mid-fusion, the image-based fusion vector is appended to the point cloud-based vector obtained from the 3D encoder.

are integrated at the input stage of the 3D encoder by appending the fusion vectors to the x, y, and z coordinates of the point cloud. Conversely, in the mid-fusion architecture, these fusion vectors are inserted between the 3D encoder and decoder by appending the fusion vectors to the point-based vectors (specifically, to the 3D semantic segmentation logits before fusion) extracted from the 3D encoder. Then, a feed-forward layer is used to process the concatenated vector to generate the 3D semantic segmentation logits after fusion.

Algorithm 1 Multi-Frame PointPainting($\mathbf{L}, \mathbf{D}, \mathbf{F}, \mathbf{E}, \mathbf{K}, s$)

Inputs:
Point cloud $\mathbf{L} \in \mathbb{R}^{N,3}$ with N points, where $\mathbf{l}_i \in \mathbb{R}^3$.
Depth images $\mathbf{D} \in \mathbb{R}^{M,W,H,1}$ with M images, where $d_{j,u,v} \in \mathbb{R}$.
Image-based fusion vectors $\mathbf{F}^{M,W,H,C}$ with C channels, where $\mathbf{f}_{j,u,v} \in \mathbb{R}^C$.
Camera extrinsic parameter matrices $\mathbf{E} \in \mathbb{R}^{M,4,4}$, where $\mathbf{E}_j \in \mathbb{R}^{4,4}$.
Camera intrinsic parameter matrix $\mathbf{K} \in \mathbb{R}^{3,4}$.
Image increment value s.

Output:
Decorated point cloud $\mathbf{P} \in \mathbb{R}^{N,3+C}$, where $\mathbf{p}_i \in \mathbb{R}^{3+C}$.

for $i = 0$ to $N - 1$ **do**
  for $j = 0$ to $M - 1$ incrementing every s indices **do**
    $u, v, d =$ PROJECT($\mathbf{l}_i, \mathbf{E}_j, \mathbf{K}$)
    if $u \in [0, W)$ **and** $v \in [0, H)$ **and** $|d - d_{j,u,v}| \leq 1$ **then**
      $\mathbf{p}_i =$ CONCATENATE($\mathbf{l}_i, \mathbf{f}_{j,u,v}$)
      break
    end if
  end for
end for

We use a modified version of the PointPainting algorithm to match an image-based fusion vector to each point and adapt the algorithm to work on indoor 3D datasets. The original PointPainting algorithm was designed for use with outdoor autonomous vehicle datasets. In these datasets, it is common to have just one or two images associated with each point cloud. This simplifies the matching process because each point is only projected onto one or two images to determine a pixel-fusion vector correspondence. Note that we use pixel correspondence and fusion vector correspondence interchangeably here because, as we recall from Section 4.1, there is a one-to-one correspondence between image pixels and image-based fusion vectors. In contrast to outdoor 3D datasets, indoor 3D datasets generally use dense, reconstructed point clouds, so thousands of images can be associated with each point cloud.

Our initial adaptation of the PointPainting algorithm, called Multi-Frame PointPainting and described in Algorithm 1, accommodates a large number of image frames by selecting the first matching image-based fusion vector for each point. The Multi-Frame PointPainting algorithm has two stages. In the first stage, a point is projected into each image using Equation (2.15) and Equation (2.17) until the first pixel correspondence for that point is found. A pixel correspondence is found when the

Algorithm 2 Guided Multi-Frame PointPainting($\mathbf{L}, \mathbf{D}, \mathbf{F}, \mathbf{E}, \mathbf{K}, s$)

Inputs:
Point cloud $\mathbf{L} \in \mathbb{R}^{N,3}$ with N points, where $\mathbf{l}_i \in \mathbb{R}^3$.
Depth images $\mathbf{D} \in \mathbb{R}^{M,W,H,1}$ with M images, where $d_{j,u,v} \in \mathbb{R}$.
Image-based fusion prediction vectors $\mathbf{F}^{M,W,H,C}$ with C channels, where $\mathbf{f}_{j,u,v} \in \mathbb{R}^C$.
Camera extrinsic parameter matrices $\mathbf{E} \in \mathbb{R}^{M,4,4}$, where $\mathbf{E}_j \in \mathbb{R}^{4,4}$.
Camera intrinsic parameter matrix $\mathbf{K} \in \mathbb{R}^{3,4}$.
Image increment value s.

Output:
Decorated point cloud $\mathbf{P} \in \mathbb{R}^{N,3+C}$, where $\mathbf{p}_i \in \mathbb{R}^{3+C}$.

for $i = 0$ to $N - 1$ **do**
  prediction_vector_list = []
  for $j = 0$ to $M - 1$ incrementing every s indices **do**
    u, v, d = PROJECT($\mathbf{l}_i, \mathbf{E}_j, \mathbf{K}$)
    if $u \in [0, W)$ **and** $v \in [0, H)$ **and** $|d - d_{j,u,v}| \leq 1$ **then**
      prediction_vector_list.APPEND($\mathbf{f}_{j,u,v}$)
    end if
  end for
  class_index = GET_TOP_PRED(prediction_vector_list)
  $\mathbf{f}$ = SELECT_VECTOR(prediction_vector_list, class_index)
  $\mathbf{p}_i$ = CONCATENATE($\mathbf{l}_i, \mathbf{f}_{j,u,v}$)
end for

computed pixel coordinates lie within the image and the computed depth value is within one millimetre of the ground-truth depth value (this last threshold is a tuning parameter). Unlike the original PointPainting algorithm that only uses colour images, our algorithm also uses depth images to improve matching accuracy. The ground-truth depth value is obtained by indexing into the depth image at the computed pixel coordinates. In the second stage, the image-based fusion vector associated with the computed pixel coordinates is concatenated with the point. These two stages of processing are repeated for all points in the point cloud.

The improved version of our algorithm, called Guided Multi-Frame PointPainting and described in Algorithm 2, matches to each point a fusion vector that is associated with the most frequently predicted semantic class across all matching fusion vectors. The Guided Multi-Frame PointPainting algorithm has three stages of processing. First, unlike Multi-Frame PointPainting, each point is projected into every image to find all pixel correspondences for that point. In the second stage, the most frequently predicted semantic class associated with each pixel correspondence is determined. In

the final stage, the first fusion vector that is associated with the most frequently predicted semantic class is concatenated to the point.

The main advantage of Guided Multi-Frame PointPainting over the original PointPainting algorithm is that only a fusion vector associated with the most frequently predicted semantic class from all correspondences is concatenated to each point. By leveraging the information from multiple correspondences, this approach significantly improves accuracy when matching fusion vectors to points. As we shall see in Chapter 5, increasing the matching accuracy also improves the performance of 3D instance segmentation post-fusion. The Guided Multi-Frame PointPainting algorithm also includes depth images as an additional input to further increase the accuracy of our matching algorithms. For example, two points with different depth values measured from the position of a depth camera (as computed by Equation (2.17)) can be projected to the same pixel coordinates (as computed by Equation (2.15)) when they are intersected by the same ray projected from the camera. To disambiguate the correct point-to-pixel correspondence from the incorrect one, we incorporate a condition in our algorithms to only select correspondences with projected depth values that are within one millimetre of the ground-truth depth values obtained from the depth image. Lastly, due to the vast number of images in the ScanNet dataset, we run our algorithms on a subset of images by using an image increment value s. We skip every ten images for Algorithm 1 and every three images for Algorithm 2 to reduce computation time.

4.4 Summary

In this chapter, we described the 2D semantic segmentation models and the 3D instance segmentation model that we extend and incorporate in our sequential fusion framework. We also discussed how our sequential fusion approach has been adapted for indoor 3D datasets. Initially, we delved into four possible representations of our image-based fusion vector that are derived from the 2D semantic segmentation model. Subsequently, we described our two selected 2D model architectures, namely ENet and Pri3D, and their performance on the ScanNet Frames 25k dataset. Additionally, we explained the loss functions employed by SoftGroup. Finally, we discussed the design of our early-fusion and mid-fusion architectures and the two matching algorithms used to select an image-based fusion vector to concatenate with each point or point-based vector. In the next chapter, we explain how we evaluated our cross-modal data fusion framework and present the results.

Chapter 5

Experiments

In this chapter, we discuss the datasets we use for training and evaluation, our evaluation metrics, and the implementation details of our models. Subsequently, we discuss the findings of our early-fusion experiments by evaluating performance with different representations of the image-based fusion vector (i.e., a model-based output), as well as fusing with non-model-based outputs. From the early-fusion experiments, we analyze the effect of sequential fusion on 3D semantic and on 3D instance segmentation. We further analyze the relationship between post-fusion instance segmentation performance and the accuracy of point-to-pixel correspondences, as well as the relationship between post-fusion instance segmentation performance and the performance of the 2D semantic segmentation model. Then, we discuss the findings of our mid-fusion experiments and compare them with the early-fusion experiments. Lastly, we perform an ablation study comparing different masking augmentation strategies to determine the effect on performance when certain parts of the input data are removed.

5.1 Experimental Settings

Two datasets were used to train our 3D instance segmentation network (i.e., SoftGroup) and our 2D semantic segmentation networks (i.e., ENet and Pri3D). Below, we describe these datasets, the evaluation metrics used for each task, and the hyperparameters used for each model.

5.1.1 ScanNet Dataset

In this thesis, we train and evaluate our 3D instance segmentation model using the ScanNet dataset [16]. ScanNet is the largest indoor dataset that offers both anno-

tated 2D and 3D data and is comprised of 2.5 million images from 1,513 RGB-D scans, covering 80,000 m^2 and encompassing 707 unique indoor scenes. In contrast, similar datasets such 2D-3D-S [2] contain 70,000 images in approximately 1,400 scans, covering 6,000 m^2. The reconstructed point clouds in ScanNet are accompanied by 3D camera poses, surface reconstructions, and instance-level semantic segmentations. The dataset is divided into a training, validation and test set, which consists of 1,201, 312, and 100 scans, respectively. The annotations include 18 labelled instance classes: bathtub, bed, bookshelf, cabinet, chair, counter, curtain, desk, door, other furniture, picture, refrigerator, shower curtain, sink, sofa, table, toilet, and window. The annotations also include three background classes: floor, wall, and unannotated.

The substantial size of ScanNet was achieved through the development of its custom labelling pipeline designed for annotation efficiency. The process involved capturing an RGB-D video sequence, uploading it to a cloud server, and reconstructing it into a 3D mesh. Crucially, the 3D mesh is subdivided into small segments using an over-segmentation algorithm to enable annotators to label each segment instead of each point, which significantly improves labelling efficiency.

We train and evaluate our 2D semantic segmentation models using the ScanNet Frames 25k dataset. ScanNet Frames 25k [16] comprises 25,000 images and corresponding semantic labels extracted from the ScanNet dataset. These images were obtained by selecting every 100th image from the complete set of ScanNet images.

5.1.2 Evaluation Metrics

Following previous work, we use the mean intersection over union (mIoU) evaluation metric for 2D and 3D semantic segmentation, because it measures the amount of overlap between predicted and ground-truth masks. We use mean average precision based on a 50% IoU threshold (mAP_{50}) as the evaluation metric for 3D instance segmentation because it provides a single metric to measure the overall precision and recall of the predictions. Section 2.5 provides the details for calculating these metrics.

5.1.3 Implementation Details

Our models are built using PyTorch [36] and we use hyperparameter values that are similar to the values used by the authors of each respective model. All experiments that train the 3D instance segmentation model, SoftGroup [43], are trained on two Nvidia V100 GPUs and all experiments that train a 2D semantics segmentation model (i.e., ENet [35] or Pri3D [23]) are trained on one Nvidia Quadro 8000 GPU.

2D Semantic Segmentation Model - ENet. ENet is trained using a batch size of two RGB-D images, resized to dimensions of $[240, 320]$ (to reduce computational load), for a total of 200 epochs with an initial learning rate set at 0.001. Adam is used for optimization with beta parameters of $\{0.7, 0.999\}$. The learning rate scheduler performs learning rate decay by applying a multiplicative factor of 0.1 on the learning rate every 60 epochs. An L2 regularization factor of 0.0002 is used.

2D Semantic Segmentation Model - Pri3D. We employ the Pri3D model variant that uses a ResNet50 encoder and initialize the model using pre-trained weights encompassing view-invariant and geometric priors. Then, we fine-tune the weights of the model by training the model on the ScanNet Frames 25k dataset. For fine-tuning, we use a batch size of eight RGB images resized to dimensions of $[240, 320]$ for a total of 80 epochs with an initial learning rate set at 0.1. Stochastic gradient descent is used for optimization with a momentum parameter of 0.9. The learning rate scheduler performs learning rate decay by applying a polynomial decay rate of 0.9 over the total number of epochs.

3D Instance Segmentation Model - SoftGroup. For all SoftGroup experiments, we train the model using a batch size of eight for a total of 128 epochs with an initial learning rate set at 0.002. The Adam optimizer is used with beta parameters set to $\{0.9, 0.999\}$. After epoch 50, we use a cosine annealing learning rate schedule that starts with the initial learning rate and then decreases it to a value of $1e{-}6$ before increasing the learning rate back to its initial value.

5.2 Results from the Early-Fusion Experiments

We explain the findings of our early-fusion experiments in this section. We begin by discussing our baselines and the initial tests of using ImageNet features extracted from the ScanNet images to perform early-fusion. Next, we analyze the results of fusing model-based outputs (i.e., the image-based fusion vectors) and the results of fusing non-model-based outputs into SoftGroup. From these analyses, we discuss the importance of the 2D semantic segmentation performance on post-fusion performance. Lastly, we discuss the importance of point-to-pixel alignment on the post-fusion performance that led us to design our Guided Multi-Frame PointPainting algorithm.

Method	First Conv	Encoder	mAP_{50}	$mIoU_{3D}$
SoftGroup (Repr) [43]	F	F	67.3	72.0
SoftGroup	U	F	68.0	-
SoftGroup	U	U	65.5	-
+ Sem. 2D Labels	U	F	75.0	-
+ Sem. 2D Labels	U	U	**81.8**	87.9
+ ImageNet Feats	U	F	67.1	-
+ ImageNet Feats	U	U	**67.7**	70.1
+ 3x3 RGB	U	U	64.9	-
+ 5x5 RGB	U	U	64.9	-
+ 3x3 RGBD	U	U	65.9	-
+ 5x5 RGBD	U	U	**66.2**	68.4
+ ENet Encodings	U	U	65.3	-
+ ENet Scores	U	U	**66.4**	70.5
+ ENet Logits	U	U	65.8	-
+ ENet Feats	U	U	66.0	-
+ Pri3D Encodings	U	U	63.9	-
+ Pri3D Scores	U	U	**64.4**	68.1
+ Pri3D Scores	R	R	57.3	-
+ Pri3D Logits	U	U	63.9	-
+ Pri3D Feats	U	U	63.7	-

Table 5.1: Early-fusion 3D instance segmentation results evaluated on ScanNet v2 validation set where values are bolded for the best performing experiment in each experiment set. Repr is our reproduced SoftGroup trained from scratch using two V100 GPUs. F stands for frozen weights initialized with HAIS [14] pre-trained weights, U stands for unfrozen weights initialized with HAIS pre-trained weights, and R stands for unfrozen weights that are randomly initialized. The second column denotes whether the weights of the first convolutional layer of the model is frozen or unfrozen. The third column denotes whether the weights of the rest of the SoftGroup encoder, excluding the first convolutional layer of the encoder, is frozen or unfrozen. Corresponding 3D semantic segmentation results are generated for the reproduced baseline and the best-performing experiment for each experiment set. We observe that, in general, higher semantic segmentation performance correlates with higher instance segmentation performance.

5.2.1 Baselines

We generated three baselines to compare with the early-fusion experiments, as shown in the first three rows of Table 5.1. SoftGroup was reproduced on our hardware by following the original weight initialization procedure designed by the authors of the model. Specifically, the weights of the SoftGroup encoder were initialized with the weights of a pre-trained HAIS encoder. Then, those weights were frozen during

training. Freezing of weights refers to the practice of fixing the parameters of the model, which prevents them from being modified during the training process. This proves valuable in preserving the acquired knowledge of a pre-trained model, such as the pre-trained HAIS model. Unfrozen weights refer to parameters that can be adjusted during the training process. Hence, the original SoftGroup training process only adjusts the weights of the decoder. However, in early fusion, the input size of each point in the point cloud is increased by the length of the image-based fusion vector. Consequently, it is necessary to unfreeze either the SoftGroup encoder or, at the very minimum, the input convolutional layer of the model to accommodate the processing of the extra input channels. Therefore, we trained two additional baselines: one that unfreezes the weights of the first convolutional layer while keeping the rest of the encoder frozen and another baseline that unfreezes the weights of the entire encoder, including the first convolutional layer.

We observed that for the original SoftGroup model, having a fully frozen encoder significantly improved performance compared to using an unfrozen encoder during training. Surprisingly, we also observed that unfreezing the first convolutional layer while keeping the rest of the encoder frozen resulted in improved performance compared to the original SoftGroup architecture that trained with a fully frozen encoder. This suggests that the weights of the first layer in the neural network have a significant impact on the overall performance. This is likely because the features extracted from the first layer impact all subsequent layers in the network. Hence, in general, it may be beneficial to unfreeze the first layer of an encoder when initializing with pre-trained weights.

5.2.2 Fusing Model-Based Outputs

We conducted an initial test to assess the potential improvements of image-based fusion into SoftGroup (see the third set of experiments in Table 5.1). We utilized an off-the-shelf pre-trained UNet, trained on the ImageNet dataset, to generate per-pixel features. These features were then appended to each point in the input point cloud using Algorithm 1. The resulting augmented point cloud was used to train SoftGroup on the ScanNet dataset. From this test, we observed that, in early fusion, fully unfreezing the encoder yielded better performance than only unfreezing the first input convolutional layer. This is likely because unfreezing the entire encoder provides more model capacity that is necessary to process the additional image-based data. Consequently, for the subsequent early-fusion experiments, all model weights were

unfrozen during training. Additionally, we noticed a slight performance boost when fusing with ImageNet-trained features compared to the reproduced baseline. This promising result led us to hypothesize that using 2D model outputs trained on the same dataset (i.e., ScanNet) instead of ImageNet would further enhance performance.

We found that the performance of the experiments that fused ENet or Pri3D outputs with SoftGroup surpassed the performance of the baseline with a fully unfrozen encoder, but these experiments did not surpass its performance with a fully frozen encoder. We also observed that for both ENet-fused and Pri3D-fused experiments, "soft" information (i.e., semantic scores) yielded the highest post-fusion performance, while "hard" information (i.e., one-hot encoded predictions) often yielded the lowest post-fusion performance. Semantic scores are considered "soft" information because the scores provide a probability distribution over all classes, allowing for uncertainty in the predictions of the model. One-hot encoded predictions are considered "hard" information because all values are set to zero except for one value that is set to one, so these encodings do not provide any uncertainty information. Our finding that soft information provides better performance than hard information aligns with the findings outlined in the knowledge distillation paper by Hinton et al. [21]. Knowledge distillation is a model compression technique where a smaller model is trained to replicate the predictions of a larger model by training the smaller model to predict the soft targets (i.e., the softmax scores) of the larger model. The authors found that when the soft targets have high entropy, they provide much more information compared to hard targets, so the smaller model can be trained on much less data than the larger model and use a much higher learning rate. The SoftGroup paper also demonstrates the prevailing trend that soft information is more effective than hard for transferring information. As we recall from Section 3.2, the main innovation achieved by SoftGroup, elevating its performance above HAIS, was its associatiation of each point with multiple semantic classes instead of just a single one. This approach helped reduce the errors of semantic segmentation that would propagate to instance segmentation predictions.

One more observation we made was that initializing the SoftGroup encoder with the pre-trained HAIS encoder weights was crucial for performance. Comparing the Pri3D scores experiments with randomly initialized weights versus pre-trained HAIS weights in Table 5.1, we observed a significant drop in performance for the experiment with randomly initialized weights. Both experiments had all model weights unfrozen during training. This shows that even when weights are unfrozen during training, the initialization of pre-trained weights significantly outperforms the initialization of

random weights.

5.2.3 Fusing Non-Model-Based Outputs

(a) RGB Image (b) Depth Map

Figure 5.1: Example of an RGB image and depth map [7].

In addition to fusing with model-based outputs extracted from ENet or Pri3D, we also explored the fusion of non-model-based outputs by following a method similar to LIF-Seg. LIF-Seg [47], as discussed in Section 3.3, demonstrated successful improvements in 3D semantic segmentation performance using the nuScenes outdoor dataset by appending RGB values (i.e., non-model-based outputs) from the images to the points directly. Following a similar approach, we conducted the experiments in Table 5.1 using either a 3x3 or a 5x5 window of RGB or RGB-D values to determine the effect on 3D instance segmentation performance using the ScanNet indoor dataset. The RGB-D-fused experiments performed better than their RGB-fused counterparts, which indicates that the additional depth value is useful for segmentation. Depth information plays a valuable role in segmentation tasks due to the typically narrow range of depth measurements across all pixels belonging to the same object. For example, humans can intuitively discern objects based on a depth map, as illustrated in Figure 5.1. Our experiments revealed that a 5x5 window yielded marginally better performance than a 3x3 window. Similar to the ENet and Pri3D experiments, the RGB-D-fused experiments surpassed the performance of the baseline with a fully unfrozen encoder but had lower performance than with a fully frozen encoder.

There are a few potential reasons for the difficulty in gaining post-fusion improvements in our experiments compared to the post-fusion improvements observed in prior work. First, prior work evaluated cross-modal data fusion on outdoor datasets, which tend to consist of sparse point clouds. As we recall from Section 3.3, the results from

PointPainting had the largest improvement on classes which often had fewer points on them, such as traffic cones. Therefore, dense image data can be more effectively utilized to complement sparse point clouds. In contrast, we are using an indoor dataset that consists of dense reconstructed point clouds, so the dense image data may not have as large an impact on performance improvements. Second, prior work using variants of the PointPainting algorithm have primarily concentrated on tasks such as 3D object detection and semantic segmentation. These tasks are of a higher-level nature compared to the more detailed task of instance segmentation, making it potentially more challenging to attain significant improvements after the fusion process. Last, in prior work, the 2D semantic segmentation model demonstrated strong performance compared to a 3D model with relatively weak performance. However, in our work, the 3D instance segmentation model performs well and the 2D semantic segmentation model does not perform notably better than the 3D model. Consequently, any enhancements in post-fusion performance may be relatively modest. The findings in the PointPainting paper also indicated that after the fusion process, improvements were less significant on classes where the 2D semantic segmentation model exhibited lower recall, so the relative performance between the 2D and 3D model is an important factor for post-fusion performance. In the following subsection, we further analyze the class-wise metrics of the experiments and discuss two of the main factors that affect post-fusion performance.

5.2.4 Main Factors that Impact Post-Fusion Performance

We compared the quality of predictions obtained from each 2D semantic segmentation model to gain deeper insights into the modest improvements from the early-fusion approach. The results of this comparison are showcased in Table 5.2. In the second column of the table, we computed the 2D semantic segmentation performance of each method. In the third column, we computed the 3D semantic segmentation

Method	mIoU_{2D}	Projected mIoU_{3D}
ENet 2D Preds	36.1	27.1
Pri3D 2D Preds	62.3	38.3
Sem. 2D Labels	100.0	61.9

Table 5.2: 2D semantic segmentation (mIoU) results evaluated on ScanNet Frames 25k validation set and 3D semantic segmentation results computed by back-projecting the 2D predictions into 3D and directly computing the mIoU metrics.

performance of each method if the 2D semantic segmentation predictions were back-projected into 3D using Multi-Frame PointPainting. We made two key observations:

1. While Pri3D exhibited superior performance in semantic segmentation compared to ENet (shown in Table 5.2), ENet-fused SoftGroup exhibited superior performance in 3D instance segmentation compared to Pri3D-fused SoftGroup (shown in Table 5.1). This outcome was unexpected and merited further investigation.

2. Comparing the second and third columns of Table 5.2, we observe that the mIoU metric was significantly lower after the semantic segmentation predictions were back-projected from 2D to 3D. This indicated inaccuracies in the point-to-pixel matching algorithm when transitioning between the two modalities.

These two observations highlighted the need for a deeper analysis to understand the factors influencing the performance of 3D instance segmentation after fusion.

Projection Source	$mIoU_{3D}$	Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Desk	Curtain	Refrigerator	S. Curtain	Toilet	Sink	Bathtub	Other Furn.
ENet 2D Preds	27.1	45.3	**70.9**	22.4	34.6	32.3	31.3	29.0	20.1	20.0	27.6	12.3	24.9	24.6	24.8	17.0	18.9	20.8	20.1	33.7	11.2
Pri3D 2D Preds	38.3	**54.4**	70.3	**29.8**	**49.1**	**45.4**	**44.4**	**38.0**	**27.9**	**25.4**	**45.2**	**12.7**	**30.4**	**31.1**	**30.7**	**24.9**	**25.8**	**58.4**	**40.3**	**55.6**	**25.5**
Sem. 2D Labels	61.9	63.4	79.0	58.8	69.5	66.4	71.3	66.4	49.0	45.0	57.3	35.2	62.7	59.7	52.7	57.7	65.4	71.6	63.3	75.3	67.8
Sem. Seg. Method	$mIoU_{3D}$																				
SoftGroup (Repr) [43]	72.0	85.3	95.0	64.7	80.1	89.2	78.7	74.6	63.9	65.4	76.9	31.1	65.3	67.6	72.4	59.9	71.1	91.8	63.9	85.1	59.0
+ ENet Logits	70.8	**83.6**	**95.0**	**66.3**	79.7	**88.8**	**77.4**	**73.9**	**60.5**	**62.9**	75.5	25.5	**64.8**	**65.8**	**73.2**	**55.2**	65.3	**93.1**	**62.7**	**87.9**	**58.0**
+ Pri3D Logits	68.7	83.0	94.6	58.9	**80.1**	88.8	75.2	73.3	59.5	59.6	**77.1**	**30.2**	56.6	62.4	67.7	52.8	**69.5**	88.5	62.7	80.7	53.1
+ Sem. 2D Labels	87.9	92.6	96.8	90.0	92.2	95.0	94.9	92.6	83.1	87.2	93.5	51.6	80.7	89.6	88.8	78.9	90.6	95.1	82.3	92.6	90.3
Inst. Seg. Method	mAP_{50}																				
SoftGroup (Repr) [43]	67.3	-	-	60.0	77.2	84.6	69.5	75.9	53.8	52.7	67.7	57.4	35.5	58.9	52.8	74.7	73.8	100.0	68.6	86.6	62.1
+ ENet Logits	65.8	-	-	**62.2**	74.5	**85.8**	**71.0**	**77.7**	**52.0**	50.0	52.7	48.6	**37.1**	**60.8**	54.7	**66.2**	**74.9**	**100.0**	**69.4**	**86.9**	**59.3**
+ Pri3D Logits	63.9	-	-	57.1	**79.8**	83.3	67.3	74.6	49.2	**50.1**	**60.9**	**51.3**	30.1	52.5	**61.4**	60.5	68.1	99.8	65.8	85.4	52.3
+ Sem. 2D Labels	81.8	-	-	79.8	95.1	86.8	83.6	91.9	61.2	74.8	78.6	57.0	75.7	85.5	70.5	87.5	95.8	100.0	76.6	94.1	77.3

Table 5.3: All results are evaluated on the ScanNetv2 validation set using early fusion and point-pixel matching is performed using Multi-Frame PointPainting with a skipping strategy of ten frames. The first set of results is computed by back-projecting 2D semantic segmentation predictions or ground-truth labels into 3D and directly computing 3D semantic segmentation metrics. The second set of experiments computes the 3D semantic segmentation metrics for the reproduced baseline and the experiments which fuse the 2D logits or labels into the input point cloud. The last set of results computes the corresponding 3D instance segmentation metrics for the same experiments as the second set of experiments. Bolded values represent the higher performing metric compared between the ENet and Pri3D experiments for each of the three sets of results. In general, we observe that for both aggregated and class-wise metrics, 3D instance segmentation improves when 3D semantic segmentation improves.

Observation 1: Importance of 2D Semantic Segmentation Quality

Our initial investigation focused on understanding the relationship between 2D semantic segmentation quality and its impact on post-fusion 3D instance segmentation. In particular, from the first key observation, we wanted to understand why there was a reversal in relative performance between ENet and Pri3D before and after fusing with SoftGroup. We hypothesized that a small number of classes might have performed much more poorly with Pri3D compared to ENet, resulting in a reduced aggregated metric. To verify this hypothesis, we generated class-wise metrics in Table 5.3. The first set of results gives the aggregated and class-wise metrics for 3D semantic segmentation when the 2D predictions are back-projected into 3D. As with the aggregated metric, we observe that most of the classes have higher scores for the predictions from Pri3D than those from ENet. The second set of results shows the metrics for 3D semantic segmentation after fusing ENet or Pri3D logits with SoftGroup. We observe that ENet-fused SoftGroup outperforms Pri3D-fused SoftGroup across all classes, except for the classes of bed, bookshelf, picture, and shower curtain. This contradicts our initial hypothesis that a subset of classes within the Pri3D-fused SoftGroup predictions would perform notably worse than the ENet-fused SoftGroup predictions, causing the overall metric for the Pri3D experiment to be lower than ENet.

We performed an additional experiment fusing 2D ground-truth semantic labels with SoftGroup to gain further insights into the impact that 2D semantic segmentation quality has on post-fusion 3D instance segmentation performance. The results are also presented in Table 5.3. We observed significant improvements across all classes. Interestingly, although the back-projected ground-truth labels performed worse pre-fusion (i.e., 61.9 mIoU) compared to the baseline SoftGroup (i.e., 72.0 mIoU), we observed a significant post-fusion improvement (i.e., 87.9 mIoU). This indicates that the network learned additional relationships from the augmented input and did not solely perform an identity function to replicate the predictions from the back-projected ground-truth labels. This finding also suggests that fusing outputs from higher-performing 2D semantic segmentation models would greatly enhance 3D instance segmentation post-fusion.

We revise our initial hypothesis and propose that the relationship between the quality of 2D semantic segmentation and the post-fusion performance of 3D instance segmentation is generally proportional. However, when the 2D semantic segmentation quality is slightly below the threshold that would begin enhancing instance segmentation post-fusion, there can be a slight degradation in post-fusion performance. One potential explanation for this phenomenon could be attributed to our early-fusion ar-

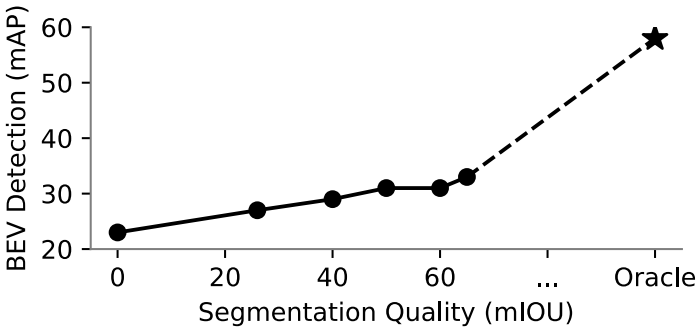


Figure 5.2: From PointPainting [42]. The PointPainting dependency on segmentation quality. The Painted PointPillars object detection performance, as measured by mean average precision (mAP) on the nuScenes [6] validation split, is compared with respect to the quality of the semantic segmentation network used in the painting step, as measured by mean intersection over union (mIoU). The oracle uses the 3D bounding boxes as semantic segmentation.

chitecture, wherein the entire model (including the encoder) remains unfrozen during training. When the entire model is unfrozen during training, many parameters require tuning. Consequently, in situations where the 2D and 3D models exhibit comparable levels of performance, the model may face challenges in discerning which predictions to favour with higher confidence and a larger number of adjustable parameters could exacerbate this problem. Figure 5.2 from PointPainting [42] demonstrates a similar trend where 3D object detection performance obtained from using early fusion generally increases as 2D semantic segmentation quality improves except for the middle region of the graph. In the middle region, the authors of PointPainting also observe that higher segmentation quality can slightly decrease performance post-fusion.

Observation 2: Importance of Point-to-Pixel Alignment

We began our investigation of the accuracy of point-to-pixel alignment by visualizing the results of the correspondences found by the Multi-Frame PointPainting algorithm (described in Algorithm 1) with a skip strategy of ten frames (shown in Figure 5.3). Pixel correspondences with a correct semantic label are shown in light grey. Correspondences with an incorrect semantic label are shown in black. Points that could not find a pixel correspondence are shown in dark grey. In this visualization, 91.2% of points are matched to pixel correspondences with a correct semantic label, 6.0% are matched to pixel correspondences with an incorrect semantic label, and 2.8% could not find a matching pixel correspondence. We observe that a majority of points with incorrect semantic labels are situated along object boundaries, which would

have a significant adverse effect on segmentation accuracy after fusion. Our proposed Guided Multi-Frame PointPainting algorithm (described in Algorithm 2) with a skipping strategy of three frames significantly reduces the number of incorrect semantic labels. In this specific scene, using Guided Multi-Frame PointPainting with a skipping strategy of three frames yielded the following results: 97.2% of pixel correspondences exhibited the correct semantic label (an increase of 6.0%); 2.2% of correspondences had an incorrect semantic label (a decrease of 3.8%); and only 0.6% of points failed to find a pixel correspondence (a decrease of 2.2%).

Figure 5.3: Visualization of the correspondences found using Multi-Frame PointPainting with a skipping strategy of ten frames. 91.2% of points are matched to pixel correspondences with a correct semantic label (shown in light grey); 6.0% are matched to pixel correspondences with an incorrect semantic label (shown in black); and 2.8% could not find a matching pixel correspondence (shown in dark grey). Most of the points with incorrect semantic labels are at object boundaries which can have an adverse effect on segmentation. By using Guided Multi-Frame PointPainting with a skipping strategy of ten frames, 94.8% of points have correct pixel correspondences, 2.4% have incorrect correspondences and 2.8% could not find a correspondence. By using Guided Multi-Frame PointPainting with a skipping strategy of three frames, 97.2% of points have correct pixel correspondences, 2.2% have incorrect correspondences and 0.6% could not find a correspondence.

Projection Source	$mIoU_{3D}$	Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Desk	Curtain	Refrigerator	S. Curtain	Toilet	Sink	Bathtub	Other Furn.
ENet Preds (G3)	32.1	51.0	77.4	25.0	43.7	42.7	37.1	38.8	23.3	18.3	32.2	10.6	30.1	31.0	22.3	20.3	22.5	26.6	26.5	48.2	14.0
Pri3D Preds (G3)	43.7	58.4	77.0	34.5	56.0	57.3	50.4	48.4	30.3	22.1	46.1	7.5	38.7	38.6	26.7	29.5	30.2	71.9	45.9	69.5	34.6
Sem. 2D Labels (G3)	67.7	70.7	87.9	61.5	72.8	78.7	74.8	79.2	53.2	44.4	61.3	33.6	70.7	70.9	44.6	63.2	65.4	85.7	72.4	89.8	73.6
Sem. Seg. Method	$mIoU_{3D}$																				
SoftGroup (Repr) [43]	72.0	85.3	95.0	64.7	80.1	89.2	78.7	74.6	63.9	65.4	76.9	31.1	65.3	67.6	72.4	59.9	71.1	91.8	63.9	85.1	59.0
+ ENet Logits (G3)	70.1	83.5	94.9	61.3	79.3	88.4	77.0	70.5	59.0	64.1	76.6	29.7	62.9	63.5	**76.3**	53.8	**71.5**	**92.3**	61.6	82.9	53.3
+ Pri3D Logits (G3)	69.5	83.6	94.7	62.4	**80.5**	88.6	77.7	74.3	63.6	61.1	76.3	24.9	55.6	63.7	66.9	53.6	67.2	91.3	62.9	82.1	57.9
+ Sem. 2D Labels (G3)	**90.9**	**94.2**	**97.6**	**91.9**	**94.0**	**96.6**	**95.6**	**94.8**	**84.3**	**86.7**	**94.2**	**67.1**	**87.7**	**92.5**	**88.3**	**89.8**	**94.3**	**96.5**	**85.5**	**94.5**	**92.3**
Inst. Seg. Method	mAP_{50}																				
SoftGroup (Repr) [43]	67.3	-	-	60.0	77.2	84.6	69.5	75.9	53.8	52.7	67.7	57.4	35.5	58.9	52.8	74.7	73.8	100.0	68.6	86.6	62.1
+ ENet Logits (G3)	66.3	-	-	**61.5**	76.1	84.1	61.6	**77.7**	48.9	52.6	58.9	52.6	**40.4**	**65.5**	**57.4**	69.6	73.2	98.3	67.9	**89.6**	58.2
+ Pri3D Logits (G3)	64.9	-	-	**62.0**	74.8	**84.4**	**71.1**	72.7	51.9	52.5	66.1	44.2	**36.0**	52.6	**62.1**	54.4	**76.8**	100.0	64.0	86.2	56.0
+ Sem. 2D Labels (G3)	**84.2**	-	-	**80.0**	**93.8**	**88.3**	**87.3**	**95.7**	**64.4**	**76.0**	**76.8**	**65.4**	**73.6**	**85.0**	**76.0**	**98.2**	**96.4**	**100.0**	**79.6**	**99.5**	**80.2**

Table 5.4: All results are evaluated on the ScanNetv2 validation set using early fusion; point-to-pixel matching is performed using Guided Multi-Frame PointPainting with a skipping strategy of three frames which is denoted by G3. The first set of results is computed by back-projecting 2D semantic segmentation predictions or ground-truth labels into 3D and directly computing 3D semantic segmentation metrics. The second set of experiments computes the 3D semantic segmentation metrics for the reproduced baseline and the experiments which fuse the 2D logits or labels into the input point cloud. The last set of results computes the corresponding 3D instance segmentation metrics for the same experiments in the second set. Bolded values represent metrics which are higher than the baseline.

Method	Input Conv	Backbone	mAP_{50}
SoftGroup (Repr) [43]	F	F	67.3
SoftGroup	U	F	68.0
SoftGroup	U	U	65.5
+ ENet Logits	F	F	67.0
+ ENet Logits (G3)	F	F	67.3
+ Pri3D Logits	F	F	66.5
+ Pri3D Logits (G3)	F	F	67.1
+ Sem. 2D Labels	F	F	70.0
+ Sem. 2D Labels (G3)	F	F	71.2

Table 5.5: Mid-fusion 3D instance segmentation results evaluated on ScanNet v2 validation set. The mid-fusion method concatenates the 2D and 3D features after the 3D encoder backbone and processes the concatenated 2D and 3D features with a feedforward layer. G3 denotes point-pixel matching using Guided Multi-Frame PointPainting while experiments with no suffix use Multi-Frame PointPainting. Repr is our reproduced SoftGroup trained from scratch using two V100 GPUs. F stands for frozen weights initialized with HAIS [14] pre-trained weights and U stands for unfrozen weights initialized with HAIS pre-trained weights.

We demonstrate that Guided Multi-Frame PointPainting considerably enhances the alignment between points and pixels. This improvement is evident when comparing the first set of experiments presented in Table 5.4 with the corresponding experiments in Table 5.3. We also observe that the increase in alignment accuracy has a positive impact on the fusion experiments, resulting in improvements across all the instance segmentation results and most of the semantic segmentation results. The best-performing ENet-fused SoftGroup experiment surpasses the baseline that trains with a fully unfrozen encoder but even with the improvements, it still does not outperform the baseline with a fully frozen encoder. This suggests that the frozen pre-trained weights of the SoftGroup encoder obtained from the pre-trained HAIS perform very well on the ScanNet dataset and should be retained if possible. In our early-fusion experiments, we were unable to keep the encoder frozen because the addition of image-based fusion vectors necessitates changes to the input layer size of the encoder. However, our mid-fusion approach appends the image-based fusion vectors after the encoder, which allows us to keep the encoder weights frozen during training. We discuss our mid-fusion experiments in the following section.

5.3 Results from the Mid-Fusion Experiments

One notable observation from Table 5.1 is that training SoftGroup with a fully frozen encoder outperforms training with a fully unfrozen encoder. This can be attributed to the fact that the HAIS pre-trained weights are already highly optimized to produce latent features beneficial for the instance segmentation task. However, in the case of early-fusion, where an image-based fusion vector is appended to each point in the input point cloud, the encoder needs to be unfrozen to process the additional dimensions introduced by the fusion. To fully leverage the features generated by the encoder without unfreezing the pre-trained weights, we explored the mid-fusion approach by fusing the image-based fusion vectors at the output of the 3D encoder. As mentioned in Chapter 4, this is achieved by concatenating the image-based fusion vector with the pre-fusion 3D semantic segmentation logits produced by the frozen 3D encoder. Then, a feed-forward layer is used to process the concatenated features into the post-fusion 3D semantic segmentation logits.

The instance segmentation metrics for the mid-fusion experiments are presented in Table 5.5. Similar to early fusion, we observe that employing Guided Multi-Frame PointPainting (denoted as G3) outperforms regular Multi-Frame PointPainting (denoted without a suffix). Additionally, the mid-fusion experiments with ENet and Pri3D yield higher results than the early-fusion experiments. The best-performing experiment using mid-fusion matches the performance of the frozen baseline of SoftGroup.

Sem. Seg. Method	$mIoU_{3D}$	Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Desk	Curtain	Refrigerator	S. Curtain	Toilet	Sink	Bathtub	Other Furn.
SoftGroup (Repr) [43]	72.0	85.3	95.0	64.7	80.1	89.2	78.7	74.6	63.9	65.4	76.9	31.1	65.3	67.6	72.4	59.9	71.1	91.8	63.9	85.1	59.0
+ ENet Logits (G3)	**72.1**	85.3	95.0	**64.8**	**80.4**	89.2	**79.6**	73.9	**64.4**	**66.1**	**78.0**	**31.7**	**65.5**	66.8	72.4	58.5	70.3	**92.1**	63.8	**85.2**	**59.4**
+ Pri3D Logits (G3)	**72.6**	85.2	94.9	**66.0**	80.3	**89.7**	**80.7**	**74.7**	64.2	65.6	**78.7**	**31.8**	64.5	**68.0**	71.9	**60.4**	**72.3**	**93.1**	**64.8**	84.7	**60.2**
+ Sem. 2D Labels (G3)	**82.8**	**89.5**	**96.9**	**79.4**	**83.0**	**93.6**	**90.4**	**88.5**	**73.8**	**74.7**	**83.1**	**45.5**	**86.8**	**85.2**	**80.6**	**75.8**	**80.2**	**96.6**	**81.7**	**93.0**	**76.9**
Inst. Seg. Method	mAP_{50}																				
SoftGroup (Repr) [43]	67.3	-	-	60.0	77.2	84.6	69.5	75.9	53.8	52.7	67.7	57.4	35.5	58.9	52.8	74.7	73.8	100.0	68.6	86.6	62.1
+ ENet Logits (G3)	67.3	-	-	**62.6**	74.4	83.8	68.4	**77.5**	52.5	**52.8**	63.9	**57.6**	**39.4**	56.4	**54.1**	74.1	**75.3**	100.0	**70.0**	86.6	**62.5**
+ Pri3D Logits (G3)	67.1	-	-	**63.0**	73.0	83.9	67.7	**76.2**	51.7	**54.0**	65.9	**59.2**	**37.1**	56.0	51.9	74.7	73.4	100.0	**71.0**	**86.7**	**62.2**
+ Sem. 2D Labels (G3)	**71.2**	-	-	**67.5**	76.9	84.3	69.5	**80.2**	53.5	**57.0**	**68.8**	**60.2**	**46.1**	**61.8**	52.6	**79.2**	**92.4**	100.0	**77.0**	**87.8**	**67.2**

Table 5.6: Class-wise mid-fusion results on semantic and instance segmentation. All results are evaluated on the ScanNetv2 validation set using mid-fusion; point-to-pixel matching is performed using Guided Multi-Frame PointPainting with a skip strategy of three frames which is denoted by G3. The first set of experiments computes the 3D semantic segmentation metrics for the reproduced baseline and the mid-fusion experiments. The second set of results computes the corresponding 3D instance segmentation metrics for the same experiments in the first set. Bolded values are metrics that are higher than the baseline.

Even though there was no enhancement in instance segmentation performance, we remained curious about the potential for improved semantic segmentation performance through mid-fusion. Initially, we conducted a statistical analysis on the validation set to estimate the maximum performance enhancement for 3D semantic segmentation using the mid-fusion approach. This statistical analysis compared the accuracy of the 3D semantic segmentation predictions produced by the frozen 3D encoder with those of the 2D semantic segmentation predictions back-projected into 3D space. We found that for 2D predictions extracted from ENet:

- 46.1% of points have the correct semantic class from both the back-projected 2D and the 3D predictions.
- 21.2% of points only have the correct semantic class from the 3D predictions.
- 1.9% of points only have the correct semantic class from the back-projected 2D predictions.
- 30.7% of points have an incorrect semantic class from both back-projected 2D and 3D predictions.

We found that for the 2D predictions extracted from Pri3D:

- 52.5% of points have the correct semantic class from both the back-projected 2D and the 3D predictions.
- 14.9% of points only have the correct semantic class from the 3D predictions.
- 2.1% of points only have the correct semantic class from the back-projected 2D predictions.
- 30.5% of points have an incorrect semantic class from both back-projected 2D and 3D predictions.

Therefore, if the network was able to retain all of the correct predictions from the 3D encoder but also accept the correct 2D predictions when the 3D predictions are incorrect, then the maximum semantic segmentation performance boost from mid-fusion would be approximately 2%. We present the class-wise 3D semantic and instance segmentation metrics for the top-performing mid-fusion experiments in Table 5.6. Based on these results, while there is no improvement in instance segmentation performance through mid-fusion, semantic segmentation performance exhibits enhancements of +0.1 mIoU with ENet and +0.6 with Pri3D.

Input	Encoder	mAP_{50}
XYZRGB [43]	F	**67.3**
XYZ	F	46.2
RGB	F	25.3
Channel Masking	F	66.9
Point Masking	F	66.3
XYZRGB	U	65.5
XYZ	U	66.1
RGB	U	27.8
Channel Masking	U	**66.2**
Point Masking	U	65.5

Table 5.7: Masking augmentation comparison with 3D instance segmentation results evaluated on ScanNet v2 validation set using the SoftGroup model. F stands for frozen weights initialized with HAIS [14] pre-trained weights and U stands for unfrozen weights initialized with HAIS pre-trained weights. Channel masking refers to a 50% probability of masking either XYZ or RGB channels with equal probability (i.e. 50% XYZRGB, 25% XYZ, 25% RGB). Point masking refers to a 20% probability of masking the entire point (i.e. 80% XYZRGB, 20% 000000).

5.4 Ablation Study from Masking Augmentation Experiments

In this final section, we conduct an ablation study to assess the importance of geometry information compared to colour information for 3D instance segmentation performance. In Table 5.7, we present a comparison based on training SoftGroup with two input channel configurations: one utilizing exclusively the XYZ channels (with RGB values set to zero) and the other using only the RGB channels (with XYZ values set to zero). During the data preprocessing stage, RGB values undergo a transformation to bring them to a scale spanning from -1 to 1, while the XYZ points are normalized by the mean of all the points within the point cloud. Consequently, when the RGB values are set to zero, this action is analogous to positioning them at the midpoint of their range, while setting the XYZ points to zero equates to setting them to the mean of the points. This analysis allows us to determine the contribution of each component towards downstream performance.

We observe that using only XYZ channels contributes nearly double the performance (i.e., 46.2 mAP_{50}) compared to using only RGB channels (i.e., 25.3 mAP_{50}) when the encoder is frozen. Surprisingly, when the 3D encoder is unfrozen, using only XYZ channels (i.e., 66.1 mAP_{50}) surpasses using both XYZ and RGB channels

together (i.e. 65.5 mAP_{50}). This implies that in the context of the 3D instance segmentation task, raw RGB values might be viewed as less reliable and more susceptible to noise compared to the geometric information derived from the XYZ channels. The susceptibility to variations in RGB values under different lighting conditions could be a contributing factor. Relying exclusively on geometric information could offer a more distinguishing input for segmenting distinct objects, particularly in cases where the shapes of object classes exhibit significant differences, such as distinguishing between a table and a chair. Conversely, colour information may prove advantageous in segmenting between object classes that possess similar shapes but have different colours, such as discriminating between a window and a picture frame.

We also conducted experiments involving two types of input masking to assess whether different masking augmentations could enhance the robustness and generalizability of the SoftGroup model. These results are also presented in Table 5.7. In channel masking, for each input point cloud, we randomly assigned a 50% chance of using the full XYZRGB channels, a 25% chance of only using XYZ channels, and a 25% chance of only using RGB values. In point masking, we introduced a probability of masking 20% of the points in the point cloud with zero values, while retaining the full XYZRGB channels for the remaining 80% of points. The motivation behind channel masking and point masking was to expose the model to examples with reduced feature dimensions or fewer points, thereby encouraging the model to maximize feature extraction from the given input data. We observed that channel masking outperformed point masking. However, although it outperformed all other ablation experiments when the encoder was unfrozen, it did not surpass the performance of the baseline SoftGroup model when the encoder was frozen.

5.5 Summary

This chapter described our experimental settings and the findings from our early-fusion, mid-fusion and masking augmentation experiments. We analyzed our early-fusion experiments, which fused different representations of the image-based fusion vectors into SoftGroup, and found that fusing with soft information (i.e., softmax scores) provided higher post-fusion performance than fusing with hard information (i.e. one-hot encoded predictions). From these early-fusion experiments, we observed that initializing the weights of the 3D encoder with pre-trained weights provided much higher performance compared to randomly initialized weights. We also observed that fusing with non-model-based inputs such as RGB and RGB-D values can improve

performance compared to the unfrozen SoftGroup baseline. In particular, we found that the additional depth value from RGB-D images provided a significant boost in performance compared to only using RGB values. Subsequently, we explored the importance of 2D semantic segmentation quality on post-fusion 3D instance segmentation performance. In general, we found that, except for a middle region, post-fusion 3D instance segmentation performance improves as 2D semantic segmentation quality increases. Then, we discussed the importance of point-to-pixel alignment accuracy on post-fusion performance. We described how our Guided Multi-Frame PointPainting algorithm significantly improved the alignment accuracy of correspondences and showed that the increased accuracy resulted in improved post-fusion performance.

We observed that training SoftGroup with a fully frozen encoder led to superior results than training it with a fully unfrozen encoder. This observation motivated us to explore our mid-fusion approach, which allowed the model to preserve the frozen weights of the encoder, unlike the early-fusion approach. The mid-fusion experiments consistently outperformed their early-fusion counterparts. Notably, among the mid-fusion experiments, Pri3D-fused SoftGroup achieved the highest 3D instance segmentation performance. However, it still only matched the 3D instance segmentation performance of the baseline model equipped with a fully frozen encoder. Nonetheless, our investigation revealed that Pri3D-fused SoftGroup, employing the mid-fusion approach, improved post-fusion 3D semantic segmentation and surpassed the 3D semantic segmentation performance of the baseline model with a fully frozen encoder by 0.6 mIoU. We also conducted a statistical analysis to estimate the upper limit of performance enhancement achievable in post-fusion semantic segmentation through the mid-fusion approach. This statistic can serve as a valuable reference for future research employing our approach by helping to assess the potential effectiveness of the fusion method.

We also conducted an ablation study by selectively masking either the XYZ or RGB input channels to assess the impact of each set of channels on performance. We found that the XYZ channels contributed significantly more to the 3D instance segmentation performance compared to the RGB channels. This is likely attributed to the sensitivity of RGB values to varying lighting conditions, potentially introducing noise that adversely affects the performance of the model. We proposed that XYZ data is important for objects with different shapes, while RGB data is valuable for objects with similar shapes but different colours. In the final chapter, we will discuss our contributions, the limitations of our approach and the potential improvements.

Chapter 6

Conclusion

Reconstructed point clouds are valuable for encoding geometric information without occlusions, but they tend to be sparse and expensive to annotate. Images are easier to label and provide dense colour information, but they suffer from incomplete single-view information. These characteristics motivated us to propose a method for fusing image-based information from indoor scenes into a 3D instance segmentation model to further enhance the performance of the model.

6.1 Contributions

Throughout our study, we explored two key aspects: where to fuse the image-based information—its location—and what information to fuse—its type. For fusion location, we investigated the effects of early fusion at the input of the 3D encoder and mid-fusion at the output of the 3D encoder. In terms of types, we considered windowed RGB/RGB-D values, ImageNet features and outputs from 2D semantic segmentation models such as features, logits, scores, and one-hot encoded predictions. Furthermore, we conducted experiments utilizing Multi-Frame PointPainting and enhanced the alignment accuracy of point-to-pixel correspondences by introducing Guided Multi-Frame PointPainting. Finally, we compared the significance of XYZ values versus RGB values for the 3D instance segmentation task. By exploring these various aspects, we aimed to leverage the strengths of both point clouds and images and investigate their fusion for improved performance in 3D instance segmentation.

The primary contribution of this work lies in the application of sequential fusion for 3D instance segmentation specifically tailored for indoor scenes. Previous works have primarily concentrated on simpler tasks such as object detection or semantic segmentation. Moreover, prior research has mainly focused on outdoor datasets, which

often exhibit one-to-one or few-to-one correspondences between pixels and points. We introduced Guided Multi-Frame PointPainting, a novel approach that enhances point-to-pixel matching, particularly in scenarios involving many-to-one correspondences, a situation more prevalent with 3D indoor datasets than outdoor ones.

Our study yielded several key findings. First, we discovered that soft information, such as softmax scores, provided stronger performance in post-fusion results than hard information, such as one-hot encoded predictions. This is because softmax scores with higher entropy contain more information for the model to interpret. Additionally, images are susceptible to lighting conditions, so having a measure of confidence in the form of softmax scores was essential to maximize performance.

Second, we found that the choice of fusion location had an impact on performance. Mid-fusion yielded better results than early fusion and, even though instance segmentation did not improve, semantic segmentation showed improvements after using the mid-fusion approach. However, early fusion performed better when fusing ground-truth labels, so it remains inconclusive which fusion location is superior when more advanced 2D semantic segmentation models become available in the future. We also provided a method to compute an estimated maximum performance boost using mid-fusion that can be useful for future work to determine the potential gains of this fusion method.

Third, we determined that geometry data contributed significantly more to 3D instance segmentation performance compared to colour data. Our hypothesis is that geometry information serves as a stronger discriminator than colour information when object classes have distinct shapes, which is often the case. However, colour information can be valuable in specific scenarios where object classes share the same shape. Nonetheless, such cases are generally uncommon in the ScanNet dataset [16], so improvements from image-based information were limited.

Lastly, we established that high-quality 2D semantic segmentation and accurate point-to-pixel alignment are crucial factors for the success of this sequential fusion method. Our results demonstrated that post-fusion results improved with more precise alignment. Furthermore, while high-quality 2D semantic segmentation (i.e., ground truth labels) led to significant improvements in post-fusion results, having a stronger 2D semantic segmentation model (i.e., Pri3D) did not outperform a weaker 2D semantic model (i.e., ENet). We propose the hypothesis that if the 2D semantic segmentation model has a comparable level of performance to the 3D instance segmentation model, the fusion-based training process could face difficulties in determining which model predictions to prioritize.

6.2 Limitations and Potential Improvements

Previous research, such as PointPainting [42], demonstrated successful enhancements to 3D object detection models by using sequential fusion evaluated on outdoor datasets. Image-based fusion with 3D models may work better for outdoor datasets because outdoor datasets tend to consist of point cloud frames that are sparse, so they can fully benefit from the dense image features. In contrast, indoor datasets tend to use reconstructed point cloud maps much denser than point cloud frames and so the performance enhancements from fusing with dense image features may be limited. This hypothesis is supported by the fact that PointPainting [42] improved most for classes that often have few points on them.

The sequential fusion method proposed in this thesis was also limited by the performance of the 2D semantic segmentation model relative to the performance of the 3D instance segmentation model. Despite Pri3D being one of the state-of-the-art models for 2D semantic segmentation on the ScanNet benchmark, it did not provide significant post-fusion improvements. In contrast, fusing with ground-truth labels yielded substantial improvements, highlighting the potential for future higher-performing 2D semantic segmentation models to enhance post-fusion results using this method.

Lastly, the accuracy of point-to-pixel alignment also limited the effectiveness of this fusion approach. In future work, we suggest investigating joint training of a model on 2D and 3D data for 3D instance segmentation in indoor scenes. Although joint fusion requires a more complex optimization process, joint fusion would allow the network to predict an alignment offset vector. The predicted alignment offset vector could be used to improve the point-to-pixel alignment, thereby improving downstream fusion performance. Nonetheless, given the many-to-one correspondences between pixels and points in indoor datasets, joint fusion is likely to incur higher computational costs than sequential fusion. Consequently, computational considerations must be factored in when pursuing this approach.

Bibliography

[1] Iro Armeni et al. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1534–1543.

[2] Iro Armeni et al. "Joint 2D-3D-Semantic Data for Indoor Scene Understanding". In: *arXiv preprint arXiv:1702.01105* (2017).

[3] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

[4] Jens Behley et al. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". In: *International Conference on Computer Vision*. 2019, pp. 9297–9307.

[5] Serge Beucher. "Use of Watersheds in Contour Detection". In: *International Workshop on Image Processing*. 1979, pp. 17–21.

[6] Holger Caesar et al. "nuScenes: A Multimodal Dataset for Autonomous Driving". In: *Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11618–11628.

[7] Jun Cen et al. "SAD: Segment Any RGBD". In: *arXiv preprint arXiv:2305.14207* (2023).

[8] Soravit Changpinyo et al. "Conceptual 12M: Pushing Web-Scale Image-Text Pre-Training To Recognize Long-Tail Visual Concepts". In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3558–3568.

[9] R. Qi Charles et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Conference on Computer Vision and Pattern Recognition*. 2017, pp. 77–85.

[10] Liang-Chieh Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 834–848.

[11] Liang-Chieh Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *European Conference on Computer Vision*. 2018, pp. 801–818.

[12] Liang-Chieh Chen et al. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[13] Liang-Chieh Chen et al. “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: *International Conference on Learning Representations*. 2015.

[14] Shaoyu Chen et al. “Hierarchical Aggregation for 3D Instance Segmentation”. In: *International Conference on Computer Vision*. 2021, pp. 15467–15476.

[15] Özgün Çiçek et al. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”. In: *Medical Image Computing and Computer-Assisted Intervention*. 2016, pp. 424–432.

[16] Angela Dai et al. “ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes”. In: *Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2432–2443.

[17] Mihir Garmella and Prathik Naidu. “Beyond the Pixel Plane: Sensing and Learning in 3D”. In: *The Gradient* (2018).

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361.

[19] Roger Grosse. *University of Toronto CSC321 Lectures*. 2017.

[20] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the Knowledge in a Neural Network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[22] Ji Hou, Angela Dai, and Matthias Nießner. “3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4421–4430.

[23] Ji Hou et al. “Pri3D: Can 3D Priors Help 2D Representation Learning?” In: *International Conference on Computer Vision*. 2021, pp. 5693–5702.

[24] Tengteng Huang et al. “EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection”. In: *European Conference on Computer Vision*. 2020, pp. 35–52.

[25] Li Jiang et al. “PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation”. In: *Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4866–4875.

[26] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1503.02531* (2017).

[27] Alexander Kirillov et al. “Segment Anything”. In: *arXiv preprint arXiv:2304.02643* (2023).

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012.

[29] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *International Conference on Computer Vision*. 2017, pp. 2980–2988.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[31] J. MacQueen. “Some Methods for Classification and Analysis of Multivariate Observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Vol. 5.1. 1967, pp. 281–298.

[32] Anas Mahmoud, Jordan S. K. Hu, and Steven L. Waslander. “Dense Voxel Fusion for 3D Object Detection”. In: *Winter Conference on Applications of Computer Vision*. 2023, pp. 663–672.

[33] Anas Mahmoud and Steven L. Waslander. “Sequential Fusion via Bounding Box and Motion PointPainting for 3D Objection Detection”. In: *Conference on Robots and Vision*. 2021, pp. 9–16.

[34] Edwin G. Ng et al. “Understanding Guided Image Captioning Performance across Domains”. In: *Conference on Computational Natural Language Learning*. 2021, pp. 183–193.

[35] Adam Paszke et al. “ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation”. In: *arXiv preprint arXiv:1606.02147* (2016).

[36] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Conference on Neural Information Processing Systems*. Vol. 32. 2019.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *International Conference on Medical Image Computing and Computer Assisted Intervention*. Vol. 9351. 2015, pp. 234–241.

[38] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Representations by Back-Propagating Errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[39] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115 (2015), pp. 211–252.

[40] Piyush Sharma et al. “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”. In: *Association for Computational Linguistics*. 2018, pp. 2556–2565.

[41] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.

[42] Sourabh Vora et al. “PointPainting: Sequential Fusion for 3D Object Detection”. In: *Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4603–4611.

[43] Thang Vu et al. “SoftGroup for 3D Instance Segmentation on Point Clouds”. In: *Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2708–2717.

[44] Chunwei Wang et al. “PointAugmenting: Cross-Modal Augmentation for 3D Object Detection”. In: *Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11789–11798.

[45] Fisher Yu and Vladlen Koltun. “Multi-Scale Context Aggregation by Dilated Convolutions”. In: *International Conference on Learning Representations*. 2016.

[46] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. “Road Extraction by Deep Residual U-Net”. In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), pp. 749–753.

[47] Lin Zhao et al. “LIF-Seg: LiDAR and Camera Image Fusion for 3D LiDAR Semantic Segmentation”. In: *IEEE Transactions on Multimedia* (2023).