FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multimodal Deep Implicit Representations for Autonomous Driving

Ricardo Jorge Cruz Fontão

Mestrado em Engenharia Informática e Computação

Supervisor: Jaime dos Santos Cardoso

External Supervisor: Ricardo Cerqueira

July 18, 2023

Multimodal Deep Implicit Representations for Autonomous Driving

Ricardo Jorge Cruz Fontão

Mestrado em Engenharia Informática e Computação

July 18, 2023

Resumo

Condução autónoma tem ganho cada vez mais importância nos últimos anos. Ela depende fortemente de representar corretamente o ambiente ao redor do veículo usando os dados obtidos pelos seus sensores. Um dos sensores mais comuns é o LiDAR, que produz uma nuvem de pontos do ambiente ao redor do carro.

O crescimento da área de *deep learning* é um grande impulsionador do desenvolvimento de veículos autónomos. Nos últimos anos, tem havido um aumento no desenvolvimento de métodos de *deep learning* para tarefas de percepção, com muitos deles a alcançar resultados de estado-da-arte.

Representações são a base de qualquer tarefa de percepção e essenciais para qualquer veículo autónomo. *Deep implicit representations* (DIRs) são uma família de métodos novos e promissores que procuram representar dados 3D usando uma *deep neural network*. Essa representação é geralmente alcançada tendo a rede neural aprender *signed distance functions* (SDFs) ou funções de ocupação. Neste momento, estas representações já alcançam resultados de ponta na tarefa *semantic scene completion* do dataset SemanticKITTI.

Esta dissertação explora como uma arquitetura que usa DIRs pode ser modificada para melhorar os seus resultados. Em primeiro lugar, explora como diferentes representações implicítas de saída afetam o desempenho do modelo. Em seguida, tenta adicionar um módulo de fusão sensorial numa tentativa de incorporar dados da câmara RGB na *pipeline*, já que este é um sensor comum em veículos autónomos.

Os resultados mostraram que a mudança da representação implícita aprendida de SDFs para uma função de ocupação melhorou o desempenho do modelo, melhorando o *Intersection over Union* (IoU) de reconstrução geométrica em 0,85% e o *mean Intersection over Union* (mIoU) de segmentação semântica em 0,36%. O módulo de *middle fusion* também forneceu resultados melhorados, mas mais modestos. O IoU de reconstrução geométrica melhorou em 0,56% e o mIoU de segmentação semântica em 0,16%.

Abstract

Autonomous driving has been gaining traction in recent years. It relies heavily on accurately representing the environment around the car using the data captured by the vehicle's sensors. One of the most common sensors is the LiDAR, which produces a point cloud of the surrounding environment, video cameras, and radars.

The growth of the field of deep learning is a big driving force behind the development of autonomous vehicles. The last few years have seen a surge in the development of deep learning methods for perception tasks, with many of them achieving state-of-the-art results.

Representations are the foundation of any perception task and essential for any autonomous vehicle. Deep Implicit Representations (DIRs) are a novel and promising family of methods that seek to represent 3D data using a deep neural network. This representation is usually achieved by having the neural network learn signed distance functions (SDFs) or occupancy functions. These representations already achieve state-of-the-art results in the Semantic Scene Completion task of the SemanticKITTI dataset.

This dissertation explores how a novel architecture based on DIRs can be modified to improve its results. In a first approach, it explores how different output implicit representations affect the model's performance. Then, it tries to add a sensor fusion module in an attempt to incorporate RGB camera data into the pipeline since this is a commonly found sensor in self-driving vehicles as well. The impact of using feature fusion with DIRs will be measured.

The results showed that switching the learned implicit representation from SDFs to occupancy functions improved the model's performance, increasing the scene completion Intersection over Union (IoU) by 0.85% and semantic segmentation mean Intersection over Union (mIoU) by 0.36%. The middle fusion module also provided improved results but more modest ones. Scene completion IoU increased by 0.56% and semantic segmentation mIoU by 0.16%.

Acknowledgements

First, I would like to thank both my supervisors, Professor Jaime Cardoso and Ricardo Cerqueira, for the valuable guidance and support they provided me during the development of this work.

I want to thank my parents and my sister for all the support they gave me during my studies and life.

I want to thank the team at Bosch for all the help during my internship.

I want to thank Vitor for all the support and encouragement.

Finally, I want to thank all my friends who were part of this five year journey with me.

This work was supported by the European Structural and Investment Funds in the FEDER component through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 047264; Funding Reference: POCI-01-0247-FEDER-047264].

Ricardo Jorge Cruz Fontão

Contents

List of Figures

List of Tables

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
CNN	Convolutional Neural Network
DIR	Deep Implicit Representation
FoV	Field of View
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
mIoU	Mean Intersection over Union
NeRF	Neural Radiance Fields
ONet	Occupancy Network
RGB	Red Green Blue
SDF	Signed Distance Function
SSC	Semantic Scene Completion

Chapter 1

Introduction

1.1 Context

Autonomous driving has been a highly researched topic in the last few years. This is because it can potentially improve people's quality of life by reducing the number of accidents and the time spent in traffic. However, the development of autonomous vehicles is still in its infancy, with many challenges to overcome. One of the challenges presented is interpreting the environment, be it identifying surrounding objects, tracking the movement of other vehicles, or predicting the movement of pedestrians. This is because the environment is constantly changing, and autonomous vehicles need to be able to adapt to these changes. Thus, autonomous driving perception research branches into many different areas and tasks, such as object detection, tracking, lane estimation, ego-motion estimation, and semantic scene completion.

An autonomous vehicle must be equipped with sensors to capture information about the surrounding environment. The most common types of sensors are cameras, LiDAR sensors, and radars. Each one of these sensors has its advantages and disadvantages. For example, cameras can capture a large field of view, but they struggle to capture information about the distance to the objects. On the other hand, LiDAR sensors can capture information about the distance to an object, but they cannot capture information about its color. Radars can capture information about the distance to the objects, but they cannot capture information about the shape of the objects. Thus, the combination of these sensors can capture information about the environment in a more complete way, known as sensor fusion.

1.2 Motivation

At the moment, deep learning-based approaches achieve state-of-the-art results in most perception tasks for autonomous driving, and there is a lot of research being done in this area. Deep Implicit Representations are still a largely unexplored field in this context. This is mainly because they are not as popular as other deep learning approaches. However, they have the potential to provide a robust and accurate representation of the environment, which is a crucial aspect of autonomous

driving. This is because they can represent the environment in a compact way, which is essential for autonomous driving, where computational resources are limited. The Local-DIFs [35] architecture is a good example of the potential of implicit representations since it almost achieves state-of-the-art results on the Scene Completion part of SemanticKITTI's Semantic Scene Completion task. This is a task that involves reconstructing a dense semantically labeled representation of a 3D scene from an incomplete representation of that same scene. There is also a gap in the literature on sensor fusion methodologies in outdoor scenes. Thus, incorporating approaches from other architectures can yield good results and should provide a promising research path.

1.3 Objectives

This dissertation aims to study different approaches for 3D scene understanding for autonomous driving using deep implicit representations. It will be primarily focused on the recent Semantic Scene Segmentation (SSC) perception task. In the first step, we will directly compare two different implicit representations on the same architecture, which is something not present in the literature. This is expected to provide insights into how both different types of outputs influence the perception task at hand. After that, the impact of LiDAR-RGB sensor fusion on the SSC perception task will be studied. For this, we will modify an existing SSC network to accommodate sensor fusion approaches used in 3D object detection. Both the semantic segmentation and scene completion scores are expected to increase with these changes.

1.4 Document Structure

This chapter provided context, motivation, and objectives of the work presented in this dissertation.

Chapter 2 presents some core concepts needed to understand the work presented. In section 2.1, a brief overview of the sensors used in autonomous vehicles are presented, as well as the advantages and disadvantages of each one. Section 2.2 provides some background on deep learning. Then, section 2.3 provides a categorization of the sensor fusion techniques in the context of autonomous driving. Finally, Section 2.4 explains what implicit representations are and presents details about three different types of implicit representations.

Chapter 3 reviews the literature on learning deep implicit representations from point clouds in the context of autonomous driving, focusing primarily on deep implicit representations. The most used metrics are presented in section 3.1.1. Then, in section 3.1.2, the most used datasets for autonomous driving are presented. Section 3.3 provides a short review of point cloud data augmentation techniques. Section 3.2 presents the most used architectures for learning deep implicit representations from point clouds in the context of autonomous driving. Finally, section 2.3 presents some techniques for sensor fusion that will be used in this work.

Chapter 4 presents the modifications proposed to the model. Chapter 5 presents the results achieved by the various experiments carried out with the modifications. Finally, chapter 6 presents final thoughts about the work done and some proposed future work.

Chapter 2

Concepts

2.1 Sensors in autonomous vehicles

As previously mentioned, autonomous vehicles are equipped with different sensors, each with its strengths and weaknesses. This section will briefly describe the most common sensors used in autonomous vehicles.

2.1.1 LiDAR

LiDAR stands for Light Detection and Ranging and is a sensor that uses infrared beams or laser light to measure the distance to target objects. The laser emits a pulse of light, which is reflected by the object being measured. The time it takes for the light to be reflected back to the instrument is used to estimate the distance to the object. A 3D representation of the surrounding environment is stored as a point cloud.

LiDAR sensors can be categorized as mechanical LiDAR or solid-state LiDAR (SSL). Mechanical LiDARs use rotating lenses driven by an electric motor that directs the laser beams, thus capturing the desired field of view. The rotating lenses can achieve 360º, covering the vehicle's entire surroundings. Solid-state LiDARs eliminate the use of the rotating motor, reducing the chance of mechanical failures. Even though this type of LiDAR sensor is gaining popularity, their lower field of view (typically 120 degrees) makes their use in autonomous driving less desirable. [49]

LiDAR provides accurate depth and angle information. However, they are negatively affected by heavy rain, dust, or fog. [39]

2.1.2 Cameras

A camera works by detecting light emitted from the surroundings on an image plane through a camera lens. This process results in a 2D image of the surrounding environment. Camera systems in autonomous vehicles can use monocular cameras or stereo cameras. Conventional monocular RGB cameras cannot retrieve depth information, so they are often used in pairs to create a stereo camera system. Stereo cameras attempt to imitate the perception of depth found in animals by

capturing two images of the same scene. The sensors are placed separated by a baseline, which refers to the distance between them. [49]

Cameras can have very high resolution, providing much information about the surrounding environment. This makes them especially well-suited for modern deep-learning applications. On the other hand, they are negatively affected by poor illumination conditions (shadows, night), visibility is reduced by adverse weather conditions, and they can't provide direct distance measurements. [39]

2.1.3 Radar

Radar stands for Radio Detection and Ranging. This sensor uses the Doppler property of electromagnetic waves to determine the relative speed and position of detected obstacles. [49]

Radars are robust against adverse weather conditions, have a high range, and can provide information about the speed of the detected objects. On the other hand, their low resolution makes retrieving object shape information very difficult. This makes it not suitable for object recognition applications when used on its own. [39]

2.2 Deep Learning

Deep learning is a subfield of Machine Learning that uses neural networks to learn features from data. A common type of Neural Network is a Multilayer Perceptron (MLP). It was inspired by the human brain and nervous system and consists of an input layer, an output layer, and a varying number of hidden layers containing processing nodes called perceptrons. Each node is connected to all the nodes of the next layer, having an associated weight with that connection. At each perceptron, the input values of the incoming connections are multiplied by the connection weight and are all added. After the sum, an activation function is applied. The output of this function is then fed onto the next layer of perceptrons. The output layer's result is the "solution" to the problem. To train the neural network, backpropagation is used to compute the gradients of the loss with respect to the weights of the network based on training data. Gradient descent is used to update the weights. Neural Networks have many possible applications, such as computer vision and natural language processing (NLP), and are a crucial part of the efforts towards autonomous driving. [40]

Figure 2.1 shows an example of a neural network architecture.

2.2.1 Losses

The neural network's weights are typically initialized with random values. The goal of the training process is to find the best values for the weights that minimize a loss function. The loss function is a function that measures the difference between the predicted output and the expected output. An example of a common loss function used in deep learning is the Cross-Entropy (CE) function that is usually employed in classification problems. The CE function is defined as:

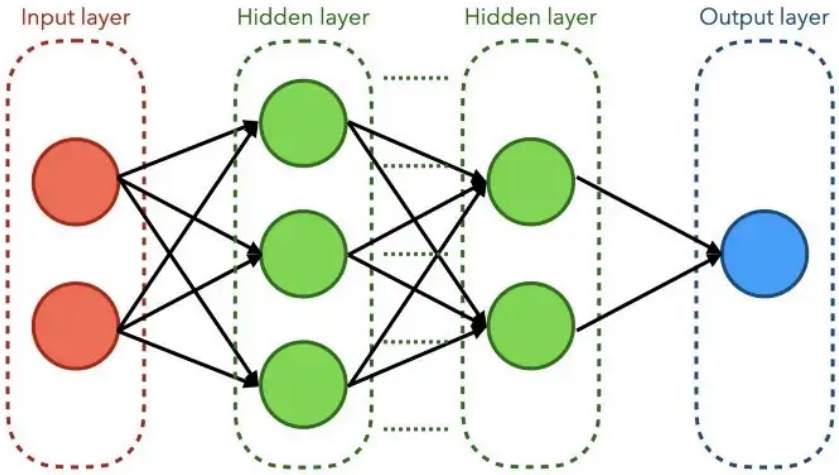


Figure 2.1: Example of an MLP Neural Network with two hidden layers. [10]

$$CE = -\sum_{i=1}^{N} y_i \log(p_i) \tag{2.1}$$

where N is the number of classes, y_i is the expected output, and p_i is the predicted output.

2.2.2 Regularisation

Overfitting the model to the training data is a common problem in machine learning. It means the model cannot generalize well to unseen data. In this section, some regularisation methods are presented.

Data augmentation is a technique used to increase the amount of training data by applying transformations to the original data. This helps prevent overfitting, as the model will be trained with a wider variety of data. In the case of learning with RGB images, the most common transformations are rotations, translations, flips, and color changes. Some examples of data augmentation in RGB images are shown in Fig. 2.2. Data augmentation can also be applied in point clouds, for example, by applying rotations, translations, and flips. More advanced methods will be presented in section 3.3.

Dropout is another regularization technique that aims to reduce overfitting. The method involves choosing nodes from the neural network and nullifying their activations so that they do not influence backpropagation. The dropout rate is the probability of a node being dropped out. A dropout rate of 0.5 means that, on average, half of the nodes will be dropped out. [39]

2.2.3 Convolutional networks

Convolutional Neural Networks (CNNs) are a type of neural network that is commonly used in computer vision tasks. They can be composed of convolutional layers, pooling layers, and fully connected layers. The convolutional layers are the most essential part of the network, as they are

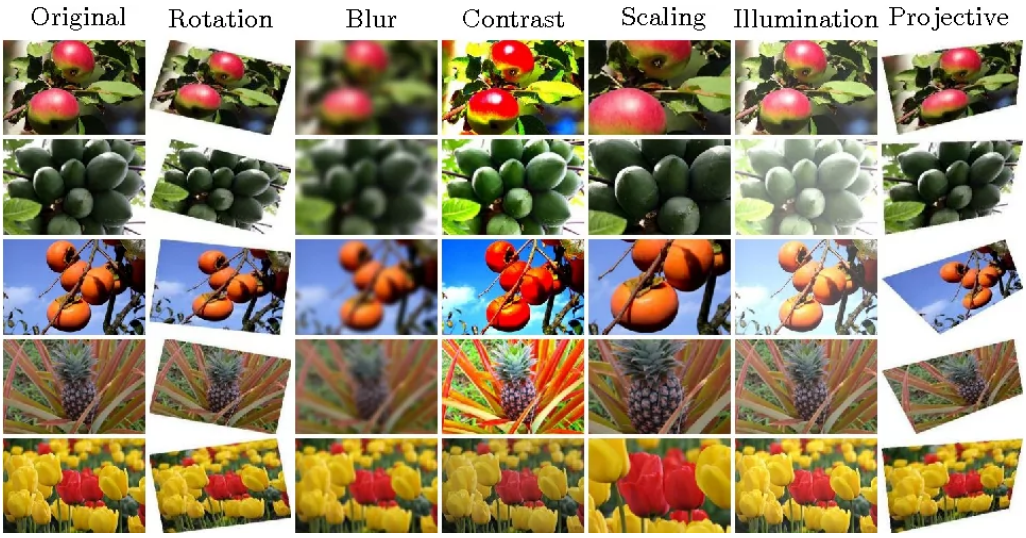


Figure 2.2: Examples of common data augmentation transformations in images. [6]

responsible for extracting features from the input data. The pooling layers are used to reduce the dimensionality of the data, while the fully connected layers are used to generate a result. The general idea of a CNN is that each layer extracts more complex features from the input data until the fully connected layers are reached, where a result is outputted. CNNs can be used, for example, for object detection, image classification, and semantic segmentation.

Figure 2.3 shows an example of a CNN architecture.

2.2.3.1 Convolutional layers

Convolutional layers are responsible for feature extraction, learning the feature representations of the input data. Inputs are convolved with the learned weights to compute the new feature map.

2.2.3.2 Pooling layers

Pooling layers are used to reduce the dimensionality of the data and thus achieve invariance to input distortions and translations. The most common pooling layers are Max Pooling and Average Pooling. While Average Pooling propagates the average of a receptive field, Max Pooling propagates its maximum value. A comparison of the two pooling layers is shown in Fig. 2.4.

2.2.3.3 Sparse Networks

Sparse CNNs are based on a similar mechanism to the one used by sparse matrices. In sparse matrices, most elements are zero, making it more efficient to store the non-zero values. Sparse CNNs assume each hidden variable has a *ground state* corresponding to not receiving any meaningful input. When the input tensor is sparse, only the values of the hidden variables that are not equal to the ground state need to be calculated. [16]. These sparse networks can be 2D [16], 3D [15] or even 4D [9]. The latter is the most used among the models presented in this work, the

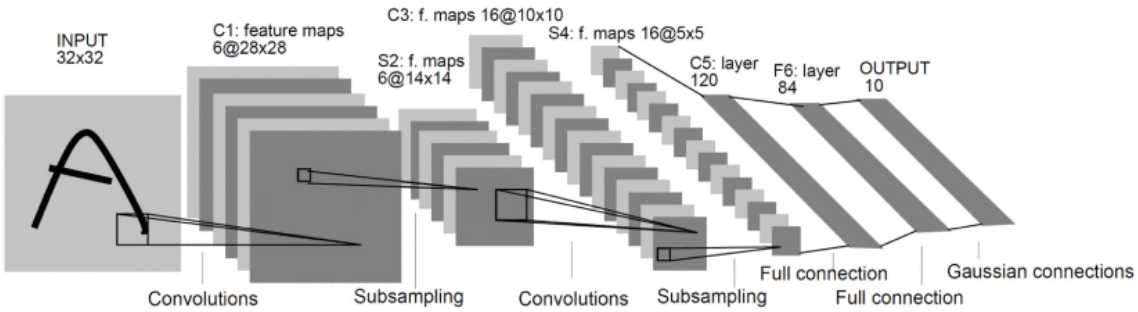


Figure 2.3: LeNet-5 architecture, a Convolutional Neural Network. [21]

Minkowski Engine library. It is an auto-differentiation library for high-dimensional sparse tensors and generalized sparse convolutions.

2.3 Sensor Fusion

To take advantage of the different sensors available in autonomous vehicles, the information from multiple sensors can be combined to improve the perception task's performance. This process is called sensor fusion. As previously stated, each sensor has its advantages and disadvantages, which can be used to complement each other. For example, cameras can provide high-resolution images but cannot detect objects in low-light conditions. On the other hand, LiDARs can detect objects in low light conditions but cannot provide high-resolution images. By combining the information from these sensors, the model performance can be increased.

According to Fayyad et al. [11], we can classify sensor fusion approaches based on their data input methods. Using this categorization, there is early fusion, middle or feature fusion, and late fusion.

Early fusion involves combining the sensor data at the raw data level. The method used can be simply concatenating both inputs, but sometimes the data might need to be projected onto a common 3D space if it is not spatially aligned [37]. Middle or feature fusion involves combining the sensor data at the feature level. This is done by extracting features from the sensor data and combining them midway through the network. Late fusion involves combining the decisions of multiple models into a single final decision.

An overview of the different sensor fusion approaches in the context of deep learning is shown in Fig. 2.5.

2.4 Implicit Representations

Implicit representations are a way to parametrize different types of signals. Conventional signal representations are usually discrete. For example, images are a discrete grid of pixels, or 3D shapes are represented as a grid of voxels or point clouds. Implicit representations, on the other

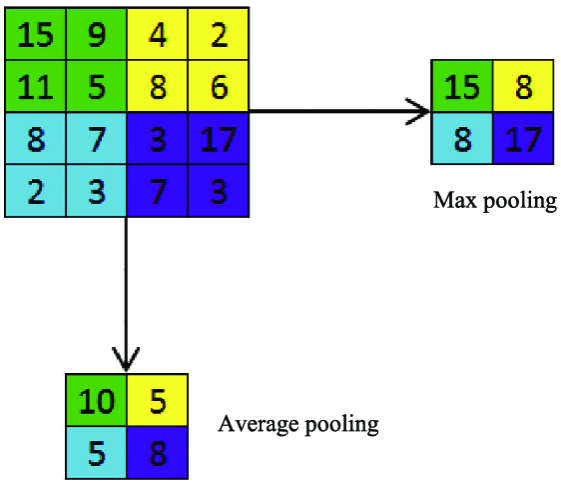


Figure 2.4: Comparison of Max Pooling and Average Pooling. [33]

hand, represent the signal as the solution to an implicit equation. In deep learning, the objective is to have a neural network learn this continuous function. [44]

2.4.1 Signed Distance Functions

Signed Distance Functions (SDF) are an extension of a Distance Function. These functions allow the implicit representation of objects by mapping a point in space to its distance to the closest surface. This formulation can be extended so that the sign of the distance value is based on whether the point is inside or outside the object's surface. This is called a Signed Distance Function (SDF).

A possible formulation for an SDF of an object $\Omega \subset \mathbb{R}^3$ is

$$f(x) = \begin{cases} d(x, \partial\Omega) & \text{if } x \in \Omega \setminus \partial\Omega \\ 0 & \text{if } x \in \partial\Omega \\ -d(x, \partial\Omega) & \text{if } x \in \mathbb{R}^3 \setminus \Omega \end{cases} \tag{2.2}$$

where $d(x, \partial\Omega)$ represents the smallest distance from x to $\partial\Omega$ (boundary of Ω) . In this formulation, points inside the object are assigned a positive distance and points outside a negative one, but this can be inverted in other formulations. [7]

2.4.2 Occupancy functions

Occupancy functions are used to represent objects implicitly. This is achieved by mapping each 3D point in the domain to 0 or 1 based on whether they are inside the object. A formulation for an occupancy function is

$$o : \mathbb{R}^3 \to \{0, 1\} \tag{2.3}$$

Occupancy functions were first introduced in Mescheder et al. [28].

2.4.3 Neural Radiance Fields

Neural Radiance Field (NeRF) models are novel view synthesis methods that leverage volume rendering with an implicit neural scene representation. They were first proposed in Mildenhall et al. [30]. A NeRF model represents a scene as a radiance field approximated by a deep neural network. The radiance field describes color and volume density for every point and view direction in the scene. This can be defined as:

$$F(\mathbf{x}, \theta, \phi) = (\mathbf{c}, \sigma), \tag{2.4}$$

where $\mathbf{x} = (x, y, z)$ is the in-scene coordinate, (θ, ϕ) represent the azimuthal and polar viewing angles, $\mathbf{c} = (r, g, b)$ represent colour and σ represents the volume density.

A comprehensive review of Neural Radiance Field models and their applications can be found in Gao et al. [12].

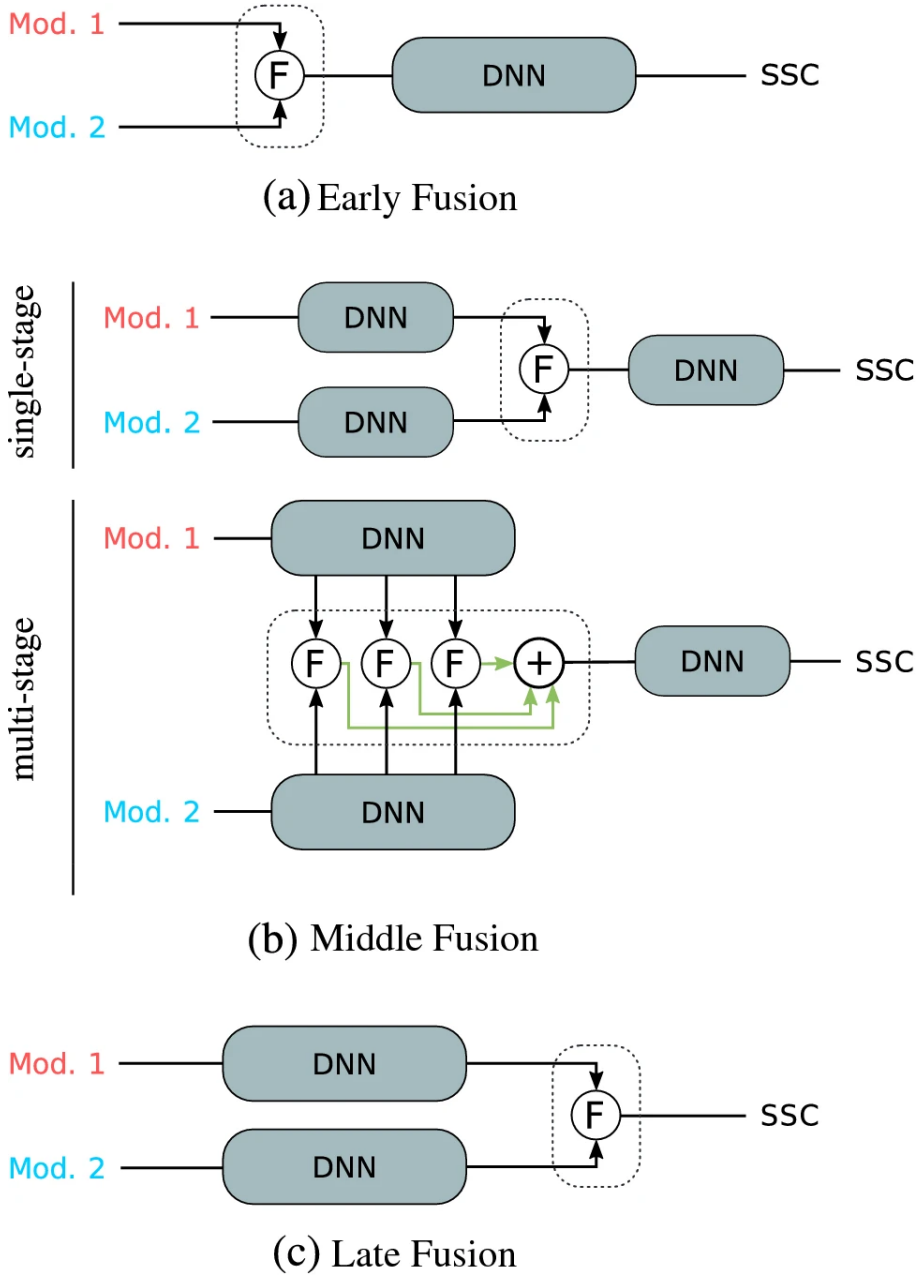


Figure 2.5: Overview of the different sensor fusion approaches in the context of deep learning for the semantic scene completion task. [37]

Chapter 3

Deep Implicit Representations for Semantic Scene Completion

Many architectures make use of implicit representations to represent the environment surrounding an autonomous vehicle. In this section, these architectures will be presented, as well as the metrics used to evaluate their performance, data augmentation techniques, different datasets that can be used for training, and some possible sensor fusion approaches.

3.1 Semantic Scene Completion

Semantic Scene Completion (SSC) is the problem of reconstructing a dense semantically labeled representation of a 3D scene from an incomplete representation of that scene. In most cases, the incomplete representation of the scene is significantly sparser than the actual scene. The complexity of this task stems from the fact that there are often large chunks of missing data due to occlusions and sensing sparsity. The most common sensors in data acquisition for Semantic Scene Completion are RGB and RGB-D cameras, and LiDAR. [37]

This task can be divided into two subtasks: scene completion, which involves reconstructing the geometry of the scene, and semantic completion or semantic segmentation, which involves inferring the semantic labels of the scene.

In this dissertation, the main focus will be outdoor scene reconstruction, specifically for autonomous driving applications. The following sections will present some of the most common architectures used for Semantic Scene Completion in outdoor scenes.

3.1.1 Metrics

The metrics presented in this section are used to evaluate the performance of the models in the Semantic Scene Completion task. They are the ones used by the SemanticKITTI dataset [2].

3.1.1.1 IoU

To measure geometric completion performance, the preferred metric is the Intersection over Union (IoU). This metric is commonly used in the binary free/occupied scene representation. [37]. It is given by the quotient of the intersection volume between the predicted and ground truth scene and the union volume between them.

3.1.1.2 mIoU

The mean Intersection over Union (mIoU) or Jaccard Index is used to measure the semantic performance of the model. It considers all semantic classes for prediction without considering free space. It is defined as

$$mIoU = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c + FN_c} \tag{3.1}$$

where TP_c, FP_c and FN_c are the true positives, false positives, and false negatives predictions for class c, respectively. [37]

3.1.2 Datasets

Training a deep learning model requires a large amount of data. For autonomous driving, there are quite a few publically available datasets. Some of these datasets are: SemanticKITTI [2], Argoverse 2 [45], nuScenes [4], Waymo Open Dataset [1] and KITTI-360 [25].

Roldão et al. [37] categorizes a dataset as Semantic Scene Completion ready when it contains pairs of sparse/dense data with semantic labels. The dataset that will be most used in this dissertation is the SemanticKITTI [2] dataset. It is based on the KITTI vision odometry dataset [14], showing inner city traffic, residential areas, highway scenes, and countryside roads around Karlsruhe, Germany, but provides semantic annotations for the points clouds. To maintain consistency with the original dataset, SemanticKITTI maintains the same splits for training, testing, and validation. The dataset is composed of 22 sequences, splitting sequences 00 to 10 as the training set and 11 to 21 as the test set. It contains 28 classes, including classes distinguishing non-moving and moving objects. Overall, the classes cover pedestrians, but also functional classes for ground, like parking areas and sidewalks.

A representation of the class distribution in the SemanticKITTI dataset can be seen in Fig. 3.1. In Fig. 3.2, we can also see a single scan and multiple superimposed scans from the SemanticKITTI dataset. Fig. 3.3 shows an example of a voxel grid used by SemanticKITTI for the Semantic Scene Completion task.

3.1.3 State of the Art

This section presents the state-of-the-art models used for the Semantic Scene Completion task.

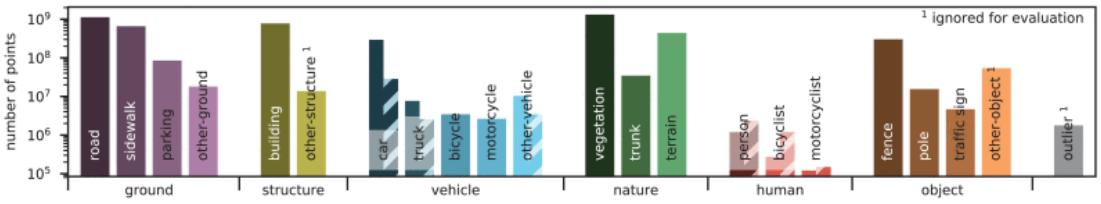


Figure 3.1: Class distribution in the SemanticKITTI dataset. [2]

3.1.3.1 LiDAR-based

Up until recently, S3CNet [8] presented the highest mIoU of all models for this task. From a LiDAR scan, S3CNet constructs two sparse tensors (using Minkowski Engine [9]) that encapsulate the scene into 2D and 3D representations. Then, each tensor is passed through its own SSC network to semantically complete the scene in its respective dimension. Finally, using a dynamic voxel fusion method, the reconstructed scene is further densified.

Also published in 2020, LMSCNet [36] presented the best completion performance at the time of publishing. It uses a UNet [38] architecture, employing mostly 2D convolutions along the height axis, akin to a bird-eye view. Then, 3D segmentation heads are used to perform 3D semantic segmentation and completion. An interesting feature of this proposed architecture is the possibility of creating a lower-resolution output that reduces the computation and memory requirements.

In 2021, SISC [23] and Local-DIFs [35], both architectures based on Deep Implicit Representation, were published, with the latter achieving the best mIoU at the time of publishing. Both architectures are described in detail in sections 3.2.2.2 and 3.2.2.3, respectively. Still, in 2021, SSC-SA [48] was published, achieving the best completion performance until today. It uses a 3D semantic segmentation and 2D scene completion branches, following a UNet-like structure. Features from the semantic segmentation branch are fused with the first levels of the scene completion branch.

The recently published SCPNet [46] achieves the highest mIoU among all the models (36.7% over S3CNet's 29.5%). It uses an architecture with two sub-networks: a completion sub-network and a segmentation sub-network based on Cylinder3D [52]. It is trained based on a knowledge distillation strategy using a teacher network that learns from multiple input frames, thus achieving better completion performance.

The quantitative results of all these architectures can be found in Table 3.1.

3.1.3.2 RGB-Based

RGB-only methods appear later, with one of the earliest architectures, MonoScene [5], published only in 2022. This architecture features 2D and 3D UNets bridged by a custom Features Line of Sight Projection module responsible for lifting 2D features into plausible 3D locations. Between the 3D encoder and decoder, a 3D Context Relation Prior component is inserted to capture long-range semantic context information.

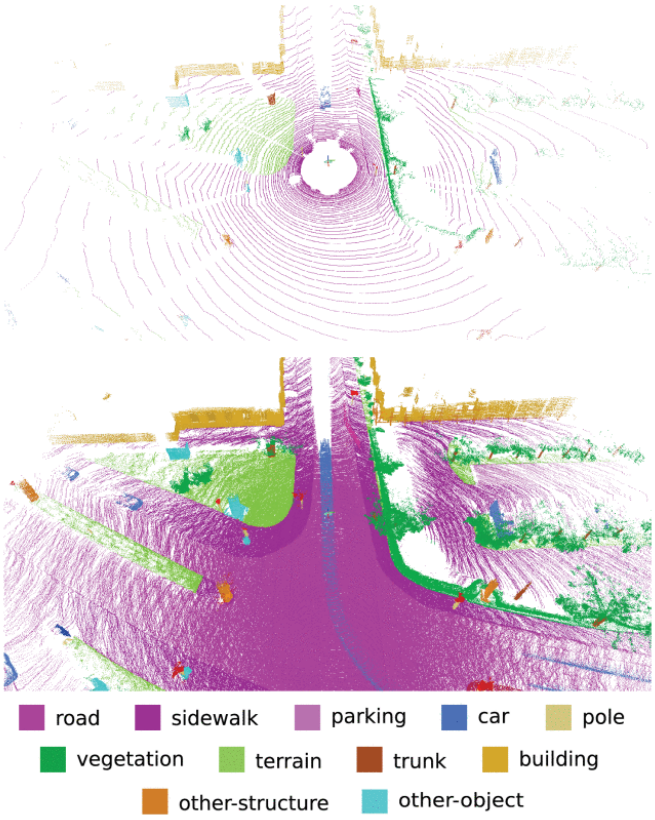


Figure 3.2: Single scan and multiple superimposed scans from the SemanticKITTI dataset. [2]

In 2023, OccDepth [29] and VoxFormer [24] were published, achieving better performance than MonoScene. OccDepth uses stereo images as input. It uses a Stereo-SFA module to lift features into 3D space, an OAD module to enhance depth prediction, and a 3D UNet to extract geometry and semantics. VoxFormer first extracts 2D features using a ResNet-50 [19] network. Then, depth is estimated using a depth prediction model. Finally, a cross-attention and a self-attention mechanism followed by an upsampling module are used to generate the final 3D output.

The quantitative results of these RGB-only architectures can be found in Table 3.2.

Table 3.1: Quantitative results of the state-of-the-art LiDAR-based models on the Semantic Scene Completion task from the SemanticKITTI dataset [2]

Model	IoU (%)	mIoU (%)
S3CNet (2020) [8]	45.6	29.5
LMSCNet (2020) [36]	55.3	17.0
SSC-SA (2021) [48]	**58.8**	23.5
Local-DIFs (2021) [35]	57.7	22.7
SISC (2021) [23]	50.8	18.0
SCPNet (2023) [46]	56.1	**36.7**

Figure 3.3: Left: Visualization of the incomplete voxel grid input for SSC. Right: Corresponding target output representing completed and fully labeled 3D scene. [2]

3.2 Architectures using Deep Implicit Representations

3.2.1 Occupancy-based

This section will describe the architectures that use occupancy (section 2.4.2) as their main representation.

3.2.1.1 Occupancy Networks

Occupancy networks (ONet) [28] are models that aim to learn the occupancy (section 2.4.2) of the entire 3D space and consequentially be able to represent 3D objects implicitly. The method proposed can learn this representation from various inputs, such as point clouds, images, or voxel representations. To learn the network parameters, random points are sampled from the 3D bounding volume of the desired object. At inference time, the MISE (section 3.4.2) algorithm is used to generate the resulting mesh.

3.2.1.2 Convolutional Occupancy Network

Convolutional Occupancy Networks [32] leverage CNNs to improve upon the previously mentioned Occupancy networks.

Unlike Occupancy Networks, Convolutional Occupancy Networks extract local features using a task-specific convolutional encoder. It can be done either in a 2D plane using an orthographic projection of the points or in 3D space.

To retrieve the features associated with a query point **p**, bilinear interpolation or trilinear interpolation is used. The resulting feature vector is used as an input to an Occupancy Network, which

Table 3.2: Quantitative results of the state-of-the-art RGB-based models on the Semantic Scene Completion task

Model	IoU (%)	mIoU (%)
MonoScene (2022) [5]	34.2	11.1
VoxFormer (2023) [24]	44.2	13.4
OccDepth (2023) [29]	**45.1**	**15.9**

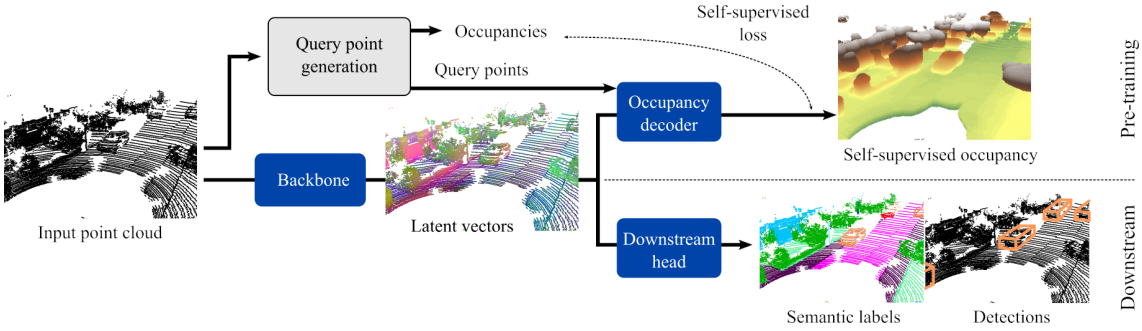


Figure 3.4: Overview of the ALSO network [3]

returns the desired occupancy. At training time, the query points are uniformly sampled from the volume of interest.

3.2.1.3 ALSO

ALSO [3] is a network that uses the scene completion task to improve other downstream tasks. The strategy involves pretraining a self-supervised occupancy decoder and feeding the learned latent vectors to the downstream task. The intuition is that if the network can reconstruct the scene, then some semantic information will also be learned. It achieves state-of-the-art results on both semantic segmentation and detection tasks.

To train the network, three points are generated for each input LiDAR point. The first step to obtaining these points is drawing the line between the query point $\mathbf{p}$ and the sensor location $\mathbf{c}$. The first two points are acquired by moving a certain δ distance along the line, starting in point $\mathbf{p}$. The point closer to the sensor is guaranteed to be free space, and the second point, farthest from the sensor, will be considered occupied space. This can't be verified as this method is self-supervised, which means there is no ground truth, but it is a good approximation. The δ parameter should be adjusted to the minimum thickness of the scene objects. The third point is randomly sampled from the line between the two points and is guaranteed to be free space as well.

Fig. 3.4 shows the architecture of the ALSO network.

3.2.2 SDF-based

This section will describe the architectures that use SDF (section 2.4.1) as their main representation.

3.2.2.1 SIREN

SIRENs [43] are a class of networks that use the sine function as an activation function. This has many uses, such as representing images, wavefields, video, sound, and their derivatives. More relevant to our problem is the representation of boundary value problems, specifically, the Eikonal equations that return a Signed Distance Function. This SDF represents the 3D surface of a scene or object. As can be seen in Fig. 3.5, the sine function was better at representing the SDF than the

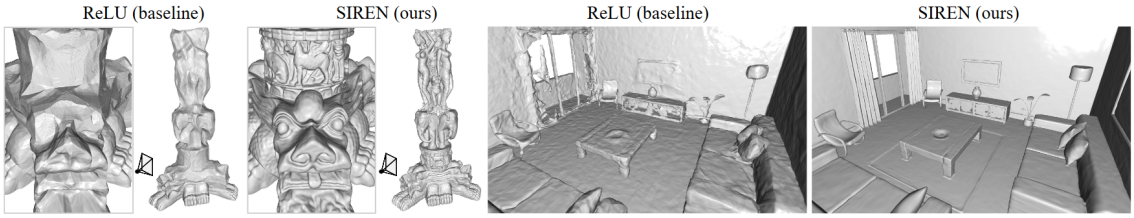


Figure 3.5: SIREN object and scene representation [43]

ReLU function. For training, each minibatch contains an equal number of points on and off the mesh, randomly sampled.

3.2.2.2 SISC

The Semi-supervised Implicit Scene Completion from Sparse LiDAR [23] publication proposes an architecture that tackles the Semantic Scene Completion task, trying to reconstruct a geometric representation of the scene as well as its semantic information. This paper uses the SemanticKITTI (section 3.1.2) dataset for the results.

The proposed architecture (see Fig. 3.6) has two main components, a discriminative model that generates latent shape embeddings and a generative model that aims to predict the final SDF of the scene.

Before entering the discriminative model, the point cloud input must be voxelized. Since the dataset used for training is SemanticKITTI, all the voxelized data is already available.

The discriminative model uses the voxelization of the LiDAR points Ω_1 as input and outputs the shape embeddings. This model uses a multiscale encoder-decoder architecture called ComNet, which can be seen in Fig. 3.7. Since the input is sparse, it uses the Minkowski Engine [9], which is a library for creating neural networks of sparse tensors. Overall, this model works by using the decoder modules to generate new voxels, thus using shape completion to predict the shape embeddings. However, the sparsity of the input is lost if the voxels are constantly generated. To fix this problem, a pruning block is used to prune off redundant voxels, which contains a convolutional layer to determine the binary classification results of whether a voxel should be pruned. This pruning block is supervised by a binary cross-entropy loss function:

$$\mathscr{L}_{com} = -\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}\left[y_{i,j}\log p_{i,j} + (1-y_{i,j})\log(1-p_{i,j})\right] \tag{3.2}$$

where m is the count of supervised blocks, n_i is the number of voxels in the i-th block, $y_{i,j}$ and $p_{i,j}$ are the ground truth and predicted probability of the existence of voxel i, respectively.

A binary cross entropy loss supervises the pruning block. The result of this process is a scene divided into voxels with edges of length b, where the latent shape information of $\mathbf{b}^3$ voxels is aggregated into a single voxel.

After generating the shape embedding volume, the pointwise shape embedding for a query point p needs to be obtained. Their proposed method uses trilinear interpolation to sample the

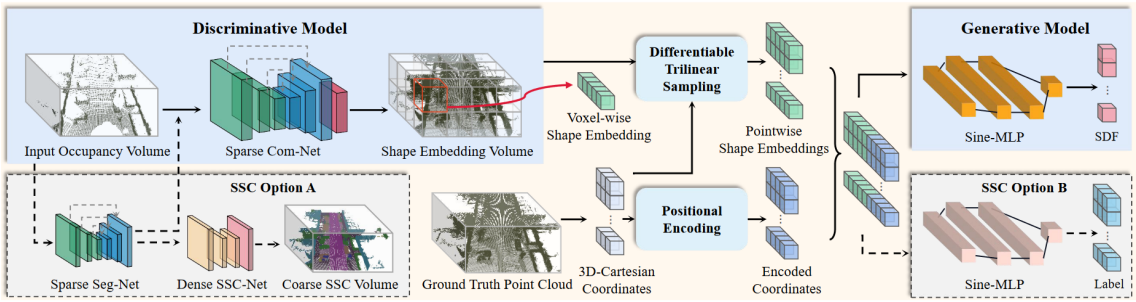


Figure 3.6: SISC architecture. [23]

pointwise shape embedding from its eight nearest voxel centers. The trilinear sampling mechanism is differentiable to allow for the cooperative training of the discriminative and generative models. A positional encoding module is used to represent more geometric details of the signed distance field.

The final module is the generative model. It utilizes a modified SIREN (section 3.2.2.1) network as a backbone. This model receives both the query point coordinates and the shape-wise point embeddings and predicts the SDF value of that specific point.

To predict semantic information, the network has two optional SSC models, option A or option B. Option A uses a sparse Seg-Net similar to the previously mentioned Com-Net for semantic segmentation and a dense convolutional network named SSC-Net to predict coarse semantic completion results. This is finally mapped to the scene representation. Option B leverages a second parallel SIREN network to predict the class probabilities. This network is supervised with a multi-classification cross-entropy loss.

For training, the same number (N) of on-surface and off-surface points are sampled from the dense ground truth.

The results achieved by this network are not state-of-the-art, with option A reaching 50.1% IoU geometric completion and 20.2% mIoU semantic segmentation. Option B achieves 50.8% IoU geometric completion and 18.0% mIoU semantic segmentation. The source code for this architecture is publicly available, which makes it very interesting for quickly testing improvements, as the paper's results can more easily be reproduced.

3.2.2.3 Local-DIFs

Local-DIFs [35] is an architecture that aims to tackle the Semantic Scene Segmentation task, which involves estimating the geometry and semantics of the scene concurrently. Similarly to the SISC architecture 3.2.2.2, it uses the SemanticKITTI dataset for training but extracts features directly from the point cloud instead of using the voxelized input.

The main objective of this architecture is, given a LiDAR scan, to output a corresponding scene completion function:

$$f^c_{LDIF} : \mathbb{R}^3 \rightarrow [0,1]^{N+1}, \tag{3.3}$$

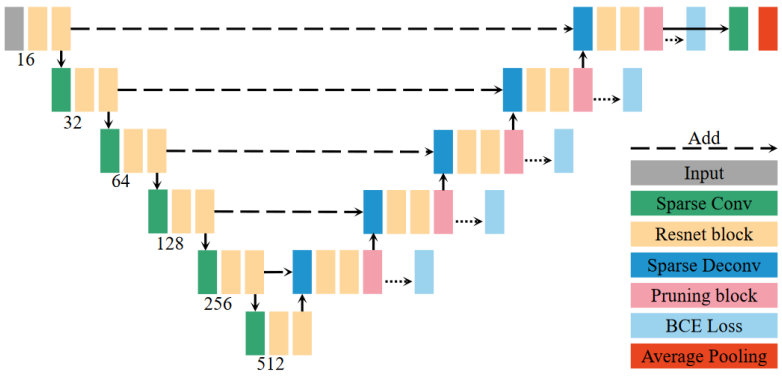


Figure 3.7: Network structure of Com-Net. [23]

which maps every 3D position $\mathbf{p}$ to a probability vector that represents its semantic class. Positions belonging to objects in the scene are categorized into N semantic classes. The additional class $(N+1)$ represents the positions that are not occupied by any object.

The global f^c_{LDIF} function is built from many local functions f_L. Each local function has two inputs: the coordinate of interest $\Delta\mathbf{p}$ and a conditioning vector $\mathbf{c}_v$. In other works aimed at single-object representation, the conditioning vector is of fixed size. However, in Local-DIFs, multiple 2D feature grids are used where each entry is a conditioning vector for a local function. Three grids are used, each with its own feature resolution.

The loss function used in this architecture is composed of three different loss function components: semantic loss L_S, geometric loss L_G, and consistency loss L_C. The semantic loss L_S is the cross-entropy loss between the classification output vector and semantic ground truth. This loss is not evaluated for free space targets. The geometric loss L_G is the binary cross-entropy between the sum of the semantic class probabilities for all objects and the remaining free space probability. The consistency loss L_C is the Jensen-Shannon divergence between probability distributions predicted by the local segmentation functions on the support region of consistency points $\mathbf{p}$. The overall loss is defined as

$$L = \lambda_S \sum_P L_S + \lambda_G \sum_P L_G + \lambda_C \sum_P L_C \tag{3.4}$$

Results-wise, the Local-DIFs architecture achieves state-of-the-art results in the geometric completion performance with 57.7% IoU. In the semantic completion task, Local-DIFs achieves 22.7% mIoU.

A representation of the Local-DIFs architecture can be seen in Fig. 3.8.

3.3 Data Augmentation for Point Clouds

As previously stated in section 2.2.2, data augmentation is a technique used to increase the size of the dataset and consequently reduce the overfitting of the models. In the specific case of point

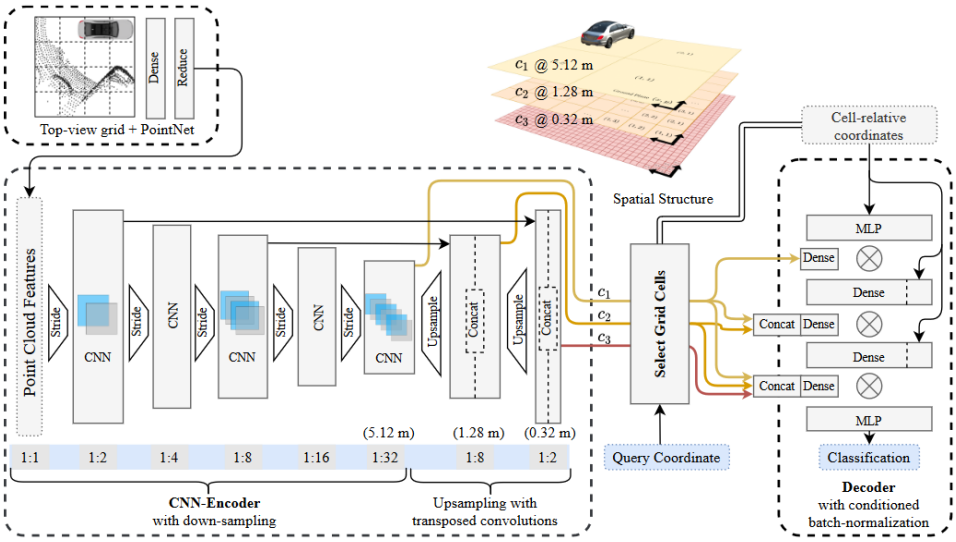


Figure 3.8: Local-DIFs architecture. [35]

clouds, simple geometric transformations can be used akin to the ones used in 2D images. However, particular caution is required when augmenting point clouds as the transformations performed can generate a point cloud that could never be captured by a real LiDAR sensor, making the use of structure-dependent networks impossible. [18]

Common transformations can be categorized as global and local. Global transformations affect the entire point cloud and include rotations, flips, and point-dropping. Local transformations affect only a specific area of the point cloud and include drop boxes, paste boxes, swap backgrounds, and adding feature noise in a frustum. [22]

More complex data augmentation techniques are being proposed in the literature, making this a very active research area. Leng et al. [22] propose LidarAugment, which uses only two global hyperparameters to control all data augmentation policies. Hasecke et al. [18] proposes a method to merge two point clouds from the same dataset to generate a new one. One of the point clouds is rotated in a horizontal sensor resolution (maximum of 10º) step, and the points closer to the sensor are kept. In the same paper, a second method is also proposed. This method keeps a database of underrepresented items gathered from semantic annotations, which can then be injected into other point clouds. Xiao et al. [47] propose PolarMix, which also describes two different augmentation methods. The first method involved cutting a sector of two different scenes (on the same azimuth angles) and switching them. The second method involves sampling points from an initial scene, creating rotated copies of them, and then pasting them into another scene.

It is important to note that most of the literature doesn't make extensive use of these advanced point cloud augmentation techniques. Furthermore, as said previously, this is a very active research area, so future architectures might make more use of these techniques.

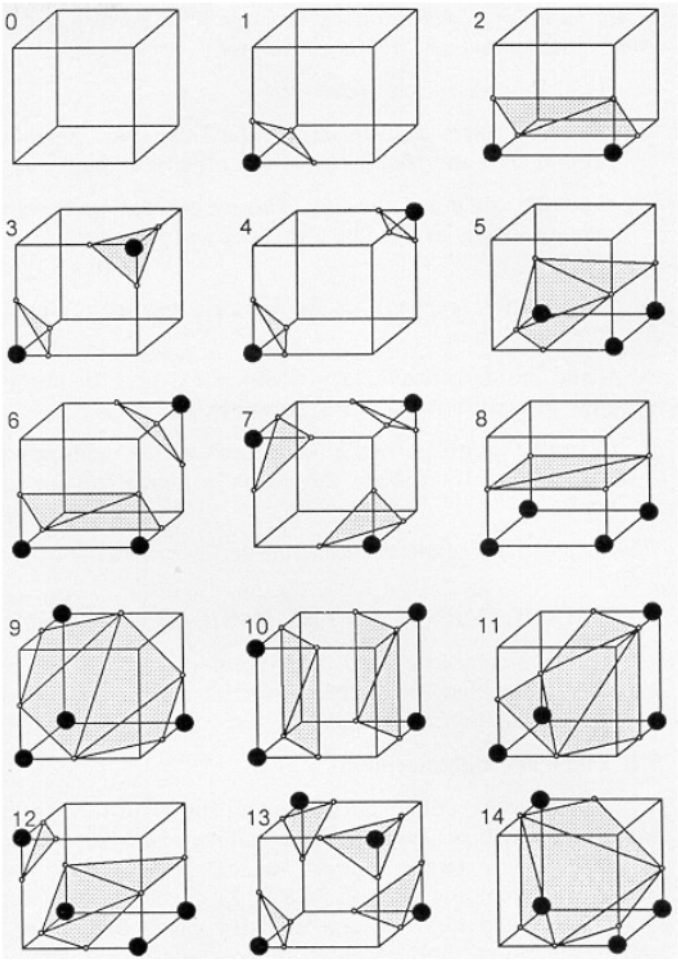


Figure 3.9: Triangulated Cubes from original Marching Cubes publication. [27]

3.4 Surface extraction

This section presents the methods used to extract the resulting meshes from occupancy and SDF-based networks.

3.4.1 Marching Cubes

The Marching Cubes algorithm [27] is used to generate a polygonal mesh from a scalar field. The first step of the algorithm consists of dividing space into a 3D grid. Then, each cube formed by adjacent grid points is visited and is triangulated based on whether its vertices belong to the surface. Since cubes have eight vertices, there are $2^8 = 256$ possible combinations for the cube triangulations. The original authors asserted that all those combinations could be obtained from 15 combinations using rotations and symmetries. Those 15 combinations were presented in the original publication and can be found in Fig. 3.9.

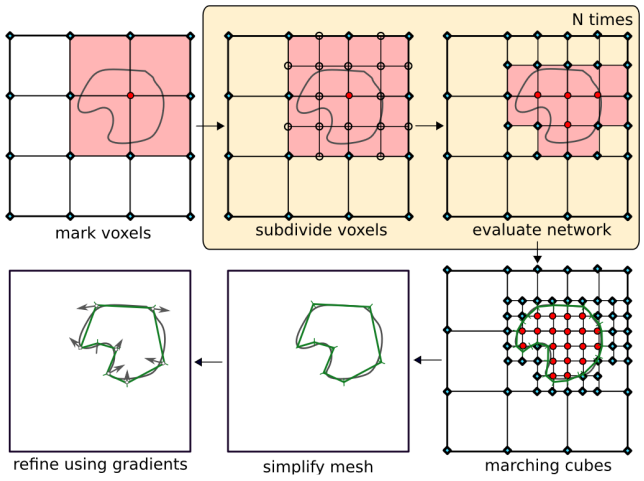


Figure 3.10: Multiresolution IsoSurface Extraction (MISE) algorithm

3.4.2 Multiresolution IsoSurface Extraction

Most occupancy and SDF-based models use the Multiresolution IsoSurface Extraction (MISE) method to extract the resulting surface. It was first presented in the Occupancy Networks [28] publication. The first step of the method is to discretize the volumetric space at an initial resolution and evaluate all the occupancy functions for each point $\mathbf{p}$ in this grid. Each point $\mathbf{p}$ is marked occupied if the occupancy function is higher than a threshold τ. Next, all voxels have two adjacent points with different occupancies. Then, all active voxels are subdivided into eight sub voxels, and all new grid points just added are re-evaluated. These steps are repeated until the desired resolution is reached. Next, the Marching Cubes [27] algorithm is applied to extract the final surface. Finally, the extracted mesh is simplified using the Fast-Quadric-Mesh-Simplification [13] algorithm and refined using gradient information.

Fig. 3.10 shows a visual representation of this algorithm.

3.5 Sensor Fusion

Mota [31] presents, in his dissertation, an attempt to study sensor fusion in the context of the Semantic Scene Completion task. The chosen architecture was SISC (section 3.2.2.2). The type of fusion chosen was early fusion, which involved simply concatenating the voxelized input with the pixel information from an RGB camera. The result achieved did not increase or decrease the performance of the original architecture in a very significant way.

The MVX-Net architecture, proposed in Sindagi et al. [42], presents two different approaches for fusing LiDAR scans with RGB information at the feature level. The task it tries to solve is 3D object detection. Both methods use a pre-trained Faster R-CNN [34] to extract features from the RGB image. This model was pre-trained for the 2D object detection task. The *PointFusion* method (Fig. 3.11), merges the features at an earlier level, as the point features are concatenated with the

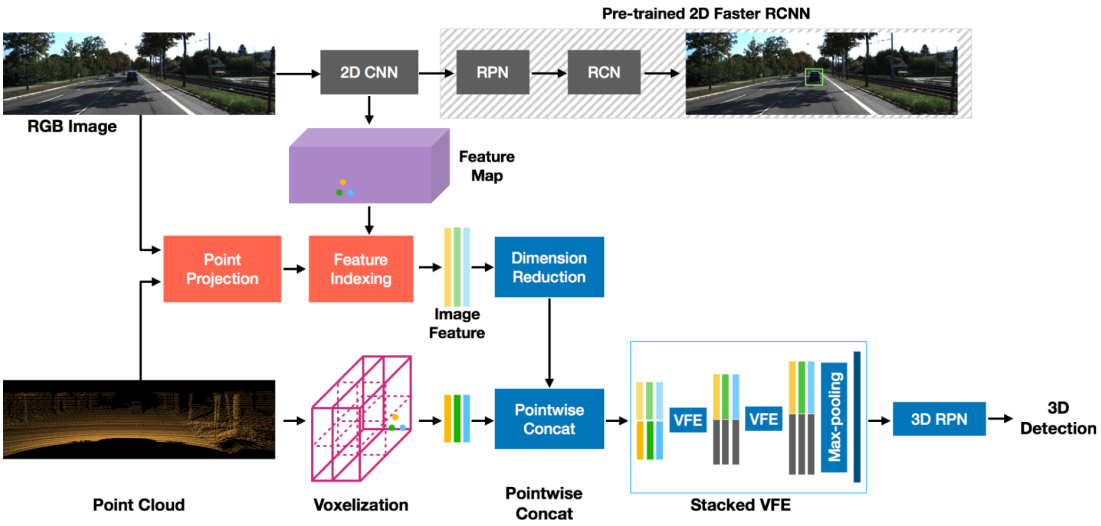


Figure 3.11: *PointFusion* method presented in MVX-Net. [42]

voxelized input. This is then used as input to a VoxelNet [51] network. In the *VoxelFusion* method (Fig. 3.12), image features are concatenated with the voxel features already extracted from the VoxelNet, making it a later fusion. The *PointFusion* method provides slightly better performance than the *VoxelFusion* method at the cost of higher memory consumption.

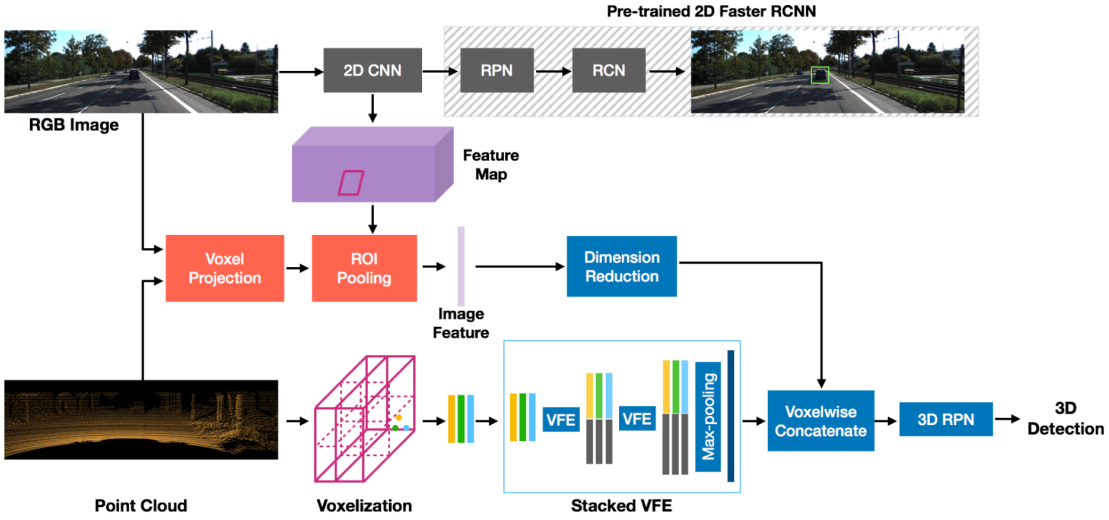


Figure 3.12: *VoxelFusion* method presented in MVX-Net. [42]

Chapter 4

Methodology

The previous chapters presented the state of the art of this work's main concepts and technologies. This chapter will present the methodology used to develop the proposed solution. We will start by presenting the main goals of this work, followed by the methodology used to achieve them.

4.1 Base model

As previously mentioned, the model that was chosen as a base for this work was the SISC model with the SSC option B activated. This model was described in detail in section 3.2.2.2. This model was chosen because its source code is available, so a reimplementation is not necessary, and because of the previous experience of the group with the model [31], which will later enable a direct comparison between the previously proposed early fusion method and the proposed middle fusion method.

This model will be modified in two significant ways. In the first experiment, the learned implicit representation (originally an SDF) will be replaced with occupancy. In a second experiment, a middle fusion module will be added to the network based on the MVX-Net architecture [42].

4.2 Occupancy

As a way to learn in a more deep way how the SISC model works, the first experiment was to replace the implicit representation used in the original model (SDF) with occupancy. Initially, the model's scene completion generative head outputs a real value which is interpreted as the distance to the surface representing the scene. Then, a threshold is applied to determine which values belong to the surface. For example, if the threshold is 0.0015, all values below that threshold are considered to belong to the surface, and all values above that threshold are considered free space.

In this experiment, the output of the scene completion generative head will output a value between 0 and 1, which will be interpreted as the probability of that voxel being occupied or free. This involves two essential changes to the original architecture. First, since the original model outputs a real value, no activation layer is used at the end of the generative scene completion

head. Thus, an activation function needs to be inserted at the end of the model, preferentially a function that clamps the values to the 0 to 1 range. The chosen activation function was the sigmoid function because of its properties: it is continuous, differentiable, symmetric, and bounded to the 0 to 1 range. It's also used in similar models for occupancy prediction. [3]

Second, the original loss function is no longer suitable for our needs and must be changed. The chosen loss function was Binary Cross Entropy (BCE) loss. This loss function is defined as:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i) \tag{4.1}$$

With these changes, there is still a threshold for the surface reconstruction in effect, but it will have to be in the 0 to 1 range as that is the new output range.

4.3 Sensor fusion

The main focus of this dissertation is to study the effects of using sensor fusion at the middle level in models which learn implicit representations. As such, the second and main experiment will be to add a middle fusion module to the SISC model. This module will be based on the MVX-Net architecture [42]. MVX-Net was chosen because it provided improved results over its base architecture (VoxelNet) in the object detection task and because it is an architecture that can be easily adapted to the SISC model without significant fundamental changes to the original architecture. The original MVX-Net architecture is shown in Fig. 3.12.

4.3.1 RGB Feature Extraction

MVX-Net extracts features from the RGB images using a Faster R-CNN [34] model. The Faster R-CNN model is an object detection model composed of two main modules, a Region Proposal Network (RPN) and a Fast R-CNN network. The RPN is responsible for generating region proposals, regions of the image likely to contain an object. The Fast R-CNN network is responsible for classifying the region proposals and refining their bounding boxes. Fig. 4.1 shows the Faster R-CNN model architecture.

The Region Proposal Network contains a fully convolutional network as its backbone. This backbone is responsible for extracting features from the input image, which are then used for the region proposal. This backbone can be changed to any other fully convolutional network. The backbone used in the original MVX-Net architecture is the VGG16 [41] network, following the original Faster R-CNN publication. Other relevant possibilities are the ResNet-50 [19] and MobileNet [20] networks.

To mimic the MVX-Net approach of using a pre-trained model for RGB feature extraction, a pre-trained Faster R-CNN model from the Torchvision framework was used. Because Torchvision does not host pretrained Faster R-CNN models with a VGG-16 backbone, the ResNet-50 backbone was used instead. This model was pretrained on the COCO dataset [26] for object detection.

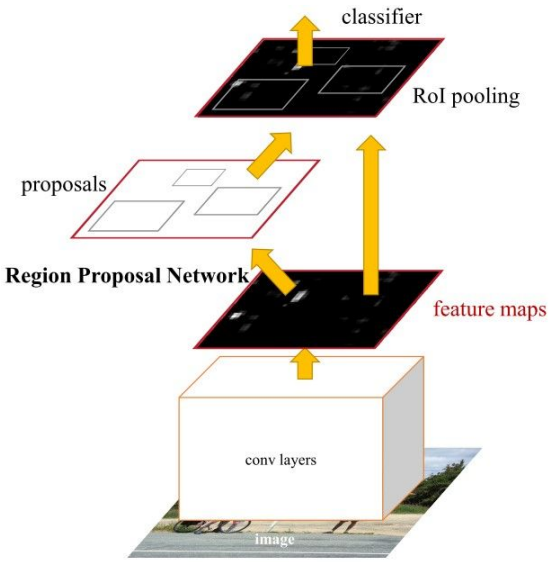


Figure 4.1: Faster R-CNN model architecture. RGB features used for fusion are the output of the *conv layers*. [34]

However, MVX-Net does not use the complete Faster R-CNN network for feature extraction. Instead, it only uses the fully convolutional backbone of the RPN to extract features from the RGB images and then uses that output features directly for sensor fusion. In the case of MVX-Net, the 512-dimensional RGB features are extracted from the output of the conv5 layer of the VGG16 backbone (Fig. 4.2).

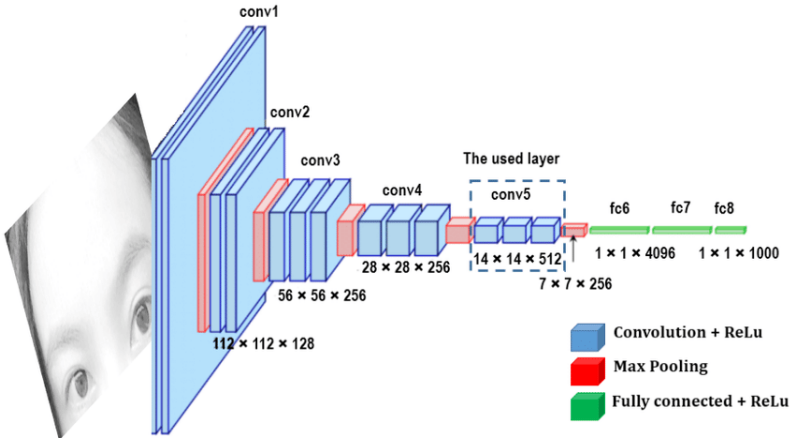


Figure 4.2: VGG-16 model architecture with the *conv5* layer highlighted [17]

The ResNet-50 used by Torchvision as the Faster R-CNN backbone outputs a multi-level feature pyramid with all five levels being 256-dimensional. There are some different approaches to using these features. One approach would be to use MVX-Net methods and just use the output of the layer with the lowest resolution. Another approach would be to use all levels and merge them, for example, using concatenation or sum operations. If concatenation is used, the resulting feature map is 1280-dimensional, while using summation or a single level results in a 256-dimensional

feature map.

4.3.2 Voxel grid projection

For each frame of the dataset, 32000 points are sampled from the corresponding voxel grid. The corresponding features must be retrieved from the RGB feature map for each of these points. All points must be projected to the image plane for this to be possible.

As previously stated, the dataset in use is the SemanticKITTI dataset, based on the KITTI Odometry dataset. This means that to project the LiDAR points onto the image, the calibration parameters from the KITTI Odometry dataset must be used. The projection for the KITTI dataset is calculated as shown in the following equation:

$$P_{img} = P^0_{rect} R^0_{rect} T_{Camera \leftarrow LiDAR} P_{LiDAR} \tag{4.2}$$

Where P_{img} is the projected point in the image plane, P^0_{rect} is the calibration matrix of the left color camera, R^0_{rect} is the rectifying rotation matrix of the left color camera, $T_{Camera \leftarrow LiDAR}$ is the transformation matrix from LiDAR coordinates to camera coordinates and P_{LiDAR} is the point in the LiDAR coordinate system. [50]

The results of this projection are shown in Fig. 4.3.

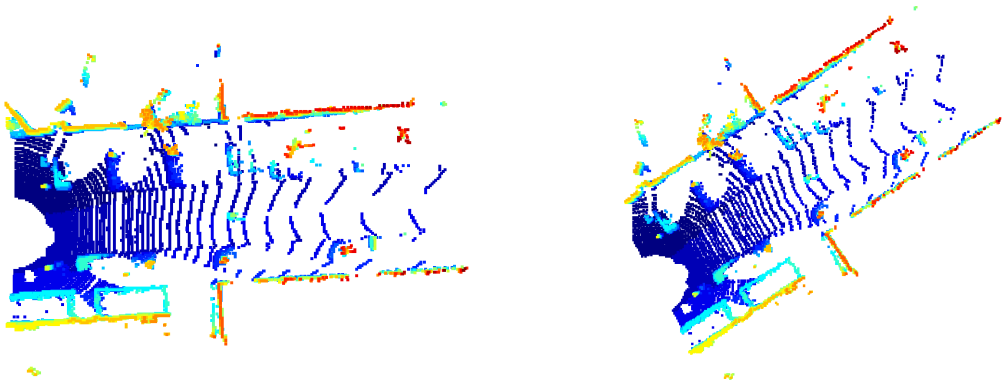
Figure 4.3: Points from a voxel grid projected onto the corresponding RGB image with the corresponding label color.

This process does not work if data augmentation is used, for example, when applying a rotation to the voxel grid. This happens because the voxel grid is rotated around an axis on the center of the grid while the camera is placed on the vehicle (see Fig. 4.4). If the above method were to be used in this situation, the points would appear in incorrect spots as represented in Fig. 4.5.

Thus, to enable data augmentation, a different approach must be used. Instead of projecting the points directly, the final transformation matrix suffers a rotation around the central axis of the voxel grid. This rotation can be calculated as follows:

$$T'_{Camera \leftarrow LiDAR} = T_{Camera \leftarrow LiDAR} Translation_{center} Rotation_{axis} Translation_{-center} \tag{4.3}$$

So whenever data augmentation is used, equation 4.2 is replaced by the following:



(a) Original voxel grid (b) Voxel grid rotated by 30 degrees

Figure 4.4: Visualization of voxel grid rotation.

$$P_{img} = P^0_{rect} R^0_{rect} T'_{Camera \leftarrow LiDAR} P_{LiDAR} \tag{4.4}$$

This way, the points are projected correctly, as shown in Fig. 4.6.

4.3.3 Dimensionality Reduction

After extracting the RGB features, the MVX-Net model uses a dimensionality reduction module to reduce the dimensionality of the features from 512 to 16. In the original architecture, this module first reduces the features from 512D to 96D and then to 16D using two fully-connected layers with Batch Normalization and ReLU. The fully-connected layers and ReLU functions are retained for the proposed implementation, but the batch normalization is removed. The input size is also adapted according to the defined approach for processing the feature pyramid.

4.3.4 Final architecture

The final architecture can be seen in Fig. 4.7. The architecture is composed of the original SISC model with the addition of the MVX-Net fusion module. The fusion module comprises the RGB feature extraction and dimension reduction modules. The features retrieved from the RGB feature extraction module are then concatenated with the features from the discriminative module of the SISC model and fed to both SISC generative heads.

This proposed architecture could also be used for late fusion, in which both the RGB and LiDAR features would be processed by the generative heads separately, and then fusion would require something like an averaging of the results. This depends on whether the network could learn from RGB features alone (see section 5.4.4).

Figure 4.5: Incorrect point projections when using voxel grid rotation.

Figure 4.6: Correct point projections when using voxel grid rotation. The labels are in the correct places.

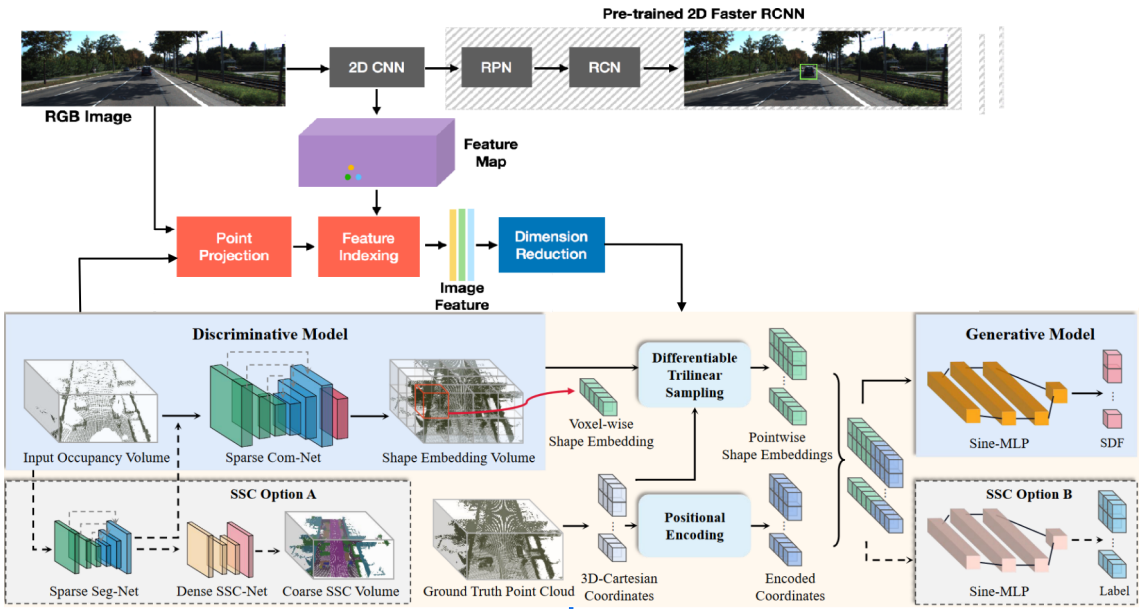


Figure 4.7: Complete middle fusion architecture

Chapter 5

Results and analysis

This chapter will present the results obtained from the experiments conducted with the model after the architectural changes presented in the previous chapter.

5.1 Test methodology

A series of experiments were conducted to understand how the previously mentioned modifications to the original SISC architecture affect its performance. In an ideal situation, each of the modifications would be run more than once to obtain results independent of randomness. However, each experiment, except the baseline, was run only once due to time and hardware constraints. The baseline was run four times to obtain a more accurate result for comparison. For each evaluation threshold (0.015, 0.012, 0.011, 0.01, 0.008, 0.006, 0.004, 0.002), a confidence interval (CI) of the results is calculated with a confidence level of 95%. These values will then be used to determine how impactful each modification is by checking if the result of the experiment sits inside or outside that confidence interval.

The confidence intervals are calculated with the following formula:

$$CI = \bar{x} \pm z\frac{s}{\sqrt{n}} \tag{5.1}$$

where $\bar{x}$ is the mean of the results, z is the z-score for a 95% confidence level, s is the standard deviation of the results and n is the number of runs. The z-score for a 95% confidence level is 1.96.

Unless otherwise stated, all models were trained until epoch 200 in line with the original SISC publication.

5.2 Baseline

The baseline chosen for this project was obtained by replicating the results obtained by the original SISC architecture and adding the dropout modification carried out in Mota [31]. As previously

Table 5.1: Baseline results. IoU (%) score for each threshold.

Run	0.015	0.012	0.01	0.008	0.004
1	47.97	48.68	48.54	48.34	48.34
2	48.94	48.78	48.55	48.26	47.90
3	48.12	47.91	47.76	48.26	47.89
4	49.25	49.13	48.99	48.72	48.32
Average	48.57	**48.63**	48.46	48.39	48.11
Std. Dev.	0.54	0.44	0.44	0.19	0.22
CI	[48.04, 49.10]	[48.19, 49.06]	[48.02, 48.89]	[48.21, 48.58]	[47.90, 48.33]

stated in section 5.1, the baseline was run four times to obtain a more accurate result for comparison. The results obtained are presented in Tables 5.1 and 5.2. A visual representation of the results can be seen in Fig. 5.1.

To enable a direct comparison with Mota's early fusion model [31], the model presented in that publication must be replicated. The model was trained with the same parameters as the baseline, except for the batch size, which was also increased from 2 to 4. The results obtained are presented in Table 5.3. In Mota's work, the early fusion models were only trained until epoch 50, so a direct comparison is not possible. However, the results obtained are consistent with the marginal improvement over the baseline. Both the IoU and mIoU scores are slightly higher than the baseline, 49.07% over 48.63% and 17.56% over 17.14%. A visual representation of the results can be seen in Fig. 5.2.

5.3 Occupancy

This section details the results obtained from the experiments conducted with the occupancy modifications described in section 4.2. Because of the different output range of the network, which is now in the 0 to 1 range, the thresholds used to evaluate it must also change. The new thresholds used are 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8. The results obtained are presented in Table 5.4.

From the results, the 0.3 threshold presents the best result, which steadily decreases as the threshold increases in value. At 0.3, this experiment presents a marginal improvement over the

Table 5.2: Baseline results. mIoU (%) score for each threshold.

Run	0.015	0.012	0.01	0.006	0.004
1	16.20	16.72	16.76	16.75	17.07
2	17.39	17.41	17.41	17.35	17.32
3	16.46	16.49	16.51	16.98	16.95
4	17.24	17.28	17.29	17.25	17.23
Average	16.82	16.98	16.99	17.08	**17.14**
Std. Dev.	0.50	0.38	0.37	0.23	0.14
CI	[16.33, 16.60]	[16.60, 17.35]	[16.63, 17.35]	[16.85, 17.31]	[17.00, 17.28]

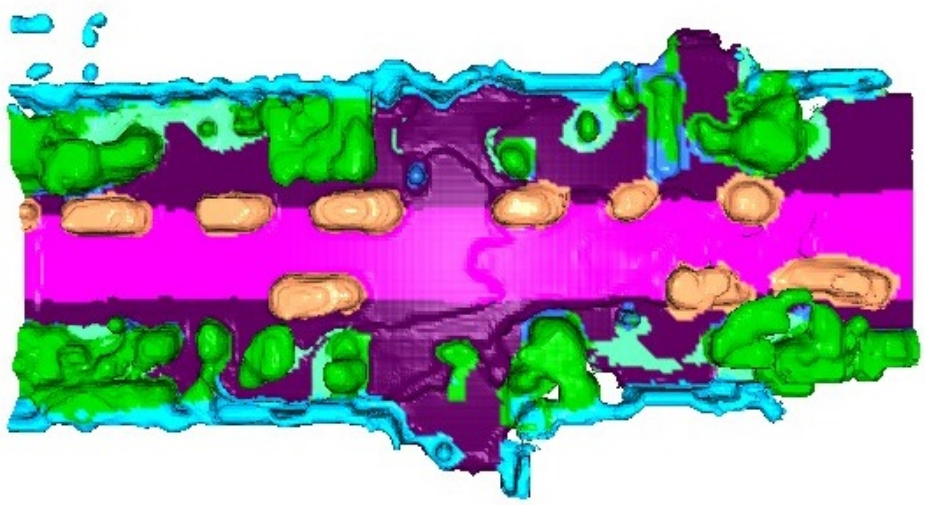


Figure 5.1: Semantic Scene Completion results from baseline for frame 545.

baseline. Because the thresholds changed, a comparison per threshold is no longer valid. Thus, only the best thresholds in each model can be compared. For scene completion, the best baseline threshold is 0.012, with an average IoU score of 48.63% and a standard deviation of 0.44. For occupancy, the best threshold is 0.3, with an IoU score of 49.48%. The results show that the occupancy modification presents an improvement over the baseline, with its score being above the baseline's confidence interval.

For the semantic segmentation part of the task, the best baseline threshold is 0.006, where the model achieved an average mIoU score of 17.14% with a standard deviation of 0.14. For the occupancy modification, the best threshold is again 0.3, with a mIoU score of 17.50%. Again, the results show that the occupancy modification presents a marginal improvement over the baseline scores by being above the confidence interval. A visual representation of the results can be seen in Fig. 5.3.

A possible explanation for this performance uplift is the reduced complexity of an occupancy function over an SDF, mainly because the occupancy function's output is bounded.

Table 5.3: Early fusion baseline results. IoU and mIoU score for each threshold.

Threshold	0.015	0.012	0.011	0.01	0.008	0.006	0.004	0.002
IoU (%)	**49.07**	48.92	48.83	48.72	48.39	48.21	48.06	47.91
mIoU (%)	17.52	17.55	**17.56**	17.55	17.51	17.49	17.48	17.45

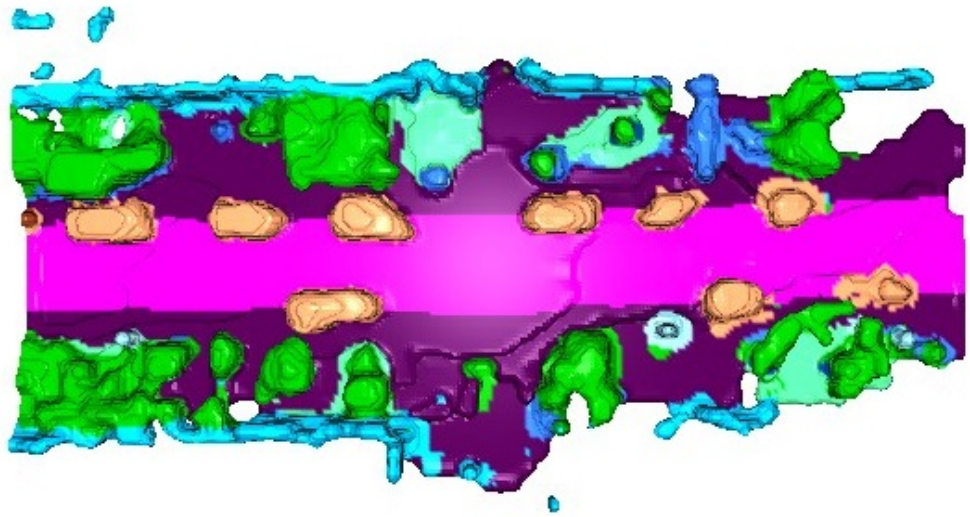


Figure 5.2: Semantic Scene Completion results from early fusion baseline for frame 545.

5.4 Middle fusion

This section details the results obtained from the experiments conducted with the middle fusion modifications described in section 4.3.

5.4.1 Default

The first relevant experiment consists of the default middle fusion model. This experiment uses the implementation exactly as defined in section 4.3. The initial intuition is that results should improve over the baseline because the model now has access to more information. This proved to be the case with MVX-Net, where it achieved better results than the non-multimodal approach.

The results of this experiment are presented in Table 5.5. Even though the results are better than the baseline, they are lower than expected, not even reaching the occupancy experiment results. The best threshold for scene completion is 0.006 with 49.19% IoU, which is 0.56% higher than the baseline's best threshold. For semantic segmentation, the best threshold is 0.002 with 17.30% mIoU, which is 0.16% higher than the baseline's best threshold. The results show that the middle fusion modification presents a marginal improvement over the baseline scores.

Compared with the early fusion approach, the middle fusion module presents a marginal improvement over the baseline scores. For scene completion, the best threshold for the early fusion

Table 5.4: Results of the occupancy output experiment. IoU and mIoU score for each threshold.

Threshold	0.2	0.3	0.4	0.5	0.6	0.7	0.8
IoU (%)	48.79	**49.48**	48.41	41.92	34.69	30.90	30.23
mIoU (%)	17.43	**17.50**	16.96	15.72	14.92	14.65	14.49

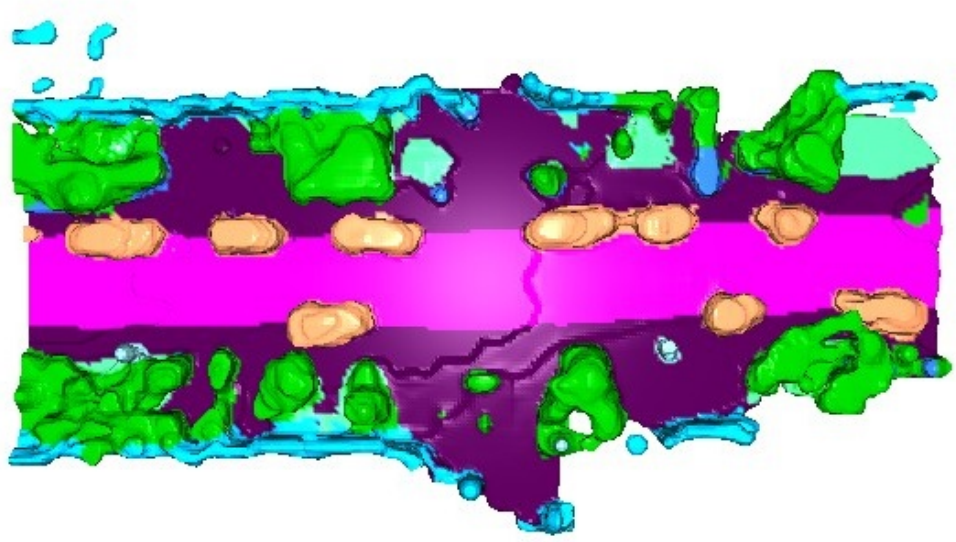


Figure 5.3: Semantic Scene Completion results from the occupancy experiment for frame 545 (threshold 0.3).

approach is 0.015 with 49.07% IoU, which is 0.12% lower than the middle fusion's best threshold. For semantic segmentation, the best threshold for the early fusion approach is 0.011 with 17.56% mIoU, which is 0.26% higher than the middle fusion's best threshold. The results show that the middle fusion modification presents a marginal improvement over the early fusion scores in scene completion but is worse in semantic segmentation. A visual representation of the results can be seen in Fig. 5.4. Because the results are within 1.0% of each other, qualitative differences are very hard to notice.

5.4.2 Backbone unfrozen

MVX-Net uses the backbone of a pretrained Faster R-CNN model to extract the RGB features from the images (section 4.3.1). In the original publication, the weights of this backbone are frozen, meaning that they are not updated during training. This results in the backbone not learning any new information.

This provides an interesting experiment possibility by unfreezing these weights. In practice, they will be updated by backpropagation like the rest of the network and could learn new information as the network trains. Table 5.6 presents the results obtained in this experiment.

Table 5.5: Results of the middle fusion output experiment. IoU and mIoU score for each threshold.

Threshold	0.015	0.012	0.011	0.01	0.008	0.006	0.004	0.002
IoU (%)	36.93	43.07	44.45	45.71	48.51	**49.19**	49.13	49.04
mIoU (%)	14.84	15.67	15.93	16.19	16.92	17.25	17.27	**17.30**

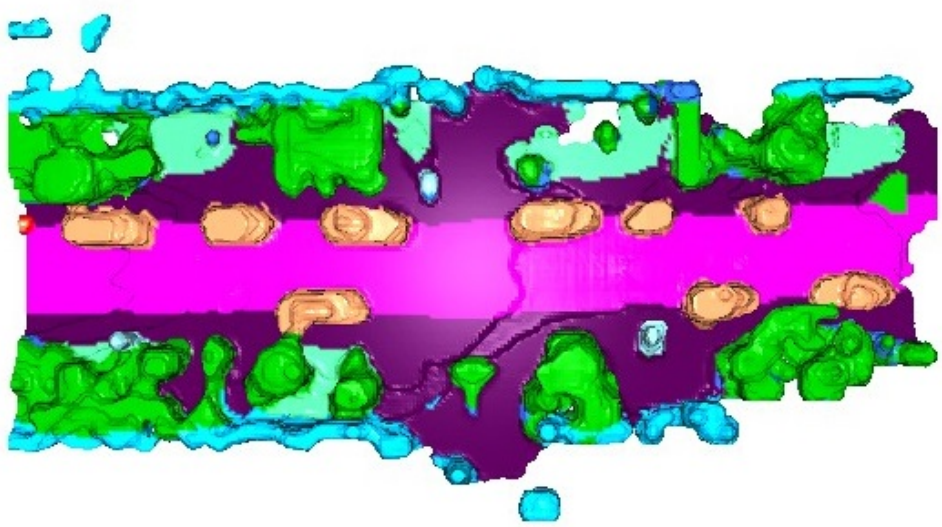


Figure 5.4: Semantic Scene Completion results from the middle fusion experiment for frame 545 (threshold 0.006).

Compared with the default middle fusion experiment, the results were about 0.29% lower at the best threshold (0.006) for the scene completion IoU. The semantic segmentation mIoU score was also lower by 0.16% at the best threshold (0.002). This is quite a small difference, so it appears unfreezing the weights does not influence the results in a significant way.

5.4.3 Occupancy

The way the middle fusion module is implemented does not interfere with the occupancy modifications described in section 4.2. This means the occupancy modifications can be applied alongside the middle fusion module. The results obtained in this experiment are presented in Table 5.7.

Much like the experiment results of the standalone occupancy experiment (section 5.3), the 0.3 threshold is still the best both in IoU and mIoU, with higher thresholds having a significant drop in performance. However, the results are lower than the standalone occupancy experiment, with the best IoU score being 0.55% lower and the best mIoU score being 0.01% lower.

Table 5.6: Results of the middle fusion experiment with Faster R-CNN backbone's weights unfrozen. IoU and mIoU score for each threshold.

Threshold	0.015	0.012	0.011	0.01	0.008	0.006	0.004	0.002
IoU (%)	36.12	41.64	43.83	47.08	48.67	**48.90**	48.86	48.76
mIoU (%)	14.60	15.43	15.83	16.44	16.90	17.06	17.12	**17.14**

Table 5.7: Results of the middle fusion experiment mixed with the occupancy output experiment. IoU and mIoU score for each threshold.

Threshold	0.2	0.3	0.4	0.5	0.6	0.7	0.8
IoU (%)	48.18	**48.93**	47.49	41.10	34.53	30.26	29.18
mIoU (%)	17.39	**17.49**	16.93	15.83	15.09	14.76	14.56

5.4.4 RGB-Only

This experiment aimed to understand if the middle fusion modifications to the model could enable the training with only RGB images. The process involves discarding the input from the discriminative model and only using the RGB image features and encoded coordinates as input for the generative heads.

This experiment ended in failure as the network failed to learn any meaningful information. The results obtained in this experiment are presented in Table 5.8. The best IoU score was 22.46%, and the best mIoU score was 2.53%, both at the 0.002 threshold. By analyzing an output frame from the validation sequence, it is clear that the network is not learning relevant information for scene reconstruction. The output frame is presented in Fig. 5.5.

A possible explanation for this failure is the lack of information provided by the extracted RGB features. These alone might not be enough to reconstruct the 3D scene. Some variations of this experiment were conducted, for example, increasing the final size of the RGB features from 16 to 32 or even 64, but the results were similar or worse. Thus, to learn from RGB images alone, a different network architecture should be used (see 3.1.3.2).

5.4.5 Final analysis

Table 5.9 presents a comparison between the best results from each experiment. The best IoU and mIoU scores for each experiment are presented, as well as the best threshold for each score. The early fusion replication produced better results than the baseline, with a 0.44% increase in IoU and a 0.42% increase in mIoU. Middle fusion was able to improve the scene completion IoU score by 0.12% over the early fusion model. However, that came at the cost of a 0.26% decrease in the mIoU score. This prevents any conclusions on which method is better, as the results are very similar.

The occupancy modifications were able to improve the IoU score by 0.85% over the baseline and the mIoU score by 0.36%. This modification proved to be the most effective in improving results, probably to the simplicity of the underlying representation.

Table 5.8: Results of the RGB-only middle fusion experiment. IoU and mIoU score for each threshold.

Threshold	0.015	0.012	0.011	0.01	0.008	0.006	0.004	0.002
IoU (%)	17.54	18.98	19.27	19.54	20.44	21.15	22.09	**22.46**
mIoU (%)	2.32	2.35	2.36	2.37	2.42	2.46	2.50	**2.53**

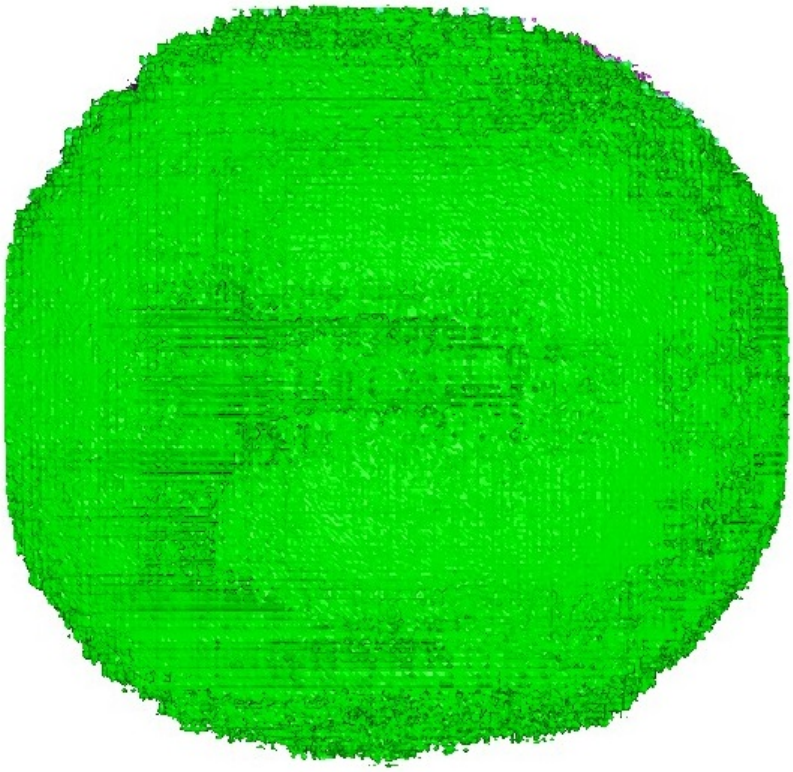


Figure 5.5: Output frame from the RGB-only middle fusion experiment. The network is not learning anything meaningful.

The final two experiments with middle fusion were unsuccessful in improving the metrics already obtained. Both occupancy and the unfreezing of the Faster R-CNN backbone produced worse results than their baseline. The middle fusion + occupancy experiment produced a 0.55% decrease in IoU with virtually the same mIoU. The unfreezing of the backbone produced a 0.29% decrease in IoU and a 0.16% decrease in mIoU over the baseline middle fusion experiment.

Table 5.9: Comparison between the best results of each experiment.

Experiment	Threshold	IoU (%)	Threshold	mIoU (%)
Baseline	0.012	48.63	0.004	17.14
Early Fusion	0.015	49.07	0.011	**17.56**
Occupancy	0.3	**49.48**	0.3	17.50
Middle Fusion	0.006	49.19	0.002	17.30
Middle Fusion + Occupancy	0.3	48.93	0.3	17.49
Middle Fusion + Backbone unfrozen	0.006	48.90	0.002	17.14

Chapter 6

Conclusions and future work

This work's main goal was to understand if using LiDAR-RGB sensor fusion could help networks that learn Deep Implicit Representations. These sensor fusion techniques are already applied in other tasks, such as object detection, succeeding in increasing performance over the single modality baselines. Deep Implicit Representation networks learn SDF or occupancy functions from their input, be it a point cloud generated by a LiDAR sensor or an RGB image, among others.

This work employed these networks specifically to solve the Semantic Scene Completion task, which involves learning how to reconstruct scene geometry and semantic information only from a sparse representation.

In the end, it was possible to conclude that using LiDAR-RGB sensor fusion, more specifically middle fusion or features fusion, does help these networks, but in a very marginal way. Compared with Mota's early fusion model, the results encountered in this work over the baseline presented an improvement in scene completion, but the semantic segmentation score was lowered. While both models presented better results than the baseline, the middle fusions geometric completion IoU was 0.12% higher than the early fusion's, and the semantic segmentation mIoU was 0.26% lower. However, the results obtained from switching the learned representation were more favorable than the ones obtained from sensor fusion, except for the semantic segmentation mIoU, which was 0.06% lower than the early fusion score. Thus, it is not possible to conclude if middle fusion is better than early fusion, as the results were very similar.

There are some possibilities for future work. One of them would be to study how to use DIRs to aid in downstream tasks such as semantic segmentation or object detection. The ALSO model already approaches this problem, but testing multimodality could be an interesting research path. Another possibility would be implementing a new RGB module based on the state-of-the-art RGB-only SSC models and using a late fusion module. They were published quite late in the development of this work, and there needed to be more time to implement them.

References

[1] Waymo open dataset: An autonomous driving dataset, 2019.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences, August 2019. arXiv:1904.01416 [cs].

[3] Alexandre Boulch, Corentin Sautier, Björn Michele, Gilles Puy, and Renaud Marlet. ALSO: Automotive Lidar Self-supervision by Occupancy estimation, December 2022. arXiv:2212.05867 [cs].

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving, May 2020. arXiv:1903.11027 [cs, stat].

[5] Anh-Quan Cao and Raoul de Charette. MonoScene: Monocular 3D Semantic Scene Completion. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3981–3991, New Orleans, LA, USA, June 2022. IEEE.

[6] Cem Dilmegani. What is data augmentation? techniques and examples in 2023. https://research.aimultiple.com/data-augmentation/, 2022. Accessed: 2023-01-22.

[7] T. Chan and W. Zhu. Level Set Based Shape Prior Segmentation. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 1164–1170, San Diego, CA, USA, 2005. IEEE.

[8] Ran Cheng, Christopher Agia, Yuan Ren, Xinhai Li, and Liu Bingbing. S3CNet: A Sparse Semantic Scene Completion Network for LiDAR Point Clouds, December 2020. arXiv:2012.09242 [cs].

[9] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks, June 2019. arXiv:1904.08755 [cs].

[10] Fahmi Nurfikri. An illustrated guide to artificial neural networks. https://towardsdatascience.com/an-illustrated-guide-to-artificial-neural-networks-f149a549ba74, 2020. Accessed: 2023-01-20.

[11] Jamil Fayyad, Mohammad A. Jaradat, Dominique Gruyer, and Homayoun Najjaran. Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review. *Sensors*, 20(15):4220, July 2020.

[12] Kyle Gao, Yina Gao, Hongjie He, Denning Lu, Linlin Xu, and Jonathan Li. NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review, November 2022. arXiv:2210.00379 [cs].

[13] M. Garland and P.S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 263–269,, Research Triangle Park, NC, USA, 1998. IEEE.

[14] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, Providence, RI, June 2012. IEEE.

[15] Ben Graham. Sparse 3D convolutional neural networks, August 2015. arXiv:1505.02890 [cs].

[16] Benjamin Graham. Spatially-sparse convolutional neural networks, September 2014. arXiv:1409.6070 [cs].

[17] Walid Hariri. Efficient Masked Face Recognition Method during the COVID-19 Pandemic. preprint, In Review, July 2020.

[18] Frederik Hasecke, Martin Alsfasser, and Anton Kummert. What Can be Seen is What You Get: Structure Aware Point Cloud Augmentation. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 594–599, Aachen, Germany, June 2022. IEEE.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs].

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, April 2017. arXiv:1704.04861 [cs].

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[22] Zhaoqi Leng, Guowang Li, Chenxi Liu, Ekin Dogus Cubuk, Pei Sun, Tong He, Dragomir Anguelov, and Mingxing Tan. LidarAugment: Searching for Scalable 3D LiDAR Data Augmentations, October 2022. arXiv:2210.13488 [cs].

[23] Pengfei Li, Yongliang Shi, Tianyu Liu, Hao Zhao, Guyue Zhou, and Ya-Qin Zhang. Semi-supervised Implicit Scene Completion from Sparse LiDAR, November 2021. arXiv:2111.14798 [cs].

[24] Yiming Li, Zhiding Yu, Christopher Choy, Chaowei Xiao, Jose M. Alvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. VoxFormer: Sparse Voxel Transformer for Camera-based 3D Semantic Scene Completion, March 2023. arXiv:2302.12251 [cs].

[25] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D, June 2022. arXiv:2109.13410 [cs].

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, February 2015. arXiv:1405.0312 [cs].

[27] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm.

[28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space, April 2019. arXiv:1812.03828 [cs].

[29] Ruihang Miao, Weizhou Liu, Mingrui Chen, Zheng Gong, Weixin Xu, Chen Hu, and Shuchang Zhou. OccDepth: A Depth-Aware Method for 3D Semantic Scene Completion, February 2023. arXiv:2302.13540 [cs].

[30] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, August 2020. arXiv:2003.08934 [cs].

[31] João Pedro Pinto Mota. Deep Implicit Representations in Autonomous Driving. page 68.

[32] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks, August 2020. arXiv:2003.04618 [cs].

[33] Waseem Rawat and Zenghui Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9):2352–2449, September 2017.

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, January 2016. arXiv:1506.01497 [cs].

[35] Christoph B. Rist, David Emmerichs, Markus Enzweiler, and Dariu M. Gavrila. Semantic Scene Completion using Local Deep Implicit Functions on LiDAR Data, April 2021. arXiv:2011.09141 [cs].

[36] Luis Roldao, Raoul De Charette, and Anne Verroust-Blondet. LMSCNet: Lightweight Multiscale 3D Semantic Completion. In *2020 International Conference on 3D Vision (3DV)*, pages 111–119, Fukuoka, Japan, November 2020. IEEE.

[37] Luis Roldão, Raoul de Charette, and Anne Verroust-Blondet. 3D Semantic Scene Completion: A Survey. *International Journal of Computer Vision*, 130(8):1978–2005, August 2022.

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015. arXiv:1505.04597 [cs].

[39] Matthias Schreier. Data fusion for automated driving: An introduction. *at - Automatisierungstechnik*, 70(3):221–236, March 2022.

[40] Ajay Shrestha and Ausif Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7:53040–53065, 2019.

[41] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015. arXiv:1409.1556 [cs].

[42] Vishwanath A. Sindagi, Yin Zhou, and Oncel Tuzel. MVX-Net: Multimodal VoxelNet for 3D Object Detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282, Montreal, QC, Canada, May 2019. IEEE.

[43] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions, June 2020. arXiv:2006.09661 [cs, eess].

[44] Vincent Sitzmann. Awesome implicit representations - a curated list of resources on implicit neural representations. https://github.com/vsitzmann/awesome-implicit-representations, 2020. Accessed: 2023-01-21.

[45] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting, January 2023. arXiv:2301.00493 [cs].

[46] Zhaoyang Xia, Youquan Liu, Xin Li, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, and Yu Qiao. SCPNet: Semantic Scene Completion on Point Cloud, March 2023. arXiv:2303.06884 [cs].

[47] Aoran Xiao, Jiaxing Huang, Dayan Guan, Kaiwen Cui, Shijian Lu, and Ling Shao. PolarMix: A General Data Augmentation Technique for LiDAR Point Clouds, July 2022. arXiv:2208.00223 [cs].

[48] Xuemeng Yang, Hao Zou, Xin Kong, Tianxin Huang, Yong Liu, Wanlong Li, Feng Wen, and Hongbo Zhang. Semantic Segmentation-assisted Scene Completion for LiDAR Point Clouds. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3555–3562, Prague, Czech Republic, September 2021. IEEE.

[49] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. *Sensors*, 21(6):2140, March 2021.

[50] Wenwei Zhang, Zhe Wang, and Chen Change Loy. Multi-Modality Cut and Paste for 3D Object Detection.

[51] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection, November 2017. arXiv:1711.06396 [cs].

[52] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation, November 2020. arXiv:2011.10033 [cs].