

Multimodal Panoptic Segmentation of 3D Point Clouds

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

**genehmigte
Dissertation**

von

M.Sc.

Fabian Dürr

aus Rothenburg o. d. T.

Tag der mündlichen Prüfung:
Erster Gutachter:
Zweiter Gutachter:

22.06.2023
Prof. Dr.-Ing. Jürgen Beyerer
Prof. Dr.-Ing. J. Marius Zollner

Abstract

The human driver is one of the main reasons for traffic accidents. Therefore, autonomous driving or Advanced Driver Assistance Systems (ADAS) have the potential to reduce the number of human-related traffic accidents drastically by supporting or replacing the human driver. Driven by the potential impact of ADAS and autonomous driving, the understanding and interpretation of a vehicle's 3D environment has become increasingly important. Knowledge about drivable space, the position and motion of other traffic participants, such as cars or pedestrians, and the static environment is of utmost importance. Therefore, autonomous vehicles are usually equipped with a variety of sensors, such as camera, lidar, radar, and ultrasonic sensors. Lidar sensors and their recorded point clouds are particularly interesting for the challenge of 3D scene understanding since they provide accurate 3D information about the current environment. An essential task in this context is panoptic segmentation, which enhances every 3D point with semantic and instance information. However, the unstructured and sparse nature of 3D point clouds requires novel approaches and algorithms to achieve high quality and robust results. Established technologies from the 2D image domain, such as Convolutional Neural Networks, cannot be applied directly. Hence, the first key challenge is the representation of point clouds to effectively leverage the power of approaches based on deep learning. Alongside the chosen representation, the sequential nature of the recorded sensor data over time offers great potential as an additional modality to improve the 3D panoptic segmentation. Nevertheless, it is challenging to propagate information through time from a 3D point cloud to its successor since spatial correspondences must be found. Finally, different sensor modalities with distinct measurement principles offer further potential for enhancement. However, sensor fusion requires an efficient combination of diverse information from distinct sensor spaces. An even

greater and still unsolved challenge is a joint solution for these aspects and dense 3D tasks, such as semantic or panoptic segmentation.

The objective of this thesis is the design of a multimodal approach based on deep learning for 3D panoptic segmentation. It builds upon and combines the three key aspects multi view point cloud architecture, temporal feature fusion, and deep sensor fusion. The multi view architecture exploits multiple point cloud representations, also called *views*, to combine their strengths and compensate for individual weaknesses. The complementary 2D range view and bird's eye view are utilized for efficient context aggregation to support a high resolution point view. The point view combines multi view context while preserving fine details. Afterwards, a recurrent temporal fusion approach is introduced to exploit temporal dependencies by aggregating and propagating feature maps through time. It builds upon a temporal memory in range or bird's eye view containing the aggregated past information, which is updated in every step with the current information. A temporal alignment step compensates for the ego motion and ensures spatial consistency between frames. Furthermore, a novel deep sensor fusion approach combines lidar and camera features to enhance 3D panoptic segmentation. Motivated by the promising combination of image and depth information, camera and lidar feature maps are fused in the range view following either an iterative or pyramid-based fusion strategy. Finally, the individual contributions are combined into a novel multimodal multi view architecture that simultaneously exploits the proposed multi view, temporal, and sensor fusion frameworks.

Extensive experiments on the two large scale public datasets nuScenes and SemanticKITTI are conducted to investigate the benefits of the three main contributions and the combined multimodal framework. First, the evaluation shows the superiority of the multi view approach over single view methods. Next, it underlines the value of learning temporal dependencies by revealing significant improvements of the presented temporal framework over single frame baselines. Furthermore, it confirms the value of fusing distinct sensor features. Finally, excellent results are achieved by combining the presented approaches into a multimodal framework, which outperforms state-of-the-art results for various tasks and benchmarks.

Kurzfassung

Einer der Hauptgründe für Verkehrsunfälle ist oftmals der menschliche Fahrer. Fahrerassistenzsysteme und autonomes Fahren haben deshalb das Potenzial, die Zahl der Verkehrsunfälle drastisch zu senken, indem sie den menschlichen Fahrer unterstützen oder ersetzen. Eine wichtige Voraussetzung dafür ist das zuverlässige Erfassen und Verstehen der 3D-Fahrzeugumgebung. Automatisierte Systeme benötigen Informationen über den Straßenverlauf, die statische Umgebung und über die Position und Geschwindigkeit anderer Verkehrsteilnehmer, wie Autos oder Fußgänger. Dafür sind autonome Fahrzeuge üblicherweise mit einer Vielzahl von Sensoren, wie beispielsweise Kamera-, Lidar-, Radar- und Ultraschallsensoren ausgestattet. Die von Lidarsensoren aufgezeichneten 3D-Punktwolken leisten dabei einen zentralen Beitrag zur 3D-Umfelderfassung, da diese präzise 3D-Informationen über die aktuelle Umgebung liefern. Eine elementare Aufgabe ist in diesem Kontext die panoptische Segmentierung, die jeden 3D-Punkt der Punktwolke einer semantischen Klasse und individuellen Objektinstanz zuordnet. Für diese Aufgabe sind aufgrund der ungeordneten und unregelmäßigen Struktur von 3D-Punktwolken jedoch neue Ansätze und Algorithmen erforderlich, um qualitativ hochwertige und robuste Ergebnisse zu erzielen. Etablierte Technologien aus der 2D-Bildverarbeitung, wie z.B. Convolutional Neural Networks, sind aufgrund der ungeordneten Struktur nicht direkt anwendbar. Eine zentrale Herausforderung ist deshalb 3D-Punktwolken geeignet zu repräsentieren, um Deep Learning Ansätze anwenden und deren Potenzial ausnutzen zu können. Neben der Repräsentation der Punktwolken bietet auch die zeitlich sequentielle Natur der Sensordaten ein erhebliches Potenzial zur Verbesserung der panoptischen Segmentierung. Eine große Herausforderung ist dabei jedoch, die aus einer Punktwolke extrahierten Informationen auf die zeitlich nachfolgende Punktwolke zu übertragen, da räumliche Korrespondenzen in 3D gefunden werden

müssen. Eine weitere vielversprechende Verbesserungsmöglichkeit bietet die Fusion verschiedener Sensormodalitäten mit unterschiedlichen Messprinzipien. Die Sensorfusion erfordert jedoch eine effiziente Kombination verschiedener Informationen aus unterschiedlichen Sensorräumen. Eine noch größere und ungelöste Herausforderung ist eine kombinierte Lösung, die all drei Aspekte vereint.

Das Ziel dieser Arbeit ist die Entwicklung eines multimodalen Ansatzes basierend auf Deep Learning für die panoptische Segmentierung von 3D Punktwolken. Die drei zentralen Aspekte des Ansatzes sind dabei eine Multi-View-Architektur für 3D-Punktwolken, die zeitliche Fusion von 3D Punktwolken sowie deren Fusion mit Kamerainformationen. Die Multi-View-Architektur vereint verschiedene Repräsentationen von 3D Punktwolken, auch Views genannt, um deren Stärken zu kombinieren und individuelle Schwächen zu kompensieren. Dabei kommen mit der Range View und Bird's Eye View zwei komplementäre 2D Repräsentationen für die effiziente Aggregation von Features und Kontext zum Einsatz. Beide unterstützen die dritte verwendete Repräsentation, die sogenannte Point View. Diese dient als Bindeglied, um die Features aller drei Repräsentationen für jeden 3D-Punkt zu fusionieren. Die Ausgabe des Point View Netzwerkes ist ein individueller Feature-Vektor für jeden 3D Punkt für dessen semantische Klassifikation. Anschließend wird ein rekurrenter zeitlicher Fusionsansatz vorgestellt, um zeitliche Abhängigkeiten zu lernen und auszunutzen. Dafür werden Feature Maps in Range und Bird's Eye View über die Zeit aggregiert, wodurch ein zeitliches Gedächtnis entsteht. Dieses enthält durch die rekursive Aggregation die Informationen der vorangehenden Punktwolken und wird in jedem Schritt mit den aktuellen Informationen aktualisiert. Ein Transformationsschritt kompensiert die Eigenbewegung des Fahrzeuges und gewährleistet die räumliche Konsistenz des Gedächtnisses über die Zeit. Im nachfolgenden Schritt wird ein Sensorfusions-Ansatz für die Fusion von Lidar- und Kamerainformationen vorgestellt, um die panoptische Segmentierung weiter zu verbessern. Motiviert durch die erfolgreiche Kombination von RGB- und Tiefeninformationen werden Kamera- und Lidar-Feature Maps in der Range View kombiniert. Dafür kommt entweder eine iterative oder Pyramiden-basierte Fusionsstrategie zum Einsatz.

Am Ende werden die einzelnen Beiträge zu einer multimodalen Multi-View-Architektur kombiniert, die als erster Ansatz die Vorteile einer Multi-View-Architektur, zeitlichen Fusion und Sensorfusion für die 3D panoptische Segmentierung vereint.

Für die Evaluation werden vielfältige Experimente auf den beiden umfangreichen und öffentlichen Datensätzen nuScenes und SemanticKITTI durchgeführt. Dabei werden die Vorteile der einzelnen vorgestellten Ansätze sowie des kombinierten multimodalen Ansatzes untersucht. Die Experimente zeigen im ersten Schritt die Überlegenheit des Multi-View-Ansatzes gegenüber Single-View-Methoden. Zusätzlich wird der Wert gelernter zeitlicher Abhängigkeiten unterstrichen, da die durchgeführten Experimente signifikante Verbesserungen als Ergebnis der zeitlichen Fusion zeigen. Weiterhin bestätigen die Experimente wesentliche Verbesserungen der panoptischen Segmentierung durch die vorgeschlagene Fusion von Lidar- und Kamerainformationen. Als Gesamtergebnis der Arbeit erzielt die Kombination der vorgestellten Ansätze zu einem multimodalen Ansatz hervorragende Ergebnisse und verbessert den Stand der Technik für verschiedene Arten der Segmentierung.

Contents

Notation	xi
1 Multimodal Scene Understanding with 3D Point Clouds	1
1.1 Motivation	1
1.2 Challenges	3
1.3 Contributions	5
2 Related Work	7
2.1 Deep Neural Networks	8
2.1.1 Multi-Layer Perceptron	8
2.1.2 Convolutional Neural Networks	11
2.1.3 Recurrent Neural Networks	15
2.2 2D Scene Understanding	17
2.2.1 Semantic Segmentation	18
2.2.2 Panoptic Segmentation	22
2.3 3D Scene Understanding	25
2.3.1 Point Cloud Representations	26
2.3.2 Semantic Segmentation	31
2.3.3 Instance Segmentation	41
2.3.4 Panoptic Segmentation	44
2.4 Temporal Point Cloud Fusion	47
2.5 Sensor Fusion for Point Clouds	51
3 Concept	53
3.1 Multi View Architecture	54
3.2 Temporal Multi View Architecture	58
3.3 Multimodal Multi View Architecture	61

3.4	Multimodal Feature Map Transformation	64
4	Multi View Panoptic Segmentation of 3D Point Clouds	71
4.1	Range View Network	71
4.2	Bird’s Eye View Network	77
4.3	Multi View Network	82
5	Temporal Panoptic Segmentation of 3D Point Clouds	89
5.1	Temporal Range View	89
5.1.1	Temporal Memory Alignment	91
5.1.2	Temporal Memory Update	95
5.1.3	Temporal Training	97
5.2	Temporal Bird’s Eye View	98
5.3	Temporal Multi View Network	101
6	Multimodal Panoptic Segmentation of 3D Point Clouds	103
6.1	Multi Sensor Range View	103
6.1.1	Sensor Fusion	105
6.1.2	Training Strategy	108
6.2	Multimodal Multi View Network	109
7	Evaluation	111
7.1	Experimental Setup	111
7.1.1	Datasets	112
7.1.2	Metrics	116
7.1.3	General Implementation Details	120
7.2	Multi View Panoptic Segmentation	122
7.2.1	Range View Experiments	122
7.2.2	Bird’s Eye View Experiments	124
7.2.3	Multi View Experiments	126
7.3	Temporal Panoptic Segmentation	132
7.3.1	Temporal Range View Experiments	133
7.3.2	Temporal Bird’s Eye View Experiments	142
7.3.3	Comparison to State-of-the-Art	144
7.4	Multi Sensor Panoptic Segmentation	146

7.4.1	Range View Fusion Experiments	147
7.4.2	Comparison to State-of-the-Art	152
7.5	Multimodal Multi View Panoptic Segmentation	154
7.5.1	Multimodal Experiments	155
7.5.2	Comparison to State-of-the-Art	164
8	Discussion and Outlook	169
8.1	Discussion	170
8.2	Outlook	176
	Bibliography	179
	Publications	209
	Acronyms	211

Notation

This chapter introduces the notation and symbols that are used in this thesis.

General Notation

Scalars	italic Roman and Greek letters	x, H, α
Vectors	bold Roman lowercase letters	f
Matrices and Tensors	bold Roman uppercase letters	F
Functions	calligraphic Roman uppercase letters	\mathcal{F}
Sets	bold calligraphic Roman uppercase letters	\mathcal{P}

Scalars

b	bias
c	channel index
c_0	speed of light under vacuum conditions
d	dimension index
e	Euler's number
i	training iteration
$k \times k$	kernel size
l	neural network layer index
n	point index
r	distance
r_{\min}, r_{\max}	lower and upper bound of distance field of view

$r_{\text{fov}}, \theta_{\text{fov}}, \phi_{\text{fov}}$	spherical field of view
S_H, S_W	vertical and horizontal stride
t, τ	discrete points in time
u, v, w	grid coordinates
x, y, z	Cartesian coordinates
Z_{fov}	vertical field of view for cylindrical coordinates
$Z_{\text{min}}, Z_{\text{max}}$	lower and upper bound of vertical field of view
α	significance level
β	negative slope of Leaky Rectified Linear Units
ζ	sequence length
η	learning rate
θ	polar angle
ι	lidar intensity
$\theta_{\text{down}}, \theta_{\text{up}}$	lower and upper bound of vertical field of view
λ	loss weight
ν	view
ϕ	azimuth angle
$\phi_{\text{min}}, \phi_{\text{max}}$	lower and upper bound of horizontal field of view
ξ	temporal alignment discount factor
ψ	momentum factor
ω	perceptron weight
B	number of residual blocks
C	number of kernels of a convolutional layer
H, W, D	tensor dimensions
L	loss value
M	number of perceptrons
N	number of elements

N_{classes}	number of semantic classes
Q	number of probes
S	sensor
acc	accuracy
IoU	intersection-over-union
$mIoU$	mean intersection-over-union
PQ	panoptic quality
mPQ	mean panoptic quality
RQ	recognition quality
mRQ	mean recognition quality
SQ	segmentation quality
mSQ	mean segmentation quality

Vectors

$\mathbf{1}$	all ones vector
\mathbf{b}	bias vector
\mathbf{f}	feature vector
$\tilde{\mathbf{f}}$	network layer activations
$\tilde{\mathbf{f}}^{\text{BN}}$	normalized network layer activations
\mathbf{g}	Cartesian position of grid cell
\mathbf{h}	hidden state vector
$\tilde{\mathbf{h}}$	candidate state of a Gated Recurrent Unit
\mathbf{o}	2D offset vector
\mathbf{p}	Cartesian 3D point
$\tilde{\mathbf{p}}$	spherical 3D point
$\tilde{\mathbf{p}}^z$	cylindrical 3D point
\mathbf{r}	reset vector of a Gated Recurrent Unit

\mathbf{u}	2D or 3D grid coordinate
\mathbf{x}	perceptron input
\mathbf{y}	neural network output
\mathbf{z}	update vector of a Gated Recurrent Unit
γ	view weight vector
$\mu_{\mathcal{B}}$	channel-wise mean over batch
$\sigma_{\mathcal{B}}^2$	channel-wise variance over batch
ω	perceptron weight vector
$\omega_{\text{BN}}, \mathbf{b}_{\text{BN}}$	Batch Normalization learnable parameters

Matrices and Tensors

\mathbf{F}	feature map
\mathbf{G}	Cartesian position of grid cells
\mathbf{H}	hidden state
\mathbf{O}	offset tensor
\mathbf{P}	point cloud
\mathbf{S}	semantic class scores
\mathbf{T}	homogeneous transformation matrix
\mathbf{U}	projection index matrix
\mathbf{V}	projection index tensor
\mathbf{W}	weight matrix of neural network layer

Functions

\mathcal{F}	multi view fusion
\mathcal{A}	activation function
\mathcal{C}	center assignment function

\mathcal{E}	bijjective ordering
\mathcal{L}	loss function
\mathcal{P}	point cloud projection
Q^θ	Cartesian to polar conversion
Q^z	Cartesian to cylindrical conversion
$\mathcal{S}, \hat{\mathcal{S}}$	spatial feature map transformation
\mathcal{T}	combined transformation and projection
\mathcal{H}	temporal memory update

Sets

\mathcal{B}	batch
\mathcal{P}	point cloud as set of points
\mathcal{U}	valid matrix index matrices
\mathcal{V}	valid matrix index tensors
\mathcal{K}	instance center candidates
\mathcal{C}	semantic classes
TP	true positives
FP	false positives
FN	false negatives
TPI	true positive instances
FPI	false positive instances
FNI	false negative instances

1 Multimodal Scene Understanding with 3D Point Clouds

1.1 Motivation

Advanced Driver Assistance Systems (ADAS) and autonomous driving are among the most impactful and disruptive technologies in the automotive industry and beyond. The purpose of these systems is to make driving safer and more convenient. According to the National Highway Traffic Safety Administration (NHTSA), the human driver is the critical reason for approximately 94% of traffic accidents in the United States [Sin15]. Therefore, ADAS and autonomous driving have the potential to support or replace the driver and to reduce the number of human-related accidents drastically. The capabilities of these systems are characterized by two key properties. The first one is the level of autonomy, which determines if the system takes over full responsibility or must be supervised by the human driver. The second one is the operational design domain, which specifies the use cases the system can handle. Hence, five levels have been proposed [SAE18] to classify these systems, starting from level one with limited functionality and restricted use cases while the entire responsibility remains with the driver. On the other hand, systems of the fifth level provide full autonomy for any use case and take over full responsibility. The increasing amount of autonomy, functionality, and variety in the covered use cases impose tremendous requirements on the autonomous system and especially its environmental perception. Therefore, without a robust and comprehensive understanding of its environment, an autonomous vehicle can neither fulfill its purpose nor drive at all safely.

Consequently, the understanding and interpretation of a vehicle’s 3D environment has become increasingly important. The foundation is the sensor set of an autonomous vehicle, which usually comprises a variety of different sensors, such as camera, lidar, radar, and ultrasonic sensors. Based on measurements provided by these sensors, a comprehensive and unified environment model must be predicted to provide a holistic understanding of a vehicle’s current 3D environment. Among others, this includes knowledge about drivable space, the static world, or the position and motion of other traffic participants, such as cars or pedestrians. Therefore, the different and complementary sensors are combined by sensor fusion to create the required unified environment model and to compensate for the shortcomings of individual sensor types. In addition, sensors in the context of autonomous driving record their environment sequentially, and previous recordings contain valuable information also for the current time step. Hence, a temporal fusion of current and past information has the potential to improve the environment model.

The individual sensor types provide different data, such as camera images or lidar point clouds. Lidar sensors and their recorded point clouds are particularly interesting for 3D scene understanding or environment models since they provide accurate 3D information. Various tasks can be solved based on these 3D points to provide valuable information for autonomous driving, such as object detection and semantic or panoptic segmentation. The latter is an important and complex task, depicted in Fig. 1.1, that enhances every 3D point with semantic and instance information. Semantic information describes the object class, whereas instance information allows distinguishing between individual instances of a semantic class. Hence, 3D panoptic segmentation provides a valuable combination of geometric, semantic, and instance knowledge.

Motivated by the high value of 3D panoptic segmentation for environment perception, this thesis focuses on solving this task for lidar point clouds. Methods based on deep learning achieve excellent results for scene understanding tasks in the image domain, such as panoptic segmentation. Therefore, the proposed approach builds upon deep learning to leverage its power for 3D point clouds. Furthermore, this thesis focuses on a multimodal approach capable of performing sensor and temporal fusion. It predicts panoptic segmentation

for lidar point clouds while additionally exploiting camera and temporal information.

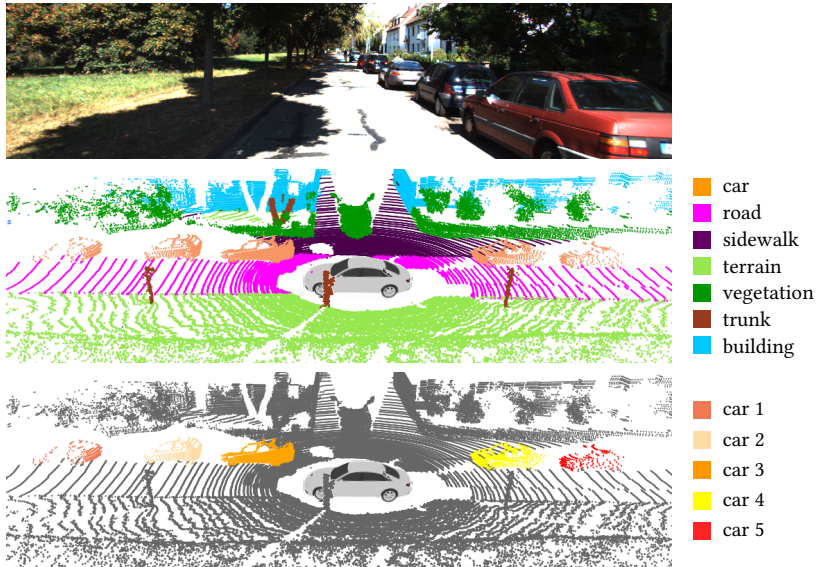


Figure 1.1: The task of 3D panoptic segmentation. It is composed of semantic and instance segmentation subtasks, shown in the middle and at the bottom, respectively. Different colors visualize the semantic classes and individual instances.

1.2 Challenges

Panoptic segmentation of 3D point clouds is a challenging task due to the irregular and sparse nature of point clouds and the necessity to distinguish simultaneously between a significant number of semantic classes and their instances. Additionally, existing training data is considerably unbalanced with respect to the semantic classes and their instances. Furthermore, approaches that include sensor and temporal fusion face additional challenges related to calibration, ego motion, and temporal synchronization.

The first set of challenges originates from the particular properties of lidar point clouds. These point clouds

- do not provide an intrinsic ordering but are an **unordered set** of points,
- are **sparse** and often provide only some measurement points on individual objects and structures, and
- have a **spatially varying density**, which decreases with increasing distance to the sensor.

These properties complicate the computation and exploitation of relations between individual points and the capturing of local structures. As a result, the hierarchical aggregation of information required to understand objects and structures sparsely represented in a point cloud is challenging. Furthermore, the unordered nature prevents the direct application of established deep learning architectures from the image domain. These require their input data to be organized as a grid and face additional challenges related to the training data:

- The semantic subtask significantly and systematically suffers from **class imbalance**. The number of points for small classes, such as pedestrians, is orders of magnitude smaller than for dominating classes, such as road, building, or vegetation. In addition, only a few unique instances of rare semantic classes exist, which makes learning the underlying concept difficult.
- Pointwise labeling of point clouds is time-consuming. Therefore, existing datasets are **small** compared to other tasks and domains, e.g., image classification.
- Pointwise labeling of point clouds is difficult. Hence, the ground truth of existing datasets usually contains a significant amount of **labeling errors**. Examples are wrong semantic classes, inaccurate boundaries between semantic classes or instances, and missing instances.
- No huge dataset similar to ImageNet [Rus15] exists for pre-training in the lidar domain. As a result, models must be **trained from scratch**, even for complex tasks such as panoptic segmentation.

Independently of the considered panoptic segmentation task and deep learning, more challenges arise when temporal and sensor fusion are considered:

- Errors in sensor calibrations and ego poses complicate the transformation of different sensor data from potentially different time steps into a unified space for fusion.
- Different sensor types usually record at distinct points in time and with different frame rates. Consequently, the perceived environment differs due to moving instances and ego motion.
- The individual sensors' Fields of View (FoVs) often overlap only partially, limiting the potential of sensor fusion.

1.3 Contributions

The objective of this thesis is the design of a multimodal architecture based on deep learning for robust and high quality panoptic segmentation of 3D point clouds. The proposed framework builds upon three main concepts: a multi view point cloud architecture, a temporal feature fusion, and a deep sensor fusion. The multi view architecture relies on different point cloud representations, also called *views*, to exploit their strengths and compensate for weaknesses. A recurrent temporal feature fusion considers information from previous time steps to exploit temporal dependencies. Finally, deep sensor fusion exploits cameras as an additional sensor modality to improve the 3D panoptic segmentation. The evaluation is performed on two challenging and large scale outdoor datasets, where the individual contributions and their combination outperform state-of-the-art results for various tasks. The contributions of this thesis are in detail:

- A novel multi view framework [Due22] addresses the shortcomings of single view approaches and individual views. It is based on range view, bird's eye view, and point view and obtains significantly improved features compared to single view approaches. Range and bird's eye view provide efficient context aggregation, while the high resolution point

view maintains a unique feature vector for every 3D point. Due to the carefully chosen views, the introduced approach considerably reduces the computational complexity compared to established multi view approaches. The framework also includes an enhanced multi view, multi task strategy. The point view provides the 3D semantic segmentation, whereas the bird's eye view is used for center-based instance recognition, required for instance segmentation.

- A temporal fusion framework [Due20a] with novel recurrent feature map fusion in range and bird's eye view considerably improves the 3D panoptic segmentation in both views. It provides enhanced features based on a temporal memory containing aggregated past information. The memory is recursively updated in every step with the current information. Two novel temporal alignment strategies compensate for the ego motion and ensure spatial consistency between two time steps. The alignment enables the recurrent architecture to reuse past computations and decouples runtime from the number of considered past steps. Consequently, and in contrast to existing methods, the temporal memory can propagate information over sequences of arbitrary length.
- A multi sensor framework [Due20b, Due21, Sch22a] with novel deep feature fusion for lidar and camera information in the range view. Two multi scale fusion strategies are proposed, following either an iterative or pyramid-based pattern to improve the feature map fusion. Furthermore, these strategies can counteract camera failure and mitigate its impact.
- The first multimodal multi view architecture that successfully combines and leverages the potential of a multi view architecture, temporal feature fusion, and deep sensor fusion. All three contributions enhance the results as part of the proposed unified architecture, which outperforms state-of-the-art methods for different tasks and datasets. Additionally, several combined frameworks for different use cases and based on two of the three contributions are presented. These provide a significantly enhanced 3D panoptic segmentation compared to approaches that exploit only one of these aspects.

2 Related Work

Panoptic segmentation [Kir19] is the combined task of semantic and instance segmentation, which provides semantic and object information about the environment. The subtask of semantic segmentation assigns one of the predefined semantic classes to every image pixel or 3D point. Instance segmentation, on the other hand, clusters pixels or points into instances. However, distinguishing between instances is only possible and useful for some semantic classes. Foreground or “thing” classes are countable classes that require instance segmentation, specifically traffic participants, such as car, bicyclist, or pedestrian. On the other hand, background or “stuff” classes are uncountable, such as road and sidewalk, or their instances are irrelevant for the considered scenario, such as buildings or poles. Therefore, instance segmentation is only provided for the subset of thing classes, which are determined by the semantic segmentation. Overall, panoptic segmentation simultaneously requires a high quality semantic segmentation and sophisticated instance recognition for convincing panoptic results. Furthermore, it requires mutually consistent predictions for both subtasks instead of an independent and trivial combination of both.

The overall goal of this thesis is a multimodal, deep learning-based approach for panoptic segmentation of 3D point clouds. It builds upon the foundations of deep neural networks and their success in 2D scene understanding. Related work is investigated in the areas of 3D semantic, instance, and panoptic segmentation, as well as deep learning-based temporal and sensor fusion.

2.1 Deep Neural Networks

While the mathematical foundations and basic unit have already been proposed in the middle of the last century [Ros57], the path from this early and simple linear classifier to a deep neural network that exceeds human-level performance in image classification [Rus15, He15b] took more than half a century. Neural networks in computer vision experienced their renaissance with the growing computational power of GPUs [Kri12], which were able to optimize these networks in reasonable time. In the following years, tremendous progress and improvements have been achieved across various computer vision tasks, and in other areas, such as neural language processing. The following section summarizes the fundamentals of neural networks and specialized network architectures, such as convolutional and recurrent neural networks. Further theoretical and mathematical details can be found in [Dud00, Bis06]. While x and y describe Cartesian coordinates throughout this thesis, they are used as scalar components of input and output vectors \mathbf{x} and \mathbf{y} in this section to follow established conventions [Dud00, Bis06].

2.1.1 Multi-Layer Perceptron

The basic building blocks of neural networks are individual neurons, called perceptrons [Ros57], and their structure is depicted in Fig. 2.1. Perceptrons take M scalar input values and produce a scalar output y . For that reason, the input vector $\mathbf{x} \in \mathbb{R}^M$ is multiplied with a weight vector $\boldsymbol{\omega} \in \mathbb{R}^M$ and a scalar bias b is added. Finally, an activation function $\mathcal{A} : \mathbb{R} \rightarrow \mathbb{R}$ is applied to produce the output:

$$y = \mathcal{A}(\boldsymbol{\omega}^T \mathbf{x} + b). \quad (2.1)$$

Common choices for the activation function of Convolutional Neural Networks (CNNs) are nowadays Rectified Linear Units (ReLUs) [Jar09, Nai10]

$$\mathcal{A}(x) = \max(x, 0) \quad (2.2)$$

and their leaky counterpart, Leaky Rectified Linear Units (LReLU) [Maa13]

$$\mathcal{A}(x) = \max(x, 0) + \beta \cdot \min(x, 0), \quad (2.3)$$

where β is a small number, such as 0.01. Alternatively, the sigmoid or hyperbolic tangent activation functions are other possible choices.

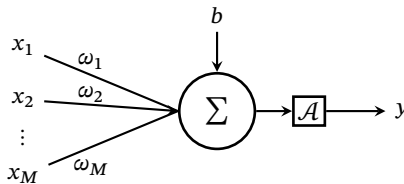


Figure 2.1: Structure of a perceptron.

A Multi-Layer Perceptron (MLP) combines multiple perceptrons to solve complex and nonlinear problems. The perceptrons are grouped into N layers and only connected to perceptrons of the previous and next layer, illustrated in Fig. 2.2. No connections exist inside a layer, and outputs are only provided to the next layer, which prohibits feedback loops to previous layers. The layer l with M_l perceptrons receives the output or feature vector $\mathbf{f}_{l-1} \in \mathbb{R}^{M_{l-1}}$ of the previous layer, representing the scalar outputs of M_{l-1} perceptrons. A weight matrix $\mathbf{W}_l \in \mathbb{R}^{M_l \times M_{l-1}}$ and bias vector $\mathbf{b}_l \in \mathbb{R}^{M_l}$ contain the weight vector and scalar bias of every perceptron in the l -th layer. This leads to the overall equation

$$\mathbf{f}_l = \mathcal{A}(\mathbf{W}_l \mathbf{f}_{l-1} + \mathbf{b}_l), \quad (2.4)$$

where $\mathbf{f}_0 = \mathbf{x}$ is the input, like an image, and $\mathbf{f}_N = \mathbf{y} \in \mathbb{R}^{M_N}$ is the output of an MLP with N layers.

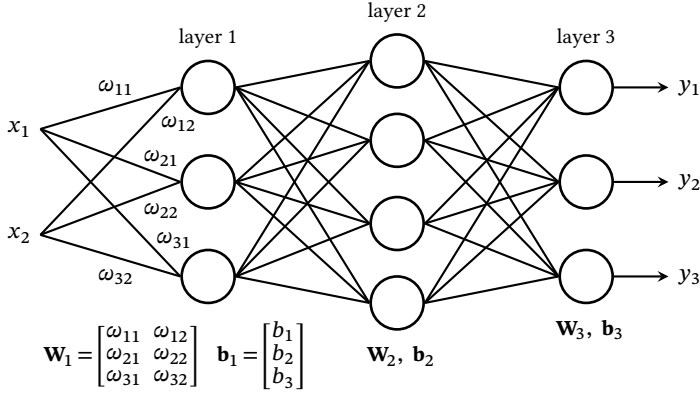


Figure 2.2: Multi-Layer Perceptron (MLP) composed of individual perceptrons.

Weight matrices and bias vectors, in the following summarized as \mathbf{W} , are the parameters of an MLP, which are optimized during training. The usual supervised training strategy requires pairs of input and ground truth output data $(\mathbf{x}, \mathbf{y}^{\text{gt}})$ for which the network computes its output \mathbf{y}^{x} . A task-dependent loss function $\mathcal{L}(\mathbf{y}^{\text{x}}, \mathbf{y}^{\text{gt}})$ measures their alignment. Common loss functions for regression are mean-squared error (MSE) and mean-absolute error (MAE):

$$\mathcal{L}_{\text{MSE}}(\mathbf{y}^{\text{x}}, \mathbf{y}^{\text{gt}}) = \frac{1}{M_N} \|\mathbf{y}^{\text{x}} - \mathbf{y}^{\text{gt}}\|_2^2 = L_{\text{MSE}}, \quad (2.5)$$

$$\mathcal{L}_{\text{MAE}}(\mathbf{y}^{\text{x}}, \mathbf{y}^{\text{gt}}) = \frac{1}{M_N} \|\mathbf{y}^{\text{x}} - \mathbf{y}^{\text{gt}}\|_1 = L_{\text{MAE}}.$$

For classification, the output vector \mathbf{y} contains the individual class scores, and its dimension M_N matches the number of classes N_{classes} . In this case, Cross-Entropy (CE) loss is commonly used:

$$\mathcal{L}_{\text{CE}}(\mathbf{y}^{\text{x}}, \mathbf{y}^{\text{gt}}) = - \sum_{cls=1}^{N_{\text{classes}}} \log \left(\frac{e^{y_{cls}}}{\sum_{cls'=1}^{N_{\text{classes}}} e^{y_{cls'}}} \right) \cdot y_{cls}^{\text{gt}} = L_{\text{CE}}. \quad (2.6)$$

Backpropagation [Rum86] computes the gradient for the loss with respect to the network parameters and is the foundation for the optimization process.

The commonly used stochastic gradient descent computes in every iteration i the gradient over a subset of the training set, called batch \mathcal{B}_i , and applies a given learning rate η :

$$\Delta \mathbf{W}_i = -\eta \cdot \sum_{(\mathbf{x}, \mathbf{y}^{\text{gt}}) \in \mathcal{B}_i} \frac{\partial \mathcal{L}(\mathbf{y}^{\text{x}}, \mathbf{y}^{\text{gt}})}{\partial \mathbf{W}}. \quad (2.7)$$

Afterwards, the parameters are updated accordingly:

$$\mathbf{W}_{i+1} = \mathbf{W}_i + \Delta \mathbf{W}_i. \quad (2.8)$$

Momentum extends this update step with a portion ψ of the previous weight update to overcome plateaus in the loss function and converge more quickly:

$$\begin{aligned} \Delta \mathbf{W}'_i &= \Delta \mathbf{W}_i + \psi \Delta \mathbf{W}'_{i-1}, \\ \mathbf{W}_{i+1} &= \mathbf{W}_i + \Delta \mathbf{W}'_i. \end{aligned} \quad (2.9)$$

2.1.2 Convolutional Neural Networks

The previously presented layers of an MLP are called fully connected because every perceptron of one layer is connected to every perceptron of the next layer. As a result, the number of connections and associated weights increases quadratically with the number of perceptrons. This is especially challenging for high dimensional inputs, like images, with potentially millions of pixels. Convolutional layers address this challenge for data with a grid-like topology by sparsity and parameter sharing. The sparsity is achieved by restricting connections to nearby perceptrons, illustrated in Fig. 2.3, based on the assumption of strong local structure and correlation [Lec98]. This locally connected region is the receptive field of a convolutional layer. Additionally, the weights are shared among all perceptrons of one layer to regularize the parameters and further reduce their number. Therefore, the weights are independent of a perceptron's position and grant spatial invariance. As a result, the operation of a convolutional layer can be considered as applying filter kernels with the size of the receptive field to its input.

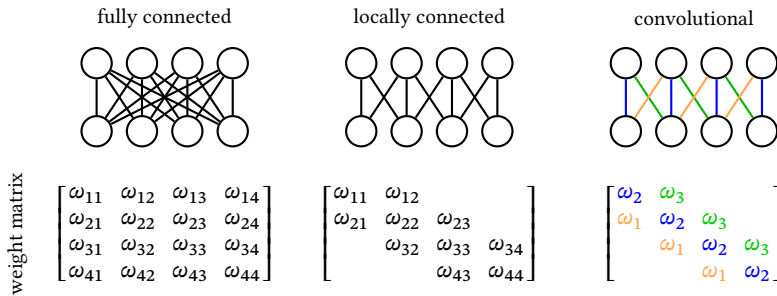


Figure 2.3: Comparison of a fully connected and 1D convolutional layer. Illustration based on [Her18].

Convolutional layers are the main building blocks of CNNs, together with pooling and fully connected layers [Lec98]. The first part of CNNs, made of convolutional and pooling layers, is called feature extractor or backbone and is responsible for extracting meaningful features from the network input. Convolutional layers apply C distinct and learnable $k \times k$ kernels to their input to compute C so-called feature maps, depicted in Fig. 2.4. These feature maps contain the extracted features, such as edges or corners when considering image input. Pooling layers, on the other hand, subsample feature maps by applying a reduction operation to local regions, e.g., of size 2×2 or 3×3 . Most commonly, the maximum is taken from each region, which has shown better results than taking the average [Sch10]. In order to achieve the desired subsampling, pooling operations are usually applied with a stride of two. As a result, the pooling operation is only applied to every other spatial location, which halves the feature map resolution. A common alternative to reduce the feature map resolution are convolutional layers with a stride of two or higher [Spr15]. The downsampling aggregates features and reduces the sensitivity of the output to shifts and distortions [Lec98]. Most importantly, repeated downsampling allows subsequent layers to extract higher-order features with increasing abstraction levels.

The basic setup of a CNN are alternating convolutional and pooling layers in the feature extractor followed by fully-connected layers in the second part, which is often referred to as head. It computes the task-specific final output,

such as a classification vector, based on the feature maps from the feature extractor.

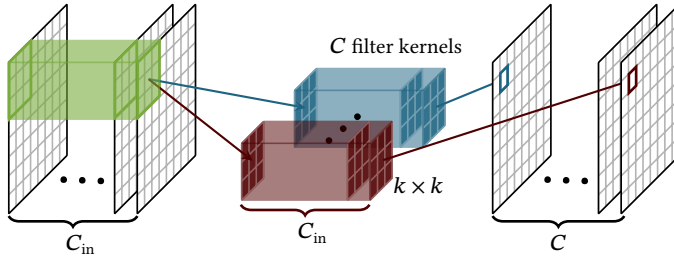


Figure 2.4: Operating principle of a convolutional layer. C learnable filter kernels are applied to C_{in} input feature maps and produce C new feature maps.

Deep Convolutional Neural Networks

Over time, state-of-the-art network architectures have become more complex, which is reflected in an increasing number of layers and has shaped the term deep learning [Sim14]. Starting with just a few layers for the task of document recognition [Lec98] and later image classification [Kri12], the number rose to 16 layers [Sim14] and afterwards up to 100 layers and beyond [Sze15, He16]. This was mainly driven by the challenging classification task of ImageNet [Den09], where images have to be classified as one of 1,000 classes. However, training very deep neural networks comes with several challenges. In order to address the vanishing or exploding gradient problem [Ben94], the activation function and weight initialization must be carefully chosen. As a result, ReLU or LReLU are the established choices, and the initial layer weights are sampled from a Gaussian distribution with zero mean, and a variance based on layer size [Glo10] or based on layer size and activation function [He15b]. Furthermore, Batch Normalization (BN) [Iof15] has been proposed to normalize the outputs of convolutional layers before the activation function is applied. Looking at the 1D case, the layer activations $\tilde{\mathbf{f}}_l = \mathbf{W}_l \mathbf{f}_{l-1} + \mathbf{b}_l$ are normalized to zero mean and unit variance $\tilde{\mathbf{f}}_l^{\text{norm}}$ by mean $\mu_{\mathcal{B}}$ and variance $\sigma_{\mathcal{B}}^2$ computed over the current batch \mathcal{B} . The learnable parameters ω_{BN} and

\mathbf{b}_{BN} ensure that the normalization layer does not negatively impact the representational capabilities:

$$\tilde{\mathbf{f}}_l^{\text{BN}} = \omega_{\text{BN}} \circ \tilde{\mathbf{f}}_l^{\text{norm}} + \mathbf{b}_{\text{BN}}, \quad (2.10)$$

where \circ denotes the Hadamard product. Batch Normalization makes the network’s weight initialization more robust and allows higher learning rates for faster convergence. Since batches are usually only used for training, running mean and variance are stored during training and applied during inference.

Based on these techniques, “going deeper” [Sze15] has shown great success and surpassed human-level performance [He15b] on the classification task of ImageNet. However, experiments have shown that the improvement achieved by stacking more layers not only diminishes but turns into a negative impact beyond a certain number [Sri15, He15a]. This effect cannot be explained by overfitting because the training error increases as well [He15a]. Residual networks [He16] are motivated by the consideration that increasing the number of layers should not negatively impact the results since they can be turned into identity functions. Therefore, instead of directly learning a mapping from input to output, a residual mapping is proposed, which is implemented by an identity skip connection and illustrated in Fig. 2.5. This enables deeper networks with improved results and allows the successful training of over a thousand layers. The core elements are residual Basic Blocks (BBs) and residual Bottleneck Blocks (BoBs) with the characteristic skip connection. These are grouped into stages which contain all layers applied to one specific feature map resolution or scale. The stages start with a subsampling layer, such as maximum pooling or strided convolution. As illustrated in Fig. 2.5, residual networks are composed of five stages with a varying number of residual blocks, depending on the specific setup. Commonly used configurations are ResNet-34, ResNet-50, ResNet-101, and ResNet-152, where the number indicates the total number of layers. Their detailed configurations can be found in [He16]. These networks have been, and still are, the predominant feature extractor in state-of-the-art methods across many tasks.

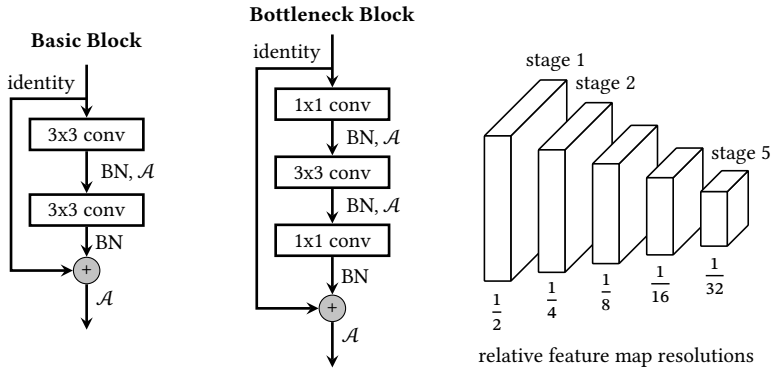


Figure 2.5: Building blocks and architecture of residual networks.

2.1.3 Recurrent Neural Networks

The huge success of neural networks is not limited to computer vision but influences other fields like natural language processing too. In this area, data is often arranged in sequences, like a sequence of words forming a sentence or text. As a result, neural architectures emerged, which are able to extract information along a sequence of data. When considering time, these principles are also fundamental for computer vision, especially in the context of 2D scene understanding, since it allows processing and exploiting video data instead of individual images.

In contrast to the feed forward networks discussed so far, Recurrent Neural Networks (RNNs) have a feedback loop, giving access to information from previous inputs. Figure 2.6 shows this recurrent loop, where the so-called hidden state \mathbf{h} loops back to the network input. In addition, Fig. 2.6 illustrates the unrolled network for a sequence of three inputs \mathbf{x}_0 , \mathbf{x}_1 , and \mathbf{x}_2 to visualize the repeated application of an RNN to the elements of a sequence and their relations more clearly. The recurrent network receives not only the input vector but also the hidden state vector with features from the processing of the previous input. As a result, the predictions \mathbf{y} are not only based on the corresponding input but also on the sequence context provided by the hidden state.

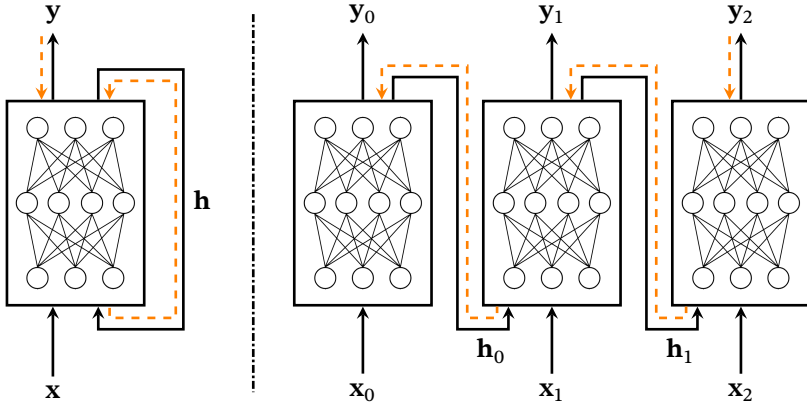


Figure 2.6: Recurrent Neural Network (RNN) with an unrolled example on the right. The gradient flow is indicated in orange.

As the unrolled architecture indicates, RNNs can also be interpreted as very deep neural networks, especially for long sequences. Therefore, they also suffer from exploding and vanishing gradients since the gradient must be propagated along the recurrent application, depicted as orange path in Fig. 2.6. One way to address this is to use gating mechanisms proposed by Long Short-Term Memory (LSTM) [Hoc97] and Gated Recurrent Units (GRUs) [Cho14]. Figure 2.7 exemplarily illustrates the latter with a focus on time series data, where the sequence index equals a discrete point in time t . Its output \mathbf{y}_t is computed by the following equations:

$$\begin{aligned}
 \mathbf{r}_t &= \text{sigmoid}(\mathbf{W}_{x,r} \mathbf{x}_t + \mathbf{W}_{h,r} \mathbf{h}_{t-1} + \mathbf{b}_r), \\
 \mathbf{z}_t &= \text{sigmoid}(\mathbf{W}_{x,z} \mathbf{x}_t + \mathbf{W}_{h,z} \mathbf{h}_{t-1} + \mathbf{b}_z), \\
 \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_{x,h} \mathbf{x}_t + \mathbf{W}_{h,h} (\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h), \\
 \mathbf{y}_t = \mathbf{h}_t &= (\mathbf{1} - \mathbf{z}_t) \circ \mathbf{h}_{t-1} + \mathbf{z}_t \circ \tilde{\mathbf{h}}_t.
 \end{aligned} \tag{2.11}$$

The reset gate computes the reset vector \mathbf{r}_t and decides which information from the previous state \mathbf{h}_{t-1} to forget and which to keep. The update vector \mathbf{z}_t on the other hand controls the element-wise combination of the previous

hidden state and the new candidate state $\tilde{\mathbf{h}}_t$. The latter is computed from the previous hidden state multiplied by the reset vector and the current input. All three gates are implemented based on a single-layer MLP with the respective weight matrices \mathbf{W} and bias vectors \mathbf{b} . When propagating the gradient along the processed sequence, it only has to pass the element-wise addition and multiplication, but not an entire MLP. This advantage also holds when the output of an MLP is provided to the GRU instead of the raw input \mathbf{x}_t . The gating mechanism counteracts vanishing or exploding gradients by reducing the number of layers the gradient passes when being backpropagated along the sequence. ConvGRU [Sia17] transfers this concept to the image domain, where input and hidden states are 2D feature maps instead of feature vectors, and the gates build upon convolutional layers instead of MLPs.

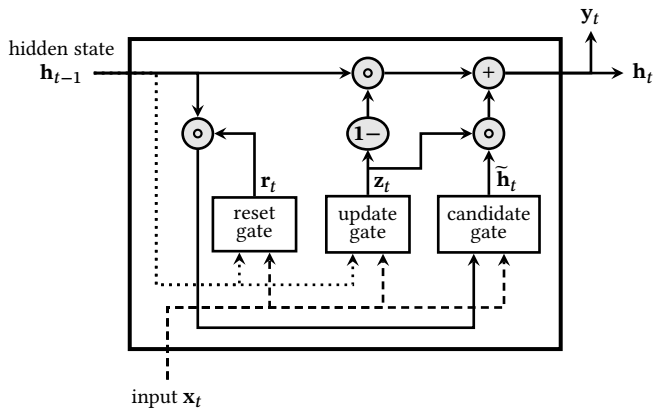


Figure 2.7: Structure of a Gated Recurrent Unit (GRU). It computes its output based on the previous hidden state \mathbf{h}_{t-1} and input \mathbf{x}_t using a reset, update, and candidate gate.

2.2 2D Scene Understanding

After the overwhelming success of CNNs in image classification [Rus15], they have been quickly deployed for other application areas, such as 2D scene understanding. The latter comprises, among others, the tasks of semantic and

instance segmentation, depicted in Fig. 2.8. This was also supported by the release of a constantly increasing number of semantic datasets [Bro09, Sil12, Lin14, Mot14, Cor16, Zho17, Yu20, Gey20] providing pixelwise semantic labels. Some of these datasets [Sil12, Lin14, Cor16, Zho17, Yu20] additionally provide pixelwise instance labels and enable the development of panoptic segmentation approaches. Since scene understanding is one of the key challenges of autonomous driving, many of these datasets belong to the outdoor driving domain [Bro09, Cor16, Yu20, Gey20]. In general, 2D scene understanding is a huge field with countless published work. The following section focuses on pioneer work and approaches that influenced this thesis or are directly used.



Figure 2.8: Semantic and instance segmentation, two tasks of 2D scene understanding [Cor16].

2.2.1 Semantic Segmentation

The task of assigning a semantic class to every pixel of an image is called semantic segmentation. One fundamental concept for approaches based on CNNs are Fully Convolutional Networks (FCNs) [Lon15], which compute a pixelwise prediction for a given input image in an end-to-end fashion. This requires a new architecture for the network head because a classification vector is required for every individual pixel instead of one for the entire image. Therefore, a 1×1 -convolutional layer replaces the standard fully-connected classification layer to generate a prediction for every pixel. However, the output of a CNN’s feature extractor usually has a considerably smaller resolution than its input due to pooling or convolutions with a stride greater one. As a result, the low resolution predictions provided by the 1×1 -convolutional layer must be upsampled again. Different FCN architectures are proposed, with a

single or multiple upsampling steps. Alongside bilinear interpolation, a new layer called deconvolution or transposed convolution learns the upsampling instead of applying a fixed one. The fully convolutional network with three upsampling steps (FCN-8s) is visualized in Fig. 2.9.

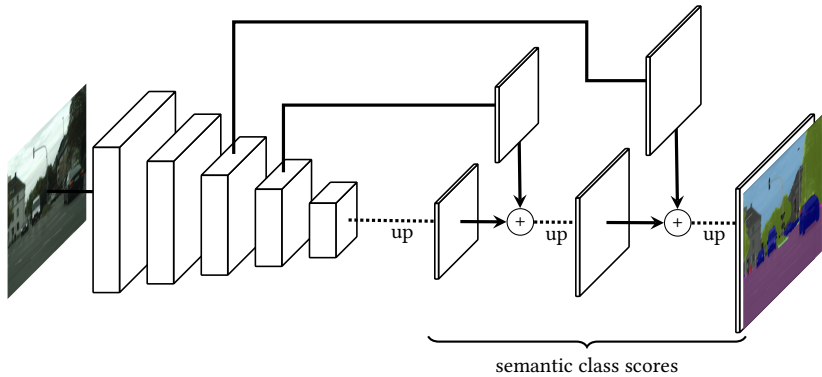


Figure 2.9: Architecture of Fully Convolutional Networks (FCNs). The low-resolution predictions are upsampled in three steps by transposed convolutions.

U-Net [Ron15] enhances FCNs by improving the upsampling processes. The number of upsampling steps is matched to the number of downsampling steps of the feature extractor. For every upsampling step, the corresponding feature maps with matching resolution are concatenated to improve spatial feature propagation, see Fig. 2.10. In contrast to native FCNs, U-Net upsamples the feature maps and not predictions. Overall, the downsampling and upsampling paths form a U-shaped architecture and are often called encoder and decoder, respectively. U-Net started to address one of the main challenges introduced by the pixelwise semantic segmentation task, which makes it necessary to simultaneously capture the global context of a scene and fine details. The former requires large receptive fields and is usually achieved by iteratively reducing the feature map resolution while dropping spatial information. However, the loss of spatial information negatively affects the capturing of fine details since information about the features' exact location is lost.

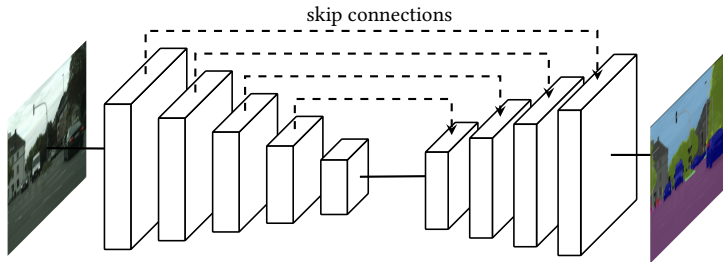


Figure 2.10: U-Net architecture for 2D semantic segmentation.

PSPNet [Zha17] addresses this challenge by introducing a pyramid pooling module. Different pyramid levels divide the feature maps into different-sized subregions and compute an aggregated representation for each region using pooling, depicted in Fig. 2.11. Aggregated context varies from local to global dependent on the subregion size. The outputs of the different pyramid levels are upsampled and concatenated with the original feature maps. As a result, the final classification layer is provided with feature maps containing local and global context at different scales, illustrated in Fig. 2.11, which significantly improves the segmentation results.

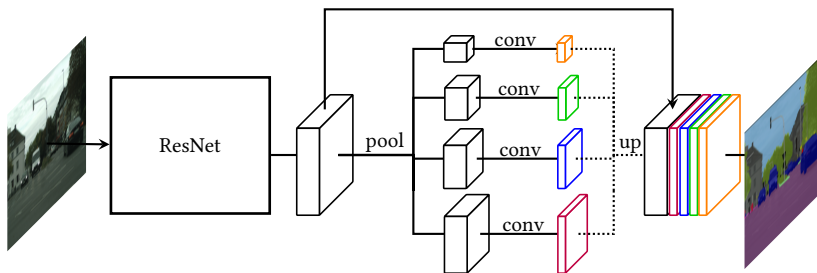


Figure 2.11: PSPNet with its pyramid pooling approach, illustration based on [Zha17].

The DeepLab family [Che18, Che17a] relies on atrous convolutions to increase the receptive fields' size without reducing the feature map resolution or increasing filter sizes. Additionally, Atrous Spatial Pyramid Pooling (ASPP) is

implemented by deploying atrous convolutions at different rates in parallel to exploit context at different scales. This pyramid pooling is similar to PSPNet but with atrous convolutions instead of pooling operations. Its goal is again the aggregation of multi scale context.

Deep Layer Aggregation (DLA) [Yu18] replaces simple skip connections with an enhanced aggregation architecture, as depicted in Fig. 2.12. Two neighboring stages of the feature extractor provide their feature maps to an aggregation node, which combines and compresses its inputs. This requires an upfront up-sampling of the lower-resolution feature maps. The compression is achieved by ensuring that the output channel size matches the channel size of a single input. The aggregation nodes are stacked in a tree-like fashion, and each iterative deep aggregation path, which replaces a conventional skip connection, aggregates features from shallow to deep.

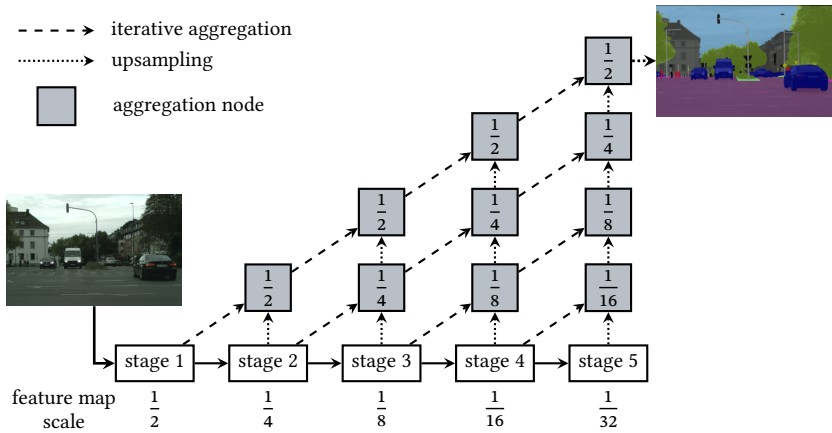


Figure 2.12: Structure of Deep Layer Aggregation (DLA), illustration based on [Yu18].

2.2.2 Panoptic Segmentation

For a long time, semantic and instance segmentation have been approached individually and were considered separate tasks. Kirillov et al. [Kir19] proposed panoptic segmentation, which unifies both tasks and requires a semantic *and* instance label for every pixel. Following the categorization for instance segmentation methods, approaches can generally be grouped into two categories. First, top-down or proposal-based methods rely on parallel semantic and object detection branches, where the latter predicts bounding boxes, which are further refined with the semantic segmentation to instance masks. Second, bottom-up or proposal-free approaches cluster pixels based on pixelwise instance embeddings, such as predicted features, relative positions, or semantic segmentation. This thesis uses the terms top-down and bottom-up because some methods generate instance proposals by clustering instance embeddings. In this case, the term proposal-free would be misleading. Nevertheless, these methods are considered bottom-up since they generate instances by clustering pixelwise embeddings instead of deploying a detection network.

Top-Down Approaches

Current state-of-the-art methods [Por19, Xio19, Moh20] mostly rely on Mask R-CNN [He17] because it predicts not only bounding boxes but also object masks. Additionally, it uses a Feature Pyramid Network (FPN) [Lin17] to better recognize objects at multiple scales. Similar to U-Net, and novel for the object detection task, FPNs upsample feature maps again after the feature extractor with the help of skip connections. However, predictions are not only performed on the last feature maps but are made independently for all feature map scales to improve multi scale object recognition, depicted in Fig. 2.13. UPSNet [Xio19] proposes a panoptic architecture based on Mask R-CNN with an additional semantic segmentation branch relying on deformable convolutions [Dai17b]. A parameter-free panoptic head resolves class conflicts between semantic and instance predictions and introduces a dedicated unknown class for non-resolvable conflicts. Seamless [Por19] also uses an architecture similar to Mask R-CNN and relies on a ResNet backbone enhanced with an

FPN. A novel semantic head exploits the multi scale features by applying individual ASPP modules to each scale to improve the aggregated information for semantic segmentation. EfficientPS [Moh20] deploys an EfficientNet [Tan19] followed by a novel 2-way FPN. The latter improves multi scale feature aggregation since aggregation is not only performed from low to high resolution but also vice versa. An enhanced semantic head captures fine details and long-range context more effectively.

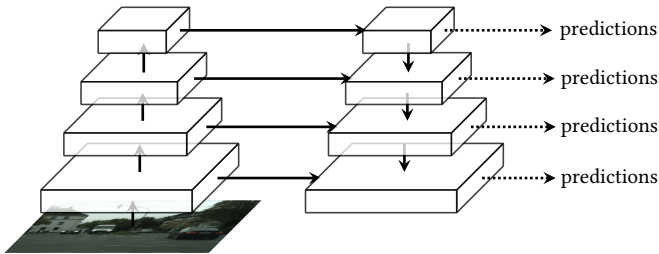


Figure 2.13: Feature Pyramid Networks (FPNs) provide predictions based on multi scale feature maps to improve object recognition.

Bottom-Up Approaches

One of the first proposal-free panoptic approaches was DeeperLab [Yan19c]. It relies on a keypoint-based representation of instances based on the four bounding box corners and its center. The instance branch predicts a heatmap for these keypoints and multiple short- to long-range offset maps, which are the foundation of the instance clustering. Single-Shot instance segmentation with Affinity Pyramids (SSAP) [Gao19] predicts a pixel-pair affinity pyramid, determining the probability that two neighboring pixels belong to the same instance. The instance clustering is computed by an efficient graph partitioning module based on affinity and semantics. Panoptic-DeepLab [Che20] proposes a clustering strategy built upon center and offset regression, illustrated in Fig. 2.14. It comprises a shared backbone followed by a dual ASPP and dual decoder setup for independent semantic and instance branches. The latter predicts a center heatmap indicating the position of instances which are represented by their centers. The second prediction, the offset vectors, point

for every pixel to its corresponding center. The class-agnostic clustering is performed based on these two predictions. It extracts k_c centers with the highest score as instance candidates from the heatmap and assigns thing pixels to the closest offset-indicated center candidate, illustrated in Fig. 2.14. Center candidates without assigned pixels are discarded.

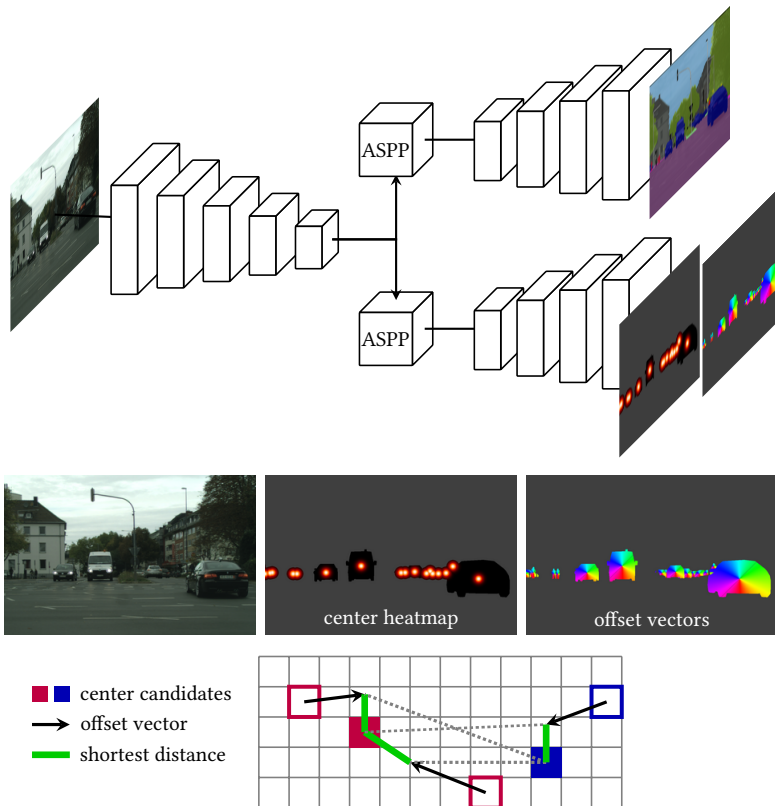


Figure 2.14: Panoptic-DeepLab clusters instances based on a center heatmap and offset vectors. The offset vectors are converted to an angle for visualization by a color wheel. The clustering assigns pixels to the center closest to the offset position, and each center represents a unique instance.

2.3 3D Scene Understanding

Robots and autonomous vehicles operate in a 3D environment, which makes 3D geometric information provided by 3D point clouds highly valuable. Common sources of point clouds are RGB-D cameras and light detection and ranging (lidar) sensors. The former are usually used in indoor scenarios [Sil12, Son15, Arm16, Dai17a], while lidar sensors are used outdoors [Hac17, Roy18, Beh19, Cae20, Pan20a, Xia21, Kur21]. The basic measurement principle of a lidar sensor is the emission of a laser pulse at time t_0 , which is reflected when hitting an obstacle. The sensor's detector recognizes this reflection at time t_1 , and the distance r to the obstacle can be computed based on the time of flight and speed of light c_0 :

$$r = \frac{1}{2} \cdot (t_1 - t_0) \cdot c_0. \quad (2.12)$$

Polar and azimuth angles (θ, ϕ) specify the laser direction for these measurements and are an intrinsic property of the sensor, determined by design or sensor rotation. The result is a measured 3D position in spherical coordinates $\tilde{\mathbf{p}} = (r, \theta, \phi)$, which can be transformed into Cartesian coordinates

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r \cos(\phi) \sin(\theta) \\ r \sin(\phi) \sin(\theta) \\ r \cos(\theta) \end{bmatrix}. \quad (2.13)$$

Repeating this measurement process thousands or even millions of times provides a point cloud of the sensor's surroundings, which can be represented as set $\mathcal{P} = \{\mathbf{p}_n \mid 1 \leq n \leq N\}$ or matrix $\mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2 \ \cdots \ \mathbf{p}_N]^T \in \mathbb{N}^{N \times 3}$. One possible and frequently used sensor setup in current outdoor datasets is a vertical stack of laser emitters and detectors spinning around the vertical axis, see Fig. 2.15. Compared to point clouds recorded with RGB-D cameras in indoor environments, outdoor point clouds from a lidar usually cover a much larger area, are relatively sparser, and have a point density strongly varying with distance. These properties impose additional challenges on approaches for 3D scene understanding.

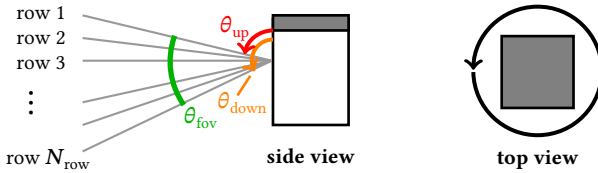


Figure 2.15: Common setup of a lidar sensor with pairwise vertically stacked laser emitters and detectors spinning around the vertical axis.

Driven by the value and importance of 3D information for scene understanding, approaches based on deep learning for various related tasks emerged, building upon the huge success of CNNs in the image domain. However, one of the main challenges is the unstructured nature of point cloud data, which CNNs cannot directly process. Therefore, a significant research effort is put into developing suitable representations across various tasks to enable the efficient processing of point clouds by CNNs. These are discussed in the following Section 2.3.1. Early approaches [Mat15, Wan15, Qi16] mainly tackled the tasks of 3D object classification, retrieval, or detection. Soon after, point-wise tasks like semantic or instance segmentation followed [Qi17a, Tch17, Wan18b, Mil19]. More recently, the combined task of panoptic segmentation gained more and more attention [Zho21, Hon21, Sir22]. Sections 2.3.2 to 2.3.4 present a detailed overview of the current state-of-the-art for these pointwise 3D tasks.

2.3.1 Point Cloud Representations

Unlike images, point clouds cannot be processed with native 2D or 3D convolutions. Therefore, and independently of the task, a point cloud representation is required, which allows the processing with established CNN architectures. Alternatively, point-based approaches [Qi17a, Tho19] propose adapted convolution operations and architectures directly applicable to point clouds in **point view (PV)**, see Fig. 2.16, without requiring a preliminary transformation. One major advantage is that no transformation-induced loss of information occurs. However, no neighborhood relations are inherently represented in an unordered set of points but instead must be explicitly computed. Also,

the usual hierarchical aggregation of local context based on consecutively sub-sampled grid-shaped feature maps must be explicitly formulated and computed. Both operations are potentially expensive, especially for large scale point clouds. Hence, different representations based on regular grids have been derived to enable the application of conventional CNNs, which are subsequently called *views*. Motivated by the discussed polar nature of lidar measurements, these views can also represent point clouds in spherical or cylindrical coordinates. If required, Cartesian coordinates \mathbf{p} can be transformed into spherical coordinates $\tilde{\mathbf{p}}$

$$\mathcal{Q}^\theta(\mathbf{p}) = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arccos\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \\ \text{atan2}(y, x) \end{bmatrix} = \begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \tilde{\mathbf{p}}, \quad (2.14)$$

or cylindrical coordinates $\tilde{\mathbf{p}}^z$:

$$\mathcal{Q}^z(\mathbf{p}) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \text{atan2}(y, x) \\ z \end{bmatrix} = \begin{bmatrix} r \\ \phi \\ z \end{bmatrix} = \tilde{\mathbf{p}}^z. \quad (2.15)$$

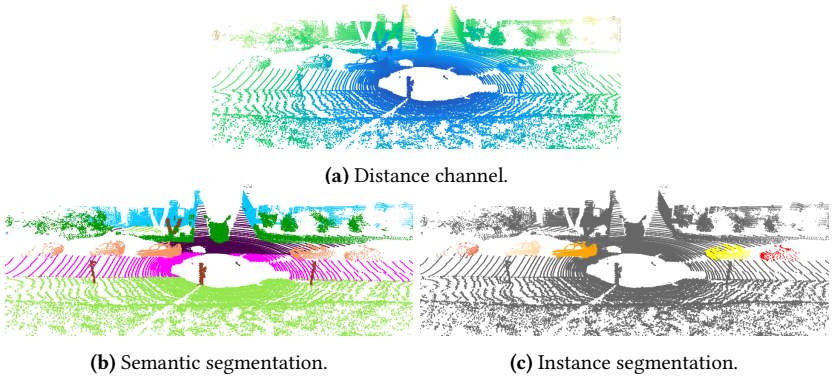


Figure 2.16: Point clouds in their native point-based representation. Visualized are the distance channel of the point cloud, as well as semantic and instance labels.

Without loss of generality, the following discretizations and projections are considered for a lidar sensor with a vertical and horizontal field of view defined as $\theta_{\text{fov}} = \theta_{\text{down}} - \theta_{\text{up}}$ and $\phi_{\text{fov}} = \phi_{\text{max}} - \phi_{\text{min}}$, whose measurements are in a distance and height interval of $r_{\text{fov}} = r_{\text{max}} - r_{\text{min}}$ and $z_{\text{fov}} = z_{\text{max}} - z_{\text{min}}$. To transform the predictions from the individual views back to the point cloud, every 3D point receives the prediction of its corresponding grid cell.

Voxel View

Motivated by the 3D nature of point clouds, a straightforward discretization into a grid are Cartesian or cylindrical voxels, the building blocks of the voxel view (VX). The discretized voxel coordinates of cylindrical 3D points for a voxel grid of size $H \times W \times D$ can be computed by [Zhu21b]:

$$\mathcal{P}^{\text{VX}}(\tilde{\mathbf{p}}^z) = \begin{bmatrix} [(r - r_{\text{min}}) \cdot r_{\text{fov}}^{-1} \cdot H] \\ [(\phi - \phi_{\text{min}}) \cdot \phi_{\text{fov}}^{-1} \cdot W] \\ [(z - z_{\text{min}}) \cdot z_{\text{fov}}^{-1} \cdot D] \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{u}^{\text{VX}}. \quad (2.16)$$

The x , y , and z coordinates of the Cartesian representation can be discretized accordingly. Point clouds and their input features can be transformed into a voxel grid based on these coordinates. In general, the transformation suffers from the many-to-one problem, meaning that multiple 3D points lie inside one voxel. Therefore, a handcrafted or learned fixed-sized feature vector is required, also called encoding, which represents an arbitrary number of points.

The voxel view’s advantages are that it retains the 3D structure of the data and inherently contains 3D neighborhoods. On the other hand, the major drawback of the dense voxel view, alongside the introduced quantization error, is the explicit representation of empty space, resulting in high memory and computational demands. At the same time, the sparsity of point clouds results in predominantly empty voxels. Sparse convolutions [Gra15] have been proposed to speed up computation and reduce the memory footprint for inputs with predominantly empty voxels. Therefore, only non-empty voxels are represented, and convolutions are only applied to these cells. One drawback of

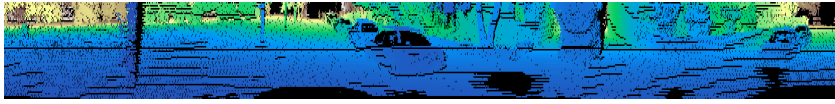
this implementation is the reduction of sparsity after each layer. The resulting voxel of a convolutional or pooling operation is only empty if all its inputs within the according receptive field are empty. Submanifold Sparse Convolutional Networks [Gra18] address this by restricting the output of these layers to the initially non-empty voxels, keeping the sparsity and the full benefit of sparse convolutions. While sparse convolutions significantly reduce memory and runtime overhead and enable the voxel view for large scale point clouds, neighborhoods are no longer implicitly represented. Nevertheless, the gains for omitting empty space outweigh the additionally required computations as long as the grids have less than 10% occupied cells [Gra15].

Range View

The range view (RV) is a 2D representation resulting from a spherical projection [Mil19]. It is closely connected to lidar sensors since it exploits the spherical representation of 3D points, which is directly provided by many lidar sensors. For a range image of size $H \times W$ the 2D projection coordinates of a spherical point are defined by:

$$\mathcal{P}^{\text{RV}}(\tilde{\mathbf{p}}) = \begin{bmatrix} [(\theta - \theta_{\text{up}}) \cdot \theta_{\text{fov}}^{-1} \cdot H] \\ [(\phi - \phi_{\text{min}}) \cdot \phi_{\text{fov}}^{-1} \cdot W] \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{u}^{\text{RV}}. \quad (2.17)$$

The point cloud and associated features, such as intensity, as well as the ground truth, can then be transformed into the range view based on these 2D coordinates, as depicted in Fig. 2.17. The advantages of the range view are its dense 2D representation, which allows for very efficient processing. Additionally, it does not depend on the covered area because the range image size is independent of r_{fov} . Its disadvantages are the distortion of physical dimensions due to the spherical projection and adjacent points with a significant difference in distance r and 3D position. Furthermore, a combined point cloud from multiple overlapping lidar sensors introduces the many-to-one problem.



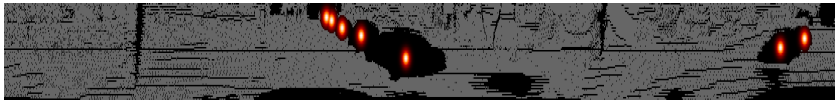
(a) Distance channel.



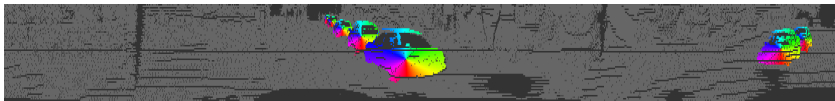
(b) Intensity channel.



(c) Semantic segmentation.



(d) Center heatmap.



(e) Offset vectors.



(f) Instance segmentation.

Figure 2.17: Point clouds represented in range view. The upper images show the distance (a) and intensity (b) measurements, followed by the ground truth semantic segmentation (c). Images (d) and (e) depict the ground truth center heatmap and offset vectors required for bottom-up instance clustering to predict an instance segmentation (f).

Bird’s Eye View

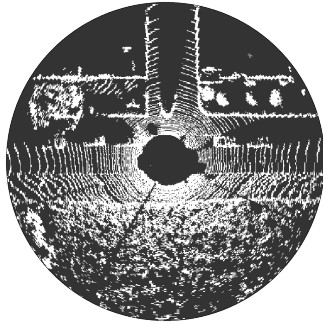
The bird’s eye view (BEV) projection omits the z -axis to project the 3D point clouds onto the xy -plane. Alternatively, a projection based on cylindrical coordinates onto the $r\phi$ -plane is also possible [Zha20c]. The 2D image coordinates for a polar bird’s eye view image of size $H \times W$ are computed by:

$$\mathcal{P}^{\text{BEV}}(\tilde{\mathbf{p}}^z) = \begin{bmatrix} [(r - r_{\min}) \cdot r_{\text{fov}}^{-1} \cdot H] \\ [(\phi - \phi_{\min}) \cdot \phi_{\text{fov}}^{-1} \cdot W] \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{u}^{\text{BEV}}. \quad (2.18)$$

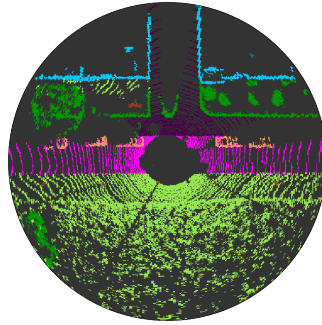
The results of this projection are depicted in Fig. 2.18. Similar to the voxel view, the bird’s eye view has to deal with the many-to-one problem by computing handcrafted or learned encodings based on the points inside each cell. It is similarly efficient as the range view, while the spatial separation of individual instances is superior since there are rarely occlusions along the z -axis for common thing classes. This is a valuable property for the clustering of bottom-up panoptic segmentation. However, the bird’s eye view is not particularly dense, with more than half of the cells being empty for standard lidar sensors. Additionally, small vertical objects are improperly represented.

2.3.2 Semantic Segmentation

After the first approaches [Mat15, Wan15, Qi16] have predominantly tackled object classification and detection, the pioneer PointNet [Qi17a] also addressed semantic segmentation. However, the first 3D semantic segmentation methods rarely scale to large scale outdoor scenarios due to the mentioned challenges of large covered areas, increased sparsity, and varying point density. With the rise of outdoor datasets [Hac17, Roy18, Beh19, Cae20, Pan20a, Xia21, Kur21], a significant amount of research effort shifted towards these scenarios, which originate mainly from the driving domain. Approaches have been proposed based on the different point cloud views discussed in the previous section, which are also combined to multi view approaches.



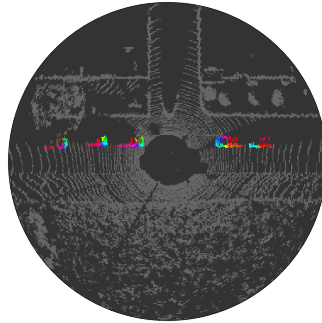
(a) Occupancy.



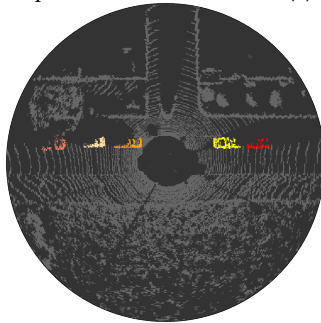
(b) Semantic segmentation.



(c) Center heatmap.



(d) Offset vectors.



(e) Instance segmentation.

Figure 2.18: The polar bird's eye view. The first image (a) shows the occupied cells, followed by the semantic segmentation (b). Bottom-up instance clustering requires center heatmap (c) and offset vectors (d) to predict the instance segmentation (e).

Point-based Approaches

Point-based approaches directly process raw point clouds in the point view without preceding transformation. New network architectures and redefined convolution operations have been proposed for this purpose, which can be roughly assigned to different categories, loosely following the survey of Guo et al. [Guo21]. The most influential categories are architectures based on the **pointwise application of MLPs** and **point convolutions** directly applicable to 3D points. Graph-based methods as the third category, such as [Lan18, Lan19a], do not play a notable role in the current state-of-the-art outdoor semantic segmentation. The interested reader is referred to [Guo21].

The pioneering approach of directly processing raw point clouds and **pointwise application of MLPs** was PointNet [Qi17a]. It repeatedly applies shared MLPs to every input point to compute individual feature vectors. Hence, these MLPs are called pointwise Multi-Layer Perceptrons (pMLPs) and are followed by a symmetric aggregation function for global feature aggregation, such as max pooling, illustrated in Fig. 2.19. The symmetric property ensures that the order of points does not influence the results since point clouds are unordered sets of points. For semantic segmentation, the global output feature vector \mathbf{y} is concatenated with the local feature vectors and further processed by pMLPs to predict the semantic labels based on local and global information. While PointNet has a low computational complexity, a single global feature aggregation strongly limits the capturing of hierarchical spatial relations, which are important for semantic segmentation. Its successor PointNet++ [Qi17b] tackles these shortcomings by applying individual PointNets to local regions in a hierarchical fashion. A sampling layer determines the region centers based on iterative Farthest Point Sampling (FPS), and a grouping layer selects points from the center’s neighborhood to form local regions. Subsequent approaches propose new and improved neighborhood aggregation strategies to capture context hierarchically. The proposed strategies are inspired by Scale Invariant Feature Transform (SIFT) [Jia18], build upon concentric spherical shells [Zha19a], are based on densely connected local webs [Zha19b], or combine geometric and feature neighborhoods [Eng19]. Other approaches deploy

attention [Yan19b, Hu20b] or RNNs [Eng17, Ye18] to exploit relations between points for context aggregation.

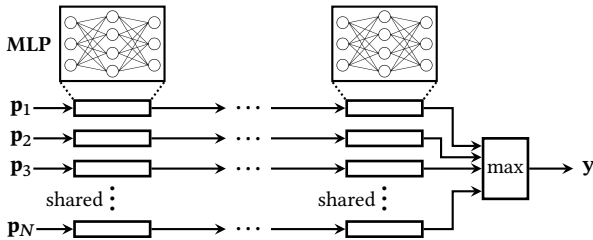


Figure 2.19: The PointNet architecture applies layer-wise shared MLPs followed by a symmetric operation, such as channel-wise maximum.

Many of the approaches mentioned so far face difficulties in scaling to outdoor scenarios, either in terms of computational complexity or quality of the results. The computational complexity is mainly impacted by FPS and k -nearest neighbor (k NN) search, with up to 57% of the overall runtime spent on data structuring [Liu19b]. The mediocre results are caused by the fact that local and global context aggregation is more challenging for outdoor point clouds, as discussed at the beginning of Section 2.3. More recent approaches started to address this challenge to improve the results. PointASNL [Yan20] refines the region centers sampled with FPS based on learned shifts and introduces a local-nonlocal module to improve the capturing of local and long-range context. Another strategy is multi task learning [Una21] with the added prediction of 3D objects. However, none of these approaches solves the high runtime demands. RandLA-Net [Hu20b] addresses both shortcomings by simultaneously improving the quality and efficiency of the neighboring feature pooling. It replaces the expensive FPS by random sampling. In addition, a novel attentive local feature aggregation module improves local context aggregation. Another approach is presented by Qiu et al. [Qiu21] and comprises a bilateral context and adaptive fusion module. The former augments pointwise features with explicit geometric information provided by the point cloud at different resolutions. The latter adaptively fuses multi resolution features to

provide enhanced features for 3D semantic segmentation. Both approaches significantly reduce runtime and considerably improve the results.

Approaches belonging to the **point convolution** category adapt the convolution operation for point clouds. They are frequently used for 3D shape classification [Guo21] but also for indoor semantic segmentation, such as convolution on \mathcal{X} -transformed features [Li18], PointConv [Wu19b], and dilated point convolutions [Eng20b]. However, only a few approaches of this category are designed for large scale point clouds with tens of thousands of points [Wan18a, Tho19, Bou20]. Parametric continuous convolution [Wan18a] is a learnable operator based on parameterized kernel functions which are approximated by an MLP. Kernel point convolutions (KPCov) [Tho19] use a flexible number of continuous and learnable locations in Euclidean space as kernel points. This property considerably increases the flexibility over fixed grid convolutions because these so-called deformable convolutions learn to adapt their kernel to local geometry. In addition, a regular subsampling strategy ensures increased robustness to varying point densities.

Despite the achieved improvements [Tho19, Hu20b, Qiu21] in the outdoor domain, point-based approaches still suffer from mediocre segmentation results [Hu20b] or high computational complexity [Tho19, Qiu21]. Hierarchical context aggregation is still more sophisticated and efficient in structured grid representations. This is reflected in a lower computational complexity while achieving predominantly better segmentation results.

Projection-based Approaches

Aiming for the application of established and efficient 2D CNNs, projection-based approaches project 3D point clouds onto 2D subspaces. Commonly deployed methods are the spherical or bird’s eye view projection presented in Section 2.3.1. However, also more complex projections exist, such as virtual tangent planes [Tat18].

Approaches based on the **range view** predominantly build upon existing 2D architectures from the image domain with novel extensions or adaptations targeted for processing the projected 3D point clouds. SqueezeSegV1 [Wu18]

was one of the first methods relying on the range view as input representation for segmenting the road-object classes car, pedestrian, and cyclist. Its backbone is based on SqueezeNet [Ian16] and followed by a Conditional Random Field (CRF) to refine the road-object segmentation. The enhanced version SqueezeSegV2 [Wu19a] has an improved model structure and exploits synthetic data combined with unsupervised domain adaption to reduce domain shift. The most recent version SqueezeSegV3 [Xu20] predicts semantic segmentation and introduces spatially-adaptive convolutions to counteract the spatially-varying feature distribution in range images. RangeNet++ [Mil19] exploits the DarkNet architecture [Red18] and presents a back-projection of range view labels to the point cloud based on the nearest neighbors. Measurement uncertainties and ego motion during the continuous scan can introduce projection errors with multiple points projected onto the same cell. The most basic strategy simply assigns the same label to all points of one cell. In contrast, RangeNet++ additionally considers the neighborhood of a point to reduce the impact of these projection errors on the segmentation. Instead of an expensive k -nearest neighbor (k NN) search, the 5×5 -neighborhood in range view is used as a proxy. Weighted majority voting based on the differences in radial distance r determines the label for every 3D point. SalsaNext [Cor20] is an enhanced version of SalsaNet [Aks20] and introduces a set of improvements, such as the use of Lovász-Softmax loss [Ber18] and the replacement of transposed convolution layers with pixel-shuffle layers [Shi16]. While primarily designed for 3D object detection, one of LaserNet’s [Mey19b] intermediate results is a semantic segmentation of the range view input. Their architecture is based on the previously introduced DLA and also inspired the range view backbone of this thesis. LiteHDSeg [Raz21b] proposes harmonic dense convolutions and an improved global contextual module to capture multi scale context. A multi class Spatial Propagation Network (MCSFN) tackles the refinement of semantic boundaries. A parameter-free full interpolation decoding module is proposed by FIDNet [Zha21c], which is based on bilinear interpolation, as a more efficient upsampling alternative compared to transposed convolutions.

The **bird’s eye view** was originally proposed and is widely used for the task of 3D object detection. Early approaches [Che17b, Yan18, Sim19] relied on a

handcrafted feature encoding for each cell, which was subsequently replaced by learned encodings. PointPillars [Lan19b], influenced by VoxelNet [Zho18], proposes a PointNet-based bird’s eye view encoding, which is also beneficial for semantic segmentation. A PointNet is applied to all 3D points inside one cell and maps them to a fixed-size feature vector, illustrated in Fig. 2.20. As a result, the input to the 2D backbone is a learned feature encoding of the projected point cloud.

Looking at semantic segmentation, Zhang et al. [Zha18] use a Cartesian bird’s eye view and a handcrafted feature encoding, which concatenates all points inside one cell. PolarNet [Zha20c] builds upon the ideas of PointPillars and relies on a learned polar bird’s eye view encoding based on PointNet. They empirically show that a polar grid better matches the point distribution of lidar sensors than a Cartesian grid and leads to fewer empty cells. Both approaches apply a U-Net to compute the required semantic segmentation.

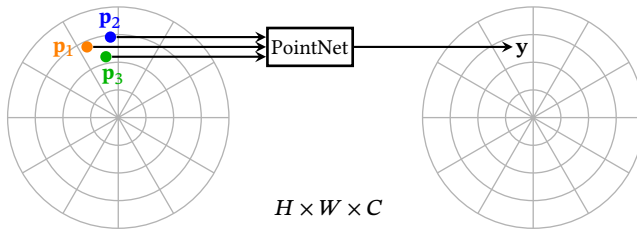


Figure 2.20: A PointNet is applied to every bird’s eye view cell to learn an embedding vector.

As a result of the 2D projections, range and bird’s eye view are the most efficient representations among the presented ones. In a direct comparison, range view methods achieve predominantly better results. While projection-based approaches outperform point-based methods in terms of segmentation quality and computational complexity, they cannot entirely compete with the current state-of-the-art segmentation results achieved in the voxel view. The projection induces either a significant amount of information loss due to the many-to-one problem or places points next to each other, which are far apart in 3D. The latter increases the challenge of separating class boundaries since implicit spatial separation along the distance or height axis is omitted.

Discretization-based Approaches

Most of the early semantic segmentation approaches [Dai17a, Tch17, Ret18], which relied on the **dense voxel view** as input representation, focused on indoor scenes, where this view’s runtime and memory downsides are less severe. SEGCloud [Tch17] deploys a 3D FCN followed by a trilinear interpolation and 3D CRF. The interpolation converts coarse voxel-level predictions back to the 3D points, and the CRF ensures global consistency and provides refined semantic segmentation. The Fully Convolutional Point Network [Ret18] applies a PointNet to the points in uniformly sampled regions to create a 3D feature map, which is processed by a 3D CNN. Nearest neighbor interpolation converts the voxel features back to the 3D points. PointLabeling [Hua16] was the only early approach for outdoor semantic segmentation. The point cloud is converted into a 3D occupancy grid and fed to a 3D CNN to predict voxel-level semantics. Afterwards, all points inside one cell receive the semantic label of this cell. The reported runtime of several minutes for an area of $100\text{ m} \times 100\text{ m}$ shows that, despite a huge voxel size of $0.3\text{ m} \times 0.3\text{ m} \times 0.3\text{ m}$, the dense voxel view cannot deal with large scale outdoor point clouds. This is also supported by the observation that representing at least 90% of a point cloud’s points in unique cells requires about 82.6 GB of memory when training with a batch size of 16 [Liu19b], even for compact indoor scenes.

Sparse convolutions enable the usage of the voxel view for outdoor scenarios and are frequently used by recent methods. These approaches propose new learnable modules to improve the segmentation quality and rely on a 3D U-Net architecture. S3Net [Che21c] introduces a sparse intra- and inter-channel attention module. The former addresses the local information loss caused by the discretization and usage of sparse convolutions, and the latter re-weights the channels of a feature map to learn better representations. JS3C-Net [Yan21] uses semantic scene completion as a supervisory signal to learn contextual shape priors from the dense aggregation of multiple point clouds. An improved sparse architecture based on attentive feature fusion and adaptive feature selection is presented by Cheng et al. [Che21d]. Attentive

feature fusion deploys small, medium, and large kernels in parallel. The former focuses on fine details and small semantic classes, while the latter aggregate global context and larger semantic classes. A learnable, weighted combination merges the features of these three branches. The second module, called adaptive feature selection, learns relations between channels across the three multi scale feature maps from the attentive feature fusion module to improve contextual information. A cylindrical [Zhu21b] instead of a Cartesian voxel view better matches the distribution of lidar point clouds, similar to the 2D polar bird's eye view. Zhu et al. [Zhu21b] further introduce an asymmetrical residual block to better match the point distribution and object shapes in point clouds of driving scenes. Additionally, context modeling based on dimension decomposition merges several low-ranked feature tensors into the final high-rank tensor.

The sparse voxel view achieves high quality 3D semantic segmentation and outperforms all other views discussed so far. Despite the sparse nature, one downside is still the considerably higher computational complexity compared to 2D representations such as range and bird's eye view.

Multi View Approaches

Considering the point cloud representations or views discussed so far, they all have individual strengths and weaknesses. Multi view approaches build upon the distinct properties of different views and exploit multiple representations to combine their strengths and compensate for individual weaknesses. These methods further improve the state-of-the-art or introduce efficient approaches with high quality results.

One category of multi view approaches builds upon voxel and point view to combine their complementary strengths. The main challenge of point-based approaches, the expensive sampling and forming of neighborhoods, can be omitted in this multi view setup since the voxel view implicitly provides 3D neighborhoods. One drawback of the voxel view, the loss of information during voxelization is simultaneously counteracted by the point view, which directly processes the original point cloud. Motivated by this, Point-Voxel CNN

(PVCNN) [Liu19b] deploys a low resolution voxel branch to extract coarse neighborhood context, combined with a high resolution point-based branch to provide individual point features. Sparse Point-Voxel CNN (SPVCNN) [Tan20] improves this further by replacing the dense voxel view with its sparse counterpart, which allows for higher voxel resolution and considerably improves the results. The combination of sparse voxel and point view can be further extended by the range view as a third branch [Xu21a]. Range and voxel features are repeatedly transformed back to the 3D points and fused across all three views by a gated fusion module. Afterwards, the fused multi view features are transformed back to the individual views and further processed by the individual backbones. FusionNet [Zha20a] proposes a VoxelMLP to exploit the voxel view for a fast neighborhood search to compute pointwise features using PointNet under consideration of a point’s neighborhood. Its sparse voxel branch additionally computes voxel features, which are fused with the pointwise features from the VoxelMLP. DRINet [Ye21b] builds upon the alternating application of Sparse Point-Voxel and Sparse Voxel-Point Feature Extraction. The former receives pointwise features as input, aggregates context with multi scale pooling, and transforms the pointwise features into voxel features. Sparse Voxel-Point Feature Extraction processes voxel features with a 3D backbone, and a geometry-aware attentive gathering generates high quality pointwise features. The successor DRINet++ [Ye21a] further improves the architecture by treating voxels as points, motivated by the observation that voxels can be considered an abstraction of the points inside with their 3D position defined by their center. In general, these multi view approaches cannot completely mitigate the computational complexity drawback of the voxel view [Liu19b, Tan20, Zha20a, Xu21a] or have to use very small 3D networks [Ye21b].

Motivated by the distinct underlying projections, another popular combination is range and bird’s eye view. Since the point clouds are projected onto complementary 2D planes, both views contribute distinct and valuable features. Additionally, this combination can be very efficient because the computational complexity of both views is relatively low. Early approaches proposed rather simple fusion strategies, e.g., Ali et al. [Ali21], which sum over the predictions from a range and bird’s eye view network to get the final 3D semantic

segmentation. However, they achieve only mediocre results, especially for a multi view approach. AMVNet [Lio21] proposes another late fusion strategy based on range view and polar bird’s eye view predictions. The respective backbones compute the semantic predictions in both views, which are back-projected to the 3D points. If the predictions of both views disagree, a point-based network refines these uncertain points based on the range and bird’s eye view features of the considered point and its nearest neighbors. Therefore, AVMNet requires an expensive nearest neighbor search and inherits one of the point view’s weaknesses. TornadoNet [Ger21] combines both views consecutively instead of deploying parallel backbones and starts with a bird’s eye view network, called pillar projection learning, to compute bird’s eye view features. Afterwards, these features are transformed into range view, combined with the range view input data, and processed by the main backbone. The methods considered so far combine multi view features or predictions only once and cannot exploit the full potential of this view combination. Recent approaches propose advanced range and bird’s eye view fusion strategies, like CPGNet [Li22c]. Pointwise features computed by a pMLP are projected into range and polar bird’s eye view and processed by individual backbones. In the next step, the features are back-projected using bilinear interpolation and fused across both views by another pMLP. Afterwards, the entire process is repeated. GFNet [Qiu22] also deploys individual backbones for range and polar bird’s eye view. In order to bidirectionally align and propagate complementary geometric information between both backbones and across views, a geometric flow module is proposed, which is applied after each upsampling step. Except for CPGNet [Li22c], these approaches are outperformed by 2D single view approaches [Ali21, Ger21] or still need expensive 3D operations [Lio21, Qiu22], leading to a computational complexity similar to single or multi view voxel-based approaches while achieving inferior results.

2.3.3 Instance Segmentation

Similar to the 2D image domain, instance segmentation is an essential task for 3D scene understanding. Significant progress has been achieved over the last years, enabled by two datasets, ScanNetV2 [Dai17a] and S3DIS [Arm16],

which provide pointwise semantic and instance labels for indoor scenes. As a result, most existing approaches tackle indoor instance segmentation and can be grouped into top-down and bottom-up methods, similar to the image domain. Another consequence of the indoor domain is the predominant focus on the point and voxel view.

Starting with top-down and **point-based** methods, Yi et al. [Yi19] introduce a Mask R-CNN inspired generative shape proposal network, which generates high quality 3D object proposals. Instances are predicted based on these proposals, supported by a semantic segmentation computed by a PointNet++. Based on pMLPs, 3D-BoNet [Yan19a] directly regresses 3D bounding boxes for all instances in a point cloud combined with the prediction of a pointwise mask. The result is a single-stage, anchor-free approach, which is end-to-end trainable.

The bottom-up approaches predominantly deploy a PointNet or PointNet++ as backbone to predict pointwise feature embeddings and focus on improving the clustering step. Similarity Group Proposal Network (SGPN) [Wan18b] computes a similarity matrix for all paired points based on the predicted embeddings. This matrix is used to generate an intermediate clustering, which is further heuristically refined and passed through Non-Maximum Suppression (NMS). Joint Semantic-Instance Segmentation (JSIS3D) [Pha19] applies a multi value CRF and incorporates the predicted semantics and embeddings for joint optimization to generate semantic and instance segmentation. Associatively Segmenting Instances and Semantics (ASIS) [Wan19a] and JSNet [Zha20d] fuse instance and semantic features before the final predictions to mutually improve these features and enhance predicted semantics and instance embeddings. Both approaches use mean shift clustering [Com02] to generate the instances. Zhang et al. [Zha21a] propose probabilistic embeddings and represent each 3D point as a tri-variate normal distribution. The core idea of the AS-Net [Jia20a] is to treat instance segmentation as candidate assignment problem. Candidates are selected based on pointwise features and represent different instances. An assignment module allocates points to the candidates, and a suppression module removes redundant candidates.

The top-down method 3D-SIS [Hou19] builds upon the dense **voxel view** to predict 3D bounding boxes with class labels. It is combined with a 3D mask network to predict a voxel-level instance mask. Bottom-up approaches predominantly rely on sparse U-Nets for feature extraction and, once more, focus on improving the clustering. The sparse backbone of 3D Multi Proposal Aggregation (3D-MPA) [Eng20a] predicts semantic and offset vectors, which the authors call object center votes. Proposal locations are then sampled from the predicted centers, and proposal features are learned by grouping and aggregating votes in the neighborhood of sampled centers. A graph neural network refines these features for a final proposal clustering. Liang et al. [Lia20] propose a structure-aware loss function, considering geometrical and embedding information to improve the learned 3D instance embeddings. A graph neural network refines the embeddings, which are finally clustered by mean shift. PointGroup [Jia20b] computes offset vectors pointing to the corresponding instance center instead of conventional embedding features. A dual clustering step based on original and offset positions produces candidate clusters. These are fed to a subnetwork, called ScoreNet, to provide a score for each cluster used by NMS to generate the final instances. Multi Task Metric Learning (MTML) [Lah19] combines feature embeddings with directional information for clustering. The clustering is provided by offset vectors pointing to the corresponding instance center. Mean shift clustering and NMS compute the final instances based on these predictions. Hierarchical Aggregation for 3D Instance Segmentation (HAIS) [Che21a] improves the clustering based on offset vectors using a hierarchical aggregation strategy to generate instance proposals progressively. OccuSeg [Han20] additionally predicts an occupancy output, while its instance clustering follows a graph-based segmentation schema.

Some approaches also combine bottom-up and top-down elements. Semantic Superpoint Tree Networks (SSTNet) [Lia21] over-segment point clouds in the first step to create superpoints. These are geometrically homogeneous neighborhoods, similar to superpixels in the image domain. Afterwards, a semantic superpoint tree is constructed bottom-up and based on semantic features of the superpoints, which are pooled from predicted pointwise semantic features. The tree is traversed top-down and split at intermediate tree nodes to

create instance clusters. SoftGroup [Vu22] performs clustering on soft semantic scores to reduce the influence of wrong semantic predictions. The instance proposals originating from the soft grouping step are refined in a top-down manner based on proposal features.

Only Zhang et al. [Zha20b] tackle instance segmentation in the outdoor domain. Point clouds are projected into **bird’s eye view** and processed by a 2D CNN to predict 2D offset vectors to instance centers for the clustering, complemented by predicted object heights used as a constraint. The other discussed methods are mostly not applicable to the outdoor domain. As explained in Section 2.3.2, approaches based on PointNet or PointNet++ are unable to aggregate sophisticated context in large scale outdoor scenarios, and runtime increases prohibitively with scenario size. The sparse voxel-based approaches predominantly use a voxel size of 2 cm, which is impractical outdoors. There, the covered area approximated for automotive grade lidar scans is about $(50 \text{ m})^2 \cdot \pi \approx 7854 \text{ m}^2$, compared to a typical indoor area [Dai17a] of 22.6 m^2 . State-of-the-art voxel-based approaches for outdoor semantic segmentation build upon larger voxel sizes [Tan20, Xu21a, Ye21b] and need to address the loss of resolution by novel extensions. Consequently, a significant research effort is still required for outdoor instance segmentation. This also applies to the clustering since the predominantly used mean shift clustering already requires at least 100 ms for small point clouds with 4096 points, which is an order of magnitude smaller than common outdoor point clouds.

2.3.4 Panoptic Segmentation

Many instance segmentation methods from the previous section also predict semantic segmentation and could potentially be extended to the task of panoptic segmentation. However, most of them use the semantics solely to support the instance task and neither jointly optimize both nor consider stuff classes, which is required for panoptic segmentation. Hence, they have been considered as instance segmentation approaches. Nevertheless, there are some exceptions [Wan19a, Pha19, Zha20d, Lia20] already aiming for the joint prediction of instance and semantic segmentation and can be considered among

the first 3D panoptic segmentation approaches, although their authors never call it panoptic. Supported by the public release of two large scale datasets, SemanticKITTI [Gei12, Beh19] and nuScenes [Cae20, Fon22], with pointwise semantic and instance labels, most of the recently published panoptic methods tackle outdoor panoptic segmentation of driving scenarios. Consequently, research not only focuses on improving the clustering [Gas21, Hon21, Li22a] but also on improved backbones and feature extraction [Mil20, Zha20c, Sir22, Xu22, Li22b]. Additionally, runtime complexity and real-time capabilities are considered. Again, existing methods follow the bottom-up [Mil20, Gas21, Li21, Hon21, Li22a, Xu22, Li22b] or top-down [Sir22] strategy, while others [Raz21a, Li21] do not follow the established patterns.

Panoster [Gas21] is a proposal-free approach with a novel learnable clustering step instead of a fixed post-processing one. Therefore, Panoster’s instance branch directly predicts instance IDs and is trained based on a differentiable confusion matrix over ground truth and predicted clusters. In addition, a post-processing step based on DBSCAN [Est96] merges fragmented instances or splits wrongly fused instances and significantly improves the results. While Panoster can be used in combination with arbitrary backbones, it has been evaluated for the point-based method KPConv and the range view-based approach SalsaNext, where KPConv achieves superior results.

Milioto et al. [Mil20] introduce a proposal-free, **range view**-based approach based on a shared DarkNet53 [Red18] encoder and dual decoder setup to predict semantics and instance centers. Instead of transposed convolutions or bilinear upsampling, which both leverage 2D proximity in the range view, a differentiable trilinear upsampling layer is introduced. It exploits 3D geometric information of the point cloud to upsample the 2D feature maps in both decoders. CPSEg [Li21] is a proposal- and clustering-free method with a task-aware attention module to force both decoders to learn comprehensive task-aware features. Geometric features extracted from the surface normals further assist the instance decoder. A similarity matrix based on the learned embeddings determines the instances. The proposal-based approach EfficientLPS [Sir22] uses a shared backbone followed by a semantic branch and a Mask R-CNN-based instance branch. A proximity convolution module

aggregates 3D neighborhoods into the range image prior to the backbone application, followed by a 2-way FPN for multi scale feature aggregation. The latter is supported by a novel range encoder network providing spatial information based on the distance channel. Additionally, a novel panoptic periphery loss is introduced to refine boundaries between instances.

The **bird's eye view** approach Panoptic-PolarNet [Zho21] stands on the shoulders of PolarNet and extends it to the panoptic segmentation task. Motivated by Panoptic-Deeplab from the image domain, their instance branch predicts a center heatmap and offset vectors for the clustering of instances. Panoptic-PolarNet shares not only the encoder among tasks but also the first part of the decoder to reduce the computational effort and simultaneously improve the results.

Based on the sparse cylindrical **voxel view**, DSNet [Hon21] introduces a clustering-based framework. The sparse backbone computes voxel features, which are shared among both tasks and further refined in the semantic and instance branch. The predicted voxel-based semantic classes and offset vectors are then transformed back to the 3D points. A learnable clustering module, called dynamic shifting, and motivated by mean shift clustering, can adapt its kernel functions on-the-fly for different instances. GP-S3Net [Raz21a] builds upon AF²-S3Net [Che21d] to compute a 3D semantic segmentation in the first step. Instead of a parallel instance branch, a downstream instance network performs over-segmentation on the thing classes. A graph is built based on these segmented clusters and processed by a graph neural network to predict the final instance segmentation.

The **multi view** approach SMAC-Seg [Li22a] presents sparse multi directional attention clustering. The predicted offsets are transformed from range to bird's eye view for clustering, and an attention module aggregates instance features for the individual clusters. Finally, a centroid-aware repel loss improves the separation of instances. Sparse Cross-Scale Attention Network (SCAN) [Xu22] uses a sparse 3D backbone to compute multi scale voxel features and derive point features. A cross scale attention module aggregates the multi scale voxel features, followed by a reduction into sparse bird's eye view to compute a sparse 2D centroid distribution. The pointwise features are used to predict

offset vectors and semantic classes for the 3D points. SCAN clusters instances based on the centroid predictions and offset vectors. PHNet [Li22b] starts with the computation of voxel features for occupied cells, which are then transformed into polar bird’s eye view. Semantic and instance features are computed by a 2D CNN and combined with the voxel features. The proposed kNN-transformer models interactions among instance classes and predicts the offset vectors for every voxel. Based on these, a pseudo-heatmap in bird’s eye view for potential instance centers is derived. A final clustering step provides the required instances.

2.4 Temporal Point Cloud Fusion

Sensors mounted on autonomous vehicles provide a constant stream of sequential measurements, such as 3D point clouds. The sequential nature combined with the continuity of the world provides a substantial potential to improve various tasks by considering past frames in order to exploit temporal information and dependencies. A frame comprises all relevant sensor measurements from one point in time. For approaches solely based on lidar, this is only the point cloud. On the other hand, a frame contains the data of multiple sensors for sensor fusion approaches. Early temporal attempts focused on object detection, quickly followed by temporal semantic segmentation approaches. Recently, the first approach started to tackle temporal panoptic segmentation. Five categories of fusion strategies can be identified across these tasks: aggregating the inputs, adding a temporal grid dimension, exploiting neighborhoods across time, employing the popular attention mechanism, or passing information recurrently through time.

Input Aggregation

The simplest temporal fusion is an early fusion strategy, merging multiple input point clouds and providing the aggregated point cloud to the backbone. In order to compensate ego motion, the point clouds are transformed to the current ego position beforehand. For the task of object detection, this can

be used [Cas18, Hu20a] to aggregate multiple point clouds before transforming them into bird’s eye view to increase its density and the detection results. Panoptic segmentation also benefits from input aggregation [Wan22b], where instance points of previous time steps are aggregated into the current point cloud. Another possibility to exploit past information on the input level are residual images in range view for the tasks of moving object [Che21b] and semantic segmentation [Wan22a]. Residual images represent the difference in the range channel of an ego motion compensated previous time step and the current range channel. The residual values are close to zero for the static environment and significantly deviate from zero for dynamic objects. While residual images are a valuable temporal input for moving object segmentation, the input aggregation strategy achieves only mediocre improvements for semantic segmentation and cannot convincingly exploit temporal information.

Temporal Grid Dimension

Approaches in this category stack feature maps from multiple time steps along a temporal dimension. Afterwards, 3D or 4D convolutions are applied to processes and fuse the stacked feature maps. Luo et al. [Luo18] extract feature maps from point clouds in bird’s eye view for multiple time steps. These feature maps are stacked along a temporal axis and subsequently fused by a 3D CNN. Alongside improving 3D object detection, the temporal approach also enables the forecasting of object motion. One additional dimension is required by MinkowskiNet [Cho19], which introduces 4D sparse tensors to incorporate previous frames. The temporally stacked sparse input tensor is processed by a 4D U-Net with generalized sparse operations to deal with the memory and runtime requirements. A hybrid kernel replaces the hypercube kernel to reduce the kernel size and the computational requirements, e.g., from $3^4 = 81$ to 29 for a 3×3 -kernel. Building upon these ideas, Mersch et al. [Mer22] propose an approach for moving object segmentation. The spatio-temporal information extracted by the sparse 4D CNN is the foundation for the decision about moving versus non-moving objects. The major drawback of the extra grid dimension is the significant impact on computational requirements, which increases with each considered past frame.

Spatio-Temporal Neighborhood

Neighborhood relations also exist across time since all point clouds can be transformed to the current pose. In contrast to input aggregation, the time is explicitly modeled as the fourth dimension in this category, and points must be spatially and temporally close to be considered neighbors. Consequently, these spatio-temporal neighborhoods allow the aggregation of point features across time. MeteorNet [Liu19a] deploys a PointNet++ for pointwise feature extraction followed by its main contribution, a temporal aggregation module called meteor module for pointwise spatio-temporal feature aggregation. A shared MLP aggregates the spatial-temporal neighborhood of a point, which is computed by a new chained-flow clustering strategy. The module is used in an early fusion strategy for semantic segmentation and applied to the aggregated point cloud. However, it can also be used in a late fusion setup. Point Spatio-Temporal Network (PSTNet) [Fan21] proposes point spatio-temporal convolutions for temporal semantic segmentation to exploit spatio-temporal neighborhoods. After disentangling space and time for consecutive point clouds, spatial convolutions capture the geometric structure of the point clouds, and temporal convolutions extract the temporal dynamics of the spatial regions. The high computational effort of computing neighborhoods in point clouds is further increased for spatio-temporal neighborhood search since the spatio-temporal point cloud is considerably bigger, or neighborhoods must be computed for multiple point clouds.

Attention

Another strategy is based on concatenating past and current features followed by channel-wise attention to fuse both. SpSequenceNet [Shi20] relies on the sparse voxel view and proposes cross-frame global attention for temporal fusion. This attention layer uses global information from the previous frame to compute channel-wise attention for the features of the current frame. Additionally, cross-frame local interpolation aggregates local information from a point's neighborhood in the previous frame. The follow-up work [Han22] uses the same attention mechanism but replaces the interpolation step with

an improved temporal-variation-aware interpolation module, which considers the feature variation inside the neighborhoods. Channel-wise attention is also used for moving object segmentation based on features from stacked residual images [Sun22] or ego motion compensated range images [Kim22]. The latter method additionally applies spatial attention to the fused feature maps. Another attention-based strategy is introduced in STELA [Kni21] and fuses the feature maps of multiple time steps after each encoder stage in order to aggregate every voxel’s sparse neighborhood across time.

Recurrent Neural Network

RNN-based approaches recursively aggregate feature maps by updating a temporal memory with current information. The memory transports the temporal information from time step to time step. This strategy was first used to improve 3D object detection [Els18, McC20] by recursively aggregating the last feature map prior to the detection head with a ConvLSTM [Shi15]. Yin et al. [Yin20] follow the same strategy but deploy an enhanced ConvGRU called attentive spatio-temporal transformer instead of a ConvLSTM. In contrast to the previous approaches, which rely on the bird’s eye view, Huang et al. [Hua20a] build their approach on the sparse voxel view. The extracted features are fed to a sparse ConvLSTM and recurrently aggregated. Instead of compensating the ego motion for the input point clouds, like previous approaches, the sparse 3D locations of the hidden state features are transformed to the current time step to compensate for the ego motion. The only semantic segmentation approach [Sch22b] proposes multiple fusion steps instead of a single temporal fusion step after the backbone. Recurrent temporal connections implemented by ConvGRU aggregate and pass feature maps at multiple locations inside the backbone from time step to time step to improve 3D semantic segmentation. In general, RNNs offer the potential to reuse their temporal memory at $t + 1$ when the next lidar scan is recorded. Except for Huang et al. [Hua20a], existing approaches compensate for ego motion in the input point clouds, prohibiting the reuse of already computed features and limiting the potential of RNNs. The other presented fusion strategies generally lack this potential, in addition to their already discussed drawbacks.

2.5 Sensor Fusion for Point Clouds

Autonomous vehicles or robots are often equipped with different sensor types and multiple sensors of the individual types. These offer the potential to exploit additional sensor data to improve various 3D tasks. Independently of the task, different fusion strategies with neural networks emerged [Che17b]. Early fusion combines multi sensor data directly at input level, deep fusion fuses intermediate feature maps potentially multiple times, and late fusion relies on the predictions of the different sensor modalities.

Sensor fusion is well established in the area of 3D object detection, especially for the combination of lidar and camera. Early fusion approaches generate regions of interest in camera images to determine where to look in the point cloud [Qi18, Wan19b]. Other methods [Vor20, Xu21b] enhance point clouds with semantic labels from image segmentation. Late fusion approaches combine bounding box predictions from both modalities [Pan20b, Pan22]. Deep fusion approaches can be further divided into **feature-level** and proposal-level fusion [Che17b, Ku18]. Strategies related to regions of interest, proposal, or bounding box fusion are specific to object detection. On the other hand, deep sensor fusion is task agnostic and can also be exploited for multimodal panoptic segmentation. Therefore, it will be the focus of this section. A comprehensive overview of the other categories is provided in [Mao22].

The first challenge to address for sensor fusion on **feature-level** is spatial alignment since camera and lidar features exist in different spaces. One common strategy is the projection of 3D lidar points [Hua20b, Zhu21a, Wan21b, Wan21a, Zha21b] or voxel centers [Sin19] onto the 2D camera image to extract spatially matching camera features. Zhao et al. [Zha21b] further improve this projection by learning pointwise correction offsets to compensate for deviations in the projection caused by calibration or time synchronization errors. Another strategy is the projection of camera features into bird's eye view based on voxel projection [Yoo20], lift splat shoot [Phi20, Liu22], parametric continuous convolution [Lia18, Lia19], or cross-attention [Che22]. For semantic segmentation, the geometric projection of camera features into range view is another popular strategy [Elm19, Mey19a, Kri20, Zhu21c].

The second challenge after the spatial alignment is the fusion of lidar and camera features. Proposed methods build upon simple addition [Lia18, Lia19], concatenation followed by convolution [Wan21a, Zha21b, Liu22] or residual-based addition [Zhu21c], cross-sensor attention [Che22], and gated fusion. The gating signal for the latter is computed from the concatenated [Yoo20, Wan21b] or added [Hua20b] lidar and camera features.

Only some mentioned approaches [Mey19a, Elm19, Kri20, Zha21b, Zhu21c] tackle semantic segmentation, which requires different architectures than object detection to restore the original resolution. This difference enables additional and distinct fusion strategies and architectures. Some methods [Elm19, Kri20] achieve notable improvements, however, for relatively weak baselines and only predict the road-object classes car, pedestrian, and cyclist. The fusion strategies of other approaches [Mey19a] achieve only small improvements, indicating that the camera’s full potential is not yet exploited. Additionally, none of the existing approaches consider the camera failure case or evaluate its impact on the results.

3 Concept

This thesis aims to design a multimodal network architecture based on deep learning for panoptic segmentation of 3D point clouds, which can exploit various information sources of an autonomous vehicle for high quality and robust results. The point cloud is the main source of information, usually originating from a lidar sensor, which can be supported by multiple other sources. First, time and temporal dependencies provide additional and valuable information when not only the current frame but also previous frames are considered. These offer the potential of temporally more consistent predictions since temporally aggregated features provide a more stable and dense context for every 3D point, compared to the sparse and incomplete context of a single lidar scan. Next, other sensors, such as camera or radar, provide complementary information about the environment based on their respective measurement principles. In particular, cameras with their usually much higher spatial resolution than a lidar contribute additional information. Consequently, different requirements can be derived, which the designed architecture needs to fulfill:

1. Process unstructured 3D point clouds based on either a single point cloud representation or a combination of multiple representations.
2. Incorporate information from previous frames to exploit temporal information and learn temporal dependencies.
3. Include information from complementary sensors, such as camera or radar, to exploit different measurement principles.

Fulfilling all these requirements provides a multimodal architecture for 3D panoptic segmentation. Therefore, the individual requirements are tackled by the main contributions of this thesis, which are then combined into one unified architecture:

1. A multi view framework [Due22] for processing unstructured and large scale point clouds based on range view, bird’s eye view, and point view.
2. A temporal fusion framework [Due20a] with novel recurrent feature map fusion enhances single frame approaches by recursively aggregating features of past frames.
3. A multi sensor framework [Due20b, Due21, Sch22a] for deep feature fusion of lidar and camera information based on two novel multi scale fusion strategies.
4. The first multimodal architecture that successfully leverages the combined potential of a multi view, temporal, and multi sensor architecture.

While the multimodal architecture provides the full potential and best results, the contributions of this thesis can be flexibly combined. For example, combining the multi view and temporal architecture provides strong results without using other sensors. Reasons not to use existing other sensors are, for example, a minor or no overlapping field of view or safety and redundancy considerations. Alternatively, if the entire multimodal framework is computationally too complex, combining the temporal and multi sensor fusion provides a sophisticated and highly efficient range view-based approach.

3.1 Multi View Architecture

Over time, various point cloud representations with different strengths and weaknesses have been proposed, with the most relevant ones introduced in Section 2.3.1. One possibility to combine different strengths and counteract weaknesses is the combination of individual views. Promising combinations are the sparse voxel and point view to reduce voxel resolution and omit neighborhood aggregation in unstructured point clouds [Tan20, Ye21b]. The sparse voxel view provides the 3D neighborhood aggregation, and the pointwise information in the point view allows a lower voxel resolution. Another promising combination is the range and bird’s eye view [Lio21, Li22c], motivated by the distinct underlying projections which omit different axes. Hence, the

2D neighborhoods of projected points differ and complement each other to reduce the impact of far apart points belonging to the same 2D neighborhood. The proposed multi view architecture [Due22] builds upon the range and bird’s eye view, motivated by four reasons:

- The range view provides a dense 2D representation of a sparse 3D point cloud with little to no loss of information for a single lidar. Small vertical objects, such as pedestrians, poles, and traffic signs, are well represented. Furthermore, the range view is independent of the covered area and directly represents a 3D point cloud without requiring a learned feature embedding for its cells.
- The bird’s eye view benefits from a low number of occlusions along the z-axis, especially for thing classes, such as traffic participants. This property facilitates the separation of individual instances and structures since they are often still spatially separated, despite the 2D projection. As a result, the bird’s eye view achieves great results for 3D object detection [Yin21], which is an essential property for the instance segmentation subtask.
- The combination of range and bird’s eye view approximates the voxel view by linearly separating it into two orthogonal 2D views. As a result, it provides a low computational complexity while preserving a strong representational power for 3D information. On the other hand, combining the sparse voxel and point view cannot completely mitigate the computational complexity drawback of the sparse voxel view [Tan20, Xu21a, Xu22].
- The range and bird’s eye views’ 2D structures support cell-based features association and aggregation across time, which is impossible in other views or amplifies their drawbacks. As discussed in Section 2.4, the aggregation of spatio-temporal neighborhoods in the point view is even more challenging than aggregating spatial neighborhoods. Additionally, the sparse voxel view requires either an additional dimension or becomes denser when aggregating features across time steps. The increased density is caused by the 3D points of previous frames, which

occupy previously empty voxels. While an increased density is generally favorable, it drastically reduces the computational benefits of the sparse representation. Above a certain density, it eventually becomes inefficient [Gra15].

Motivated by the success of Panoptic-DeepLab’s [Che20] bottom-up clustering approach in the image domain and the success of representing objects by center points [Yin21], this thesis relies on a bottom-up clustering for instance segmentation based on a center heatmap and offset vectors, introduced in Section 2.2.2.

Existing approaches [Zha20a, Lio21, Ger21, Qiu22] that rely on the range and bird’s eye view deploy either a simple fusion strategy with only minor improvements or require 3D neighborhoods for refinement and inherit the point view’s main drawback. In contrast, the presented approach [Due22] introduces a sophisticated point view backbone as a third parallel network and superior multi view link over late fusion, which simultaneously mitigates this drawback. The proposed multi view architecture illustrated in Fig. 3.1 aggregates neighborhoods and context in the range and bird’s eye view. Consequently, no context aggregation, neighborhood relations, and hierarchical point cloud subsampling are required in the point view. Instead, a unique feature vector is maintained and refined for each 3D point based on the aggregated features of range and bird’s eye view extracted at different scales. For that purpose, 2D CNNs are employed as backbones for both 2D views, which enable efficient context aggregation by exploiting the implicit neighborhood relations provided by the representations’ grid topology. Alongside mitigating the main drawbacks, this architecture provides a superior multi scale fusion strategy over late fusion for range and bird’s eye view features.

Furthermore, existing approaches based on range and bird’s eye view provide only semantic instead of panoptic segmentation. In contrast, the proposed architecture and its novel panoptic head exploit multi view benefits also for panoptic segmentation and the different requirements of its subtasks. The head relies on the point view for semantic segmentation and bird’s eye view for instance recognition. Offset vectors and center heatmap required for

bottom-up instance segmentation are predicted based on bird's eye view feature maps. This allows a dense 2D center heatmap and decouples object center positions from the measured 3D points since their center position is usually not directly measured. The head clusters all 3D points belonging to thing classes based on the 2D offset vectors and object center candidates in the bird's eye view to compute the 3D instance segmentation.

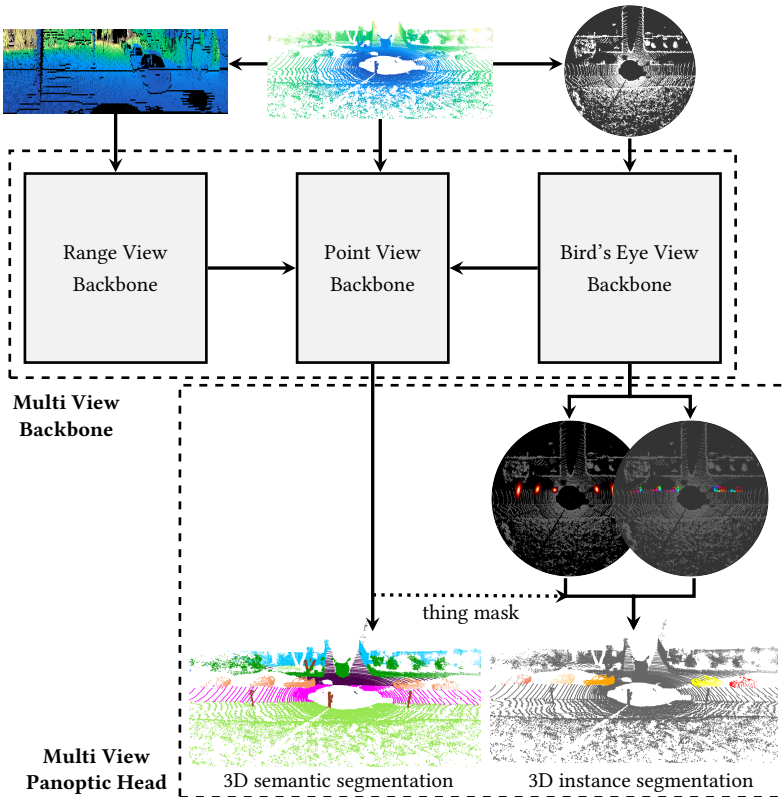


Figure 3.1: High-level architecture for multi view panoptic segmentation of 3D point clouds, which combines the range view, point view, and bird's eye view.

3.2 Temporal Multi View Architecture

Considering not only the current but also past frames provides the capability to exploit valuable temporal information and dependencies. This is especially beneficial for sensors with a high intra- and inter-class variance but a spatially rather low information density. For the targeted domain of real-time applications, waiting for a fixed number of frames and predicting them all at once is not feasible. Instead, a prediction with low latency is required for every arriving frame, as soon as it is provided by the sensor. This repetitive recording is rarely considered by existing approaches, see Section 2.4. Instead, they predominantly compute a prediction for the current frame by transforming a short sequence or temporal window of previous point clouds (≤ 5) to the current frame and processing them all at once. This strategy has two major drawbacks. First, it limits the temporal information that can be exploited due to the limited temporal window size, which linearly increases computational complexity and memory requirements. Second, all frames in the temporal window have to be processed in every time step. Consequently, individual frames are processed multiple times, as illustrated in Fig. 3.2.

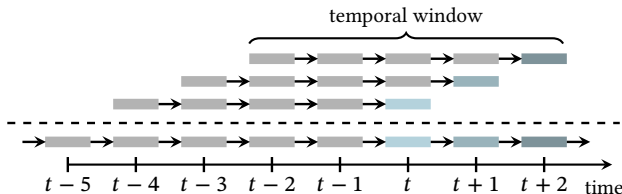


Figure 3.2: Processed frames for three arriving frames at t , $t + 1$, and $t + 2$. Existing approaches have to process their temporal window and individual frames repeatedly. In contrast, the proposed approach exploits recursion and processes only the arriving frame.

To address these limitations, the proposed architecture [Due20a] follows a different approach and introduces a novel temporal fusion of features and feature maps. It reuses a significant amount of computations from previous time steps and recursively aggregates features and information through time, without a limiting temporal window size. The general idea, which follows a

recursive filtering pattern, is shown in Fig. 3.3. Point clouds are processed in one of the chosen representations by a common backbone to compute deep aggregated feature maps for the current time step, similar to a conventional single frame approach. A temporal memory is passed through time and recursively updated with the features computed by the backbone and provides temporally aggregated features to the head for the final predictions. This recursive strategy allows reusing previous feature computations and only adds the update step as additional computation compared to single frame methods. As a result, the runtime is no longer connected to the temporal window size and the features of a potentially unlimited number of past frames can be aggregated into the memory.

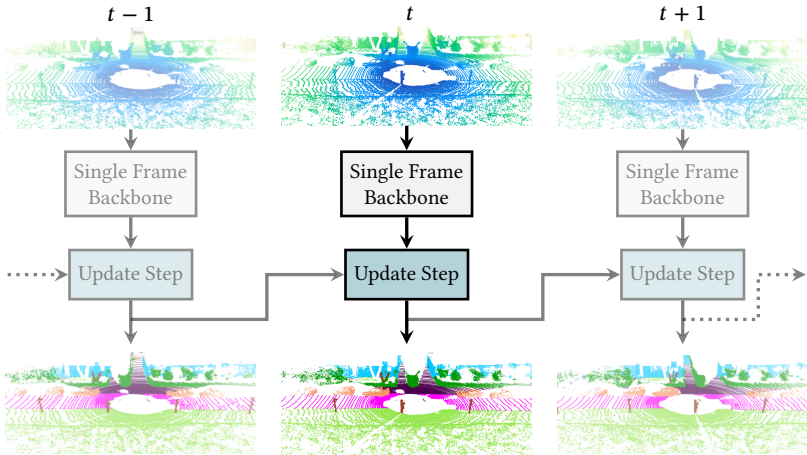


Figure 3.3: Recursive temporal update for aggregating features across time.

While this approach significantly enhances single view approaches, it is furthermore integrated into the multi view architecture to combine multi view and temporal benefits. Therefore, the range and bird’s eye view branch are extended by the proposed recursive memory aggregation, as depicted in Fig. 3.4, to incorporate feature maps from previous time steps in both views. The motivation for choosing the 2D views over the point view was already briefly

discussed in the previous section. Aggregating context across time requires finding spatially close features in the last time steps for all 3D points of the current time step, which is more efficient in a 2D grid than a 3D point cloud. Additionally, the choice of the 2D views allows both subtasks to benefit from the temporal fusion, see Fig. 3.4. Temporally fused feature maps are provided to the point view backbone to improve the point-based features for the 3D semantic segmentation. In parallel, the offset vector and center heatmap predictions required for the instance segmentation benefit from the temporally fused bird’s eye view feature maps.

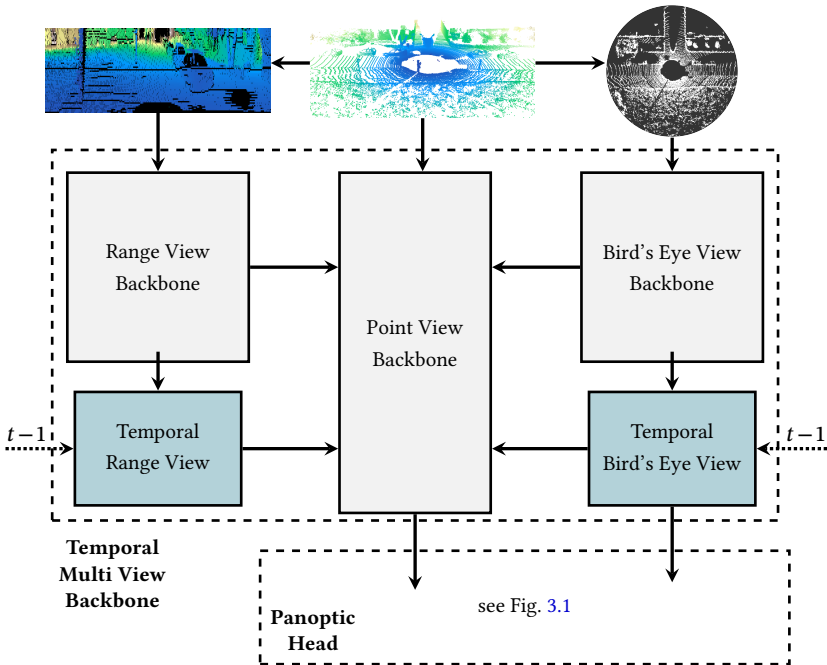


Figure 3.4: Temporal multi view concept for panoptic segmentation of 3D point clouds. It employs a recurrent temporal fusion in range and bird’s eye view to exploit temporal dependencies.

3.3 Multimodal Multi View Architecture

The presented temporal and multi view frameworks are single modality approaches focusing on the lidar sensor. However, other sensor modalities with distinct measurement principles, such as camera and radar, have great potential to provide additional information when fused with point clouds. In general, sensor fusion with neural networks follows early, deep, or late fusion, introduced in Section 2.5. Deep fusion offers the potential to fuse aggregated feature maps at multiple stages. Consequently, less information is lost when fusing sensors with different resolutions since features are spatially aggregated prior to fusion. Furthermore, deep fusion offers the potential to fuse features at multiple scales. In contrast, early and late fusion lacks these potentials. Early fusion of lidar and camera for 3D panoptic segmentation typically omits a significant amount of camera information since the camera resolution is usually much higher compared to a lidar. Late fusion combines both sensor modalities at the latest possible moment and therefore lacks the potential to improve feature extraction at earlier stages of the networks. These considerations motivate the choice of deep sensor fusion as the underlying strategy.

A promising sensor combination consists of camera and lidar because both sensors provide rather complementary information. Camera sensors usually measure RGB intensity values at high resolution, whereas lidar sensors provide valuable 3D geometric information with a relatively low resolution. Additionally, the considerable success of image-based scene understanding indicates the value of camera image information for understanding a vehicle's environment. The successful combination of RGB and depth images for 2D scene understanding [Sil12] motivates the fusion of camera and lidar information in the range view. The main advantage over the point view is, again, the grid topology which allows aggregating context for the fused features with standard 2D convolutions.

The proposed fusion architecture performs multi scale feature fusion of lidar and camera. Improving over existing approaches [Elm19, Mey19a, Kri20, Zhu21c], it ensures that the lidar baseline performance is still achieved in

case of missing camera information. Furthermore, two novel multi scale fusion strategies improve the exploited camera information and fused features. The underlying setup is illustrated in Fig. 3.5 and builds upon individual sensor backbones to extract lidar range view and camera features. These are provided at multiple scales to the proposed fusion branch, which geometrically transforms camera feature maps into range view feature maps. Afterwards, one of two proposed fusion strategies combines the multi scale feature maps. The fusion branch decouples the fusion from the backbones and is combined with an adapted training strategy so that each backbone can still provide single modality predictions as a backup in case of sensor failure. This property increases robustness against missing or unexpected camera output and is a considerable advantage over existing approaches.

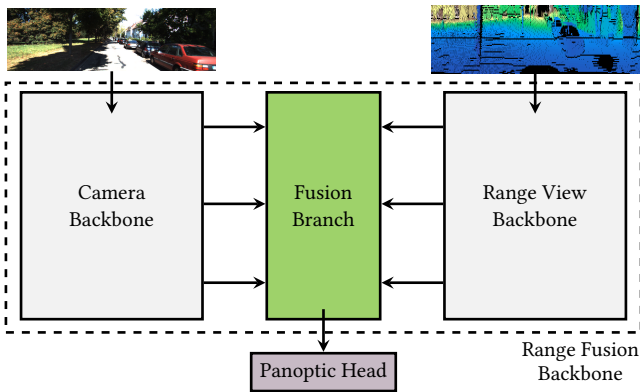


Figure 3.5: Range view-based lidar and camera fusion. The multi scale fusion branch decouples the individual sensor backbones from the fusion.

While this fusion approach considerably improves the panoptic segmentation in the range view, another important step is the integration into the temporal multi view architecture to ultimately combine multi view, temporal, and multi sensor benefits. This multimodal architecture is illustrated in Fig. 3.6, where the range fusion backbone replaces the single sensor range view backbone. Consequently, the temporal range view is applied to the multimodal features

and provides temporally enhanced multimodal range view features. In this setup, the range view feature maps propagated to the point view contain the fused features across lidar, camera, and time. On the other hand, the bird's eye view and its predictions lack the camera information. However, the main negative impact on the panoptic results are errors in the semantic segmentation, shown with an oracle test in [Zho21], which directly benefits from the sensor fusion. Additionally, the bird's eye view allows the integration of additional sensors in the future, such as radar or even online cloud-based map data.

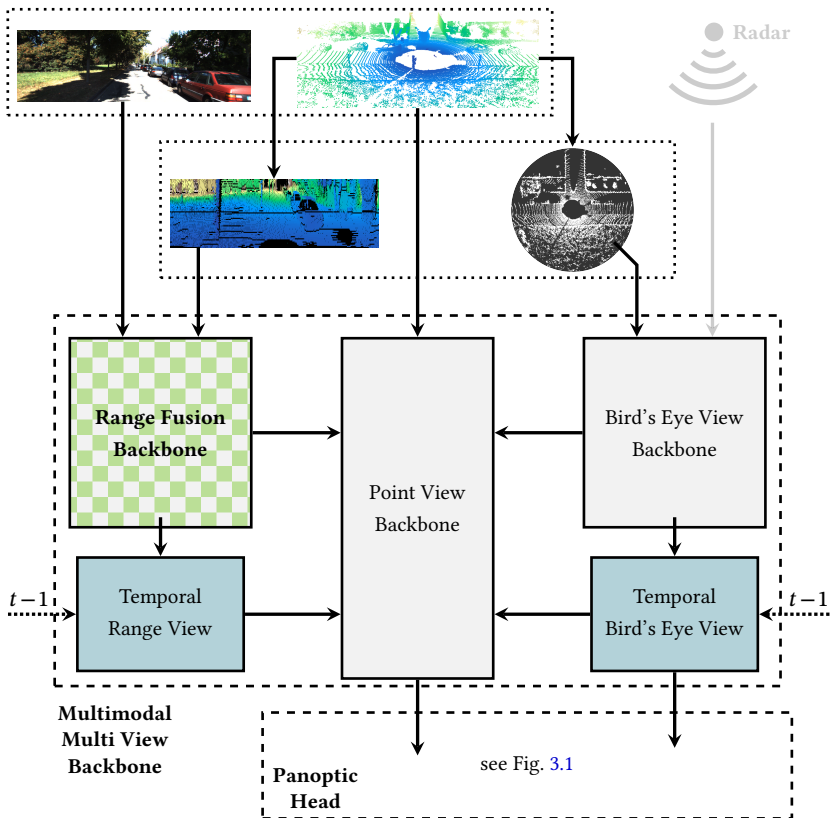


Figure 3.6: The multimodal architecture, which combines multi view, temporal, and sensor fusion benefits for 3D panoptic segmentation.

3.4 Multimodal Feature Map Transformation

The presented architectural concepts require the combination and fusion of features and entire feature maps across different sensors, time steps, and point cloud representations. Therefore, a spatial transformation is required to transform feature maps between different modalities and views. The enabling element is 3D information provided by the lidar point clouds or the lidar views themselves. These 3D points can be geometrically transformed to other sensors or time steps and can be projected into different views. The transformation requires ego poses and sensor extrinsics defined by homogeneous transformation matrices \mathbf{T} and are illustrated for a generic vehicle setup in Fig. 3.7. The transformation from the lidar (li) at time t to sensor S at time τ is defined by:

$$\mathbf{T}_{t \rightarrow \tau}^{\text{li} \rightarrow S} = (\mathbf{T}_{\tau}^{\text{ego}} \cdot \mathbf{T}^S)^{-1} \cdot \mathbf{T}_t^{\text{ego}} \cdot \mathbf{T}^{\text{li}}, \quad \mathbf{T}_{t \rightarrow \tau}^{\text{li} \rightarrow S} \in \mathbb{R}^{4 \times 4}. \quad (3.1)$$

The ego poses can be omitted if the source and destination time steps are identical. This combined transformation matrix transforms a homogeneous point cloud $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_n, \dots]^T \in \mathbb{R}^{N \times 4}$ to another sensor and time step:

$$({}_t \mathbf{P}_{\tau}^S)^T = \mathbf{T}_{t \rightarrow \tau}^{\text{li} \rightarrow S} \cdot \mathbf{P}_t^T. \quad (3.2)$$

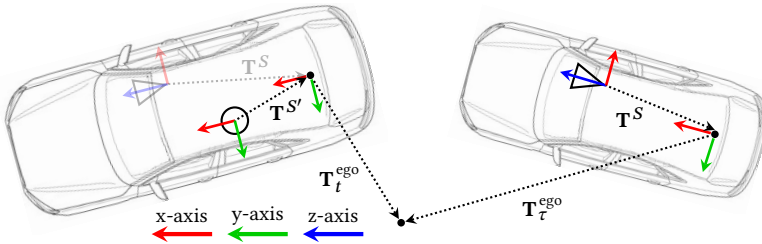


Figure 3.7: Vehicle with two sensors S and S' at two distinct points in time. The vehicle poses in the global coordinate system at time τ and t are given by $\mathbf{T}_{\tau}^{\text{ego}}$ and $\mathbf{T}_t^{\text{ego}}$. The sensor poses with regard to the vehicle coordinate system are specified by extrinsics \mathbf{T}^S and $\mathbf{T}^{S'}$.

Afterwards, a projection \mathcal{P}^ν projects the original or transformed points into one of the considered views ν :

$$\begin{aligned} \mathcal{P}^\nu: \mathbb{R}^4 &\rightarrow \{1, 2, \dots, H\} \times \{1, 2, \dots, W\} \\ \mathcal{P}^\nu(\mathbf{p}) &= \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{u}^\nu. \end{aligned} \quad (3.3)$$

Relevant views for this thesis are lidar range view, polar bird's eye view, and camera image view (IMG). The specific computation of the projection \mathcal{P}^ν depends on the view and has been introduced in Section 2.3.1. The camera projection follows the pinhole model [For12]. Additionally, Cartesian points must first be converted into spherical or cylindrical coordinates for range and polar bird's eye view, achieved by the functions \mathcal{Q}^θ and \mathcal{Q}^z , introduced in Eqs. (2.14) and (2.15). Computing the projection or cell index for a point \mathbf{p} and a target view ν at time τ usually requires more steps than just the projection itself and motivates a combined transformation \mathcal{J} :

$$\begin{aligned} \mathcal{J}_{t \rightarrow \tau}^\nu: \mathbb{R}^4 &\rightarrow \{1, 2, \dots, H\} \times \{1, 2, \dots, W\} \\ \mathcal{J}_{t \rightarrow \tau}^\nu(\mathbf{p}) &= (\mathcal{P}^\nu \circ \mathcal{Q}^{\theta|z}) (\mathbf{T}_{t \rightarrow \tau}^{\text{li} \rightarrow S} \cdot \mathbf{p}). \end{aligned} \quad (3.4)$$

Equation (3.4) combines the transformation, spherical or cylindrical conversion, and projection, where the first two steps are not always required:

1. Transform the points to another sensor S and/or time step τ via $\mathbf{T}_{t \rightarrow \tau}^{\text{li} \rightarrow S}$.
2. Convert Cartesian into spherical or cylindrical coordinates via $\mathcal{Q}^{\theta|z}$.
3. Project transformed points into target view ν of sensor S via \mathcal{P}^ν .

Computing the projection or cell index for each point \mathbf{p}_n of a point cloud provides an index matrix:

$$\mathbf{U}_\tau^\nu = \begin{bmatrix} U_{1,1} & U_{1,2} \\ \vdots & \vdots \\ U_{N,1} & U_{N,2} \end{bmatrix} = [U_{n,d}] := [\mathcal{J}_{t \rightarrow \tau}^\nu(\mathbf{p}_n)_d] \in \mathbb{N}^{N \times 2}. \quad (3.5)$$

Equations (3.4) and (3.5) are illustrated for camera and range view images in Fig. 3.8. The projection index \mathbf{u}_n of \mathbf{p}_n can be found in the n -th row of \mathbf{U} .

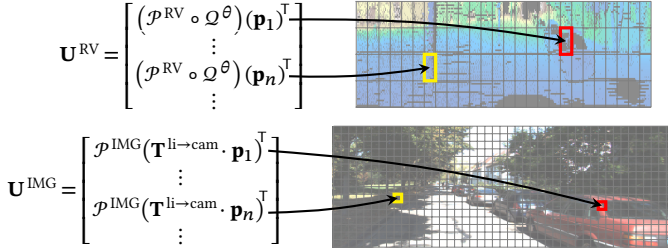


Figure 3.8: Projection of a 3D point cloud into lidar range view and camera image view.

Feature maps $\tilde{\mathbf{F}} \in \mathbb{R}^{N \times C}$ in point view, which originate from neural networks directly applied to unstructured point clouds, are in the most general sense matrices with individual entries $\tilde{F}_{n,c}$. These feature maps contain in the n -th row the feature vector of the n -th point \mathbf{p}_n of a point cloud, with \mathbf{p}_n residing in the n -th row of the point cloud matrix \mathbf{P} . On the other hand, feature maps $\mathbf{F} \in \mathbb{R}^{H \times W \times C}$ of 2D CNNs are in the most general sense 3D tensors. Their entries are in the following denoted as $F_{u,v,c}$, following [Goo16]. Since these feature maps are the result of processing a 2D input, such as camera or range view images, they can also be considered as a 2D grid of feature vectors at grid coordinates $\mathbf{u} = (u, v) \in \mathbb{N}^2$. Consequently, they are called 2D feature maps. Since 3D points can be projected into these grid cells, feature vectors from a 2D feature map can be assigned to every 3D point. Based on this relation, the 1D feature map transformation \mathcal{S} transforms a 2D feature map \mathbf{F} into a point view feature map $\tilde{\mathbf{F}}$:

$$\mathcal{S} : \mathbb{R}^{H \times W \times C} \times \mathcal{U} \rightarrow \mathbb{R}^{N \times C} \quad (3.6)$$

$$\mathcal{S}(\mathbf{F}, \mathbf{U}) := [F_{U_{n,1}, U_{n,2}, c}] = \tilde{\mathbf{F}} = [\tilde{F}_{n,c}] \in \mathbb{R}^{N \times C}$$

for all valid index matrices:

$$\mathcal{U} = \{ \mathbf{U} \mid \mathbf{U} \in \mathbb{N}^{N \times 2} \wedge 1 \leq U_{n,1} \leq H \wedge 1 \leq U_{n,2} \leq W \}. \quad (3.7)$$

The underlying idea is illustrated in Fig. 3.9. Equations (3.5) and (3.6) allow transforming feature maps \mathbf{F}_τ^y from an arbitrary 2D view and time step to the

point view of the current time step t :

$${}^{\nu}\mathbf{F}_t^{\text{PV}} := \mathcal{S}(\mathbf{F}_t^{\nu}, \mathbf{U}_t^{\nu}). \quad (3.8)$$

Equation (3.8) is the foundation for the multi view architecture to transform range and bird's eye view feature maps back to the point view.

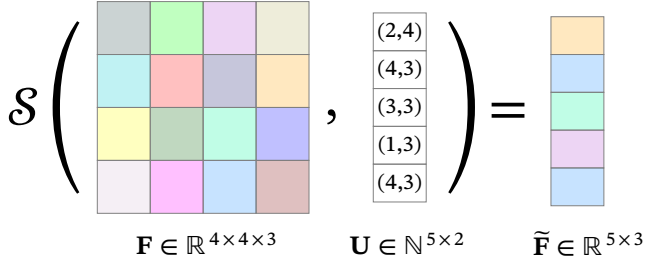


Figure 3.9: Visualization of the point view feature map transformation introduced in Eq. (3.6). The features in this example are RGB values visualized by color.

In addition, the proposed temporal and sensor fusion requires the transformation of camera image feature maps to the range view and the transformation of range and bird's eye view feature maps between time steps. Therefore, the ideas of Eq. (3.6) are extended to a general 2D feature map transformation $\hat{\mathcal{S}}$:

$$\begin{aligned} \hat{\mathcal{S}} : \mathbb{R}^{H' \times W' \times C} \times \mathcal{V} &\rightarrow \mathbb{R}^{H \times W \times C} \\ \hat{\mathcal{S}}(\mathbf{F}', \mathbf{V}) &:= [F'_{V_{u,v,1}, V_{u,v,2}, c}] = \mathbf{F} = [F_{u,v,c}] \in \mathbb{R}^{H \times W \times C} \end{aligned} \quad (3.9)$$

for all valid index tensors:

$$\mathcal{V} = \{ \mathbf{V} \mid \mathbf{V} \in \mathbb{N}^{H \times W \times 2} \wedge 1 \leq V_{u,v,1} \leq H' \wedge 1 \leq V_{u,v,2} \leq W' \}. \quad (3.10)$$

An example of the 2D feature map transformation is depicted in Fig. 3.10.

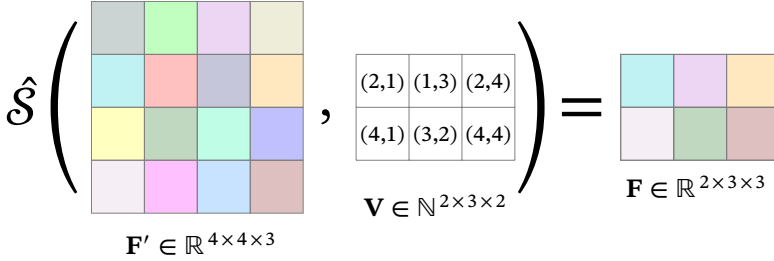


Figure 3.10: Visualization of the 2D feature map transformation introduced in Eq. (3.9).

In contrast to the index matrix \mathbf{U} , which is based on the point cloud itself, the index tensor \mathbf{V} is computed based on the 3D position $\mathbf{g}_{u,v}$ of the target view's cells:

$$\begin{aligned}
 \mathbf{V}_\tau^v &= [V_{u,v,d}] := [\mathcal{J}_{t \rightarrow \tau}^v(\mathbf{g}_{u,v})_d] \in \mathbb{N}^{H \times W \times 2}, \\
 \mathbf{g}_{u,v} &= [G_{u,v,1} \ G_{u,v,2} \ G_{u,v,3} \ 1]^\top.
 \end{aligned} \tag{3.11}$$

The definition of $\mathbf{G} = [G_{u,v,d}] \in \mathbb{R}^{H \times W \times 3}$ depends on the view. A range view cell's 3D position is defined by the 3D point of the point cloud, which is projected into the considered cell, illustrated in Fig. 3.11. If multiple points are projected into one cell, only one is kept, which is discussed in detail in the next chapter. On the other hand, the center of a bird's eye cell determines its 3D position independently of the point cloud when making a flat-earth assumption with $z = 0$.

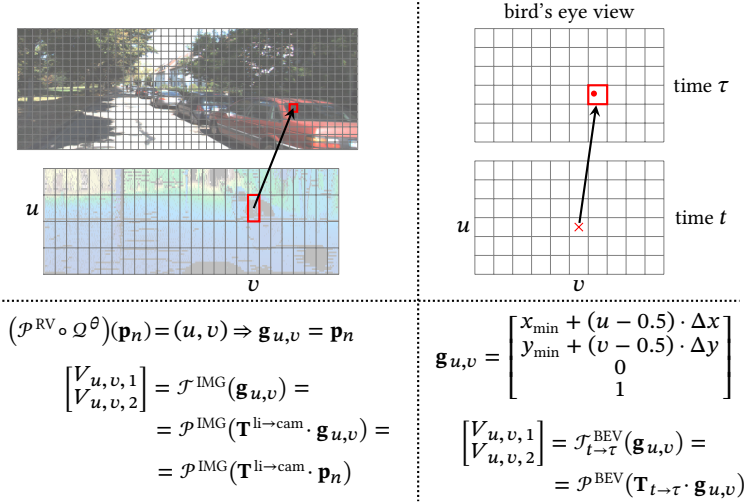


Figure 3.11: Examples for computing the index tensors. The left example illustrates the mapping from range view to camera image view and the left from bird's eye view at time t to time τ . The bird's eye view resolution or distance between cells is defined by Δx and Δy .

4 Multi View Panoptic Segmentation of 3D Point Clouds

The multi view approach presented in this chapter addresses the challenge of predicting an improved 3D panoptic segmentation for unstructured point clouds. It comprises individual backbones for range and bird’s eye view, which support a point view backbone. In the first step, range and bird’s eye view are considered individually to introduce single view approaches and derive 2D backbones for the multi view architecture. The novel multi view framework [Due22] for 3D panoptic segmentation builds upon and extends these backbones and is presented in the second half of this chapter.

4.1 Range View Network

Range view-based panoptic segmentation relies on point clouds represented as range images. These are processed by a 2D backbone to compute feature maps for semantic and instance segmentation, illustrated in Fig. 4.1. Multiple heads are deployed to provide the parallel predictions required for panoptic segmentation. On the one hand, the semantic head predicts a 2D semantic segmentation based on semantic feature maps \mathbf{F}_{sem} . On the other hand, the center and offset head use instance feature maps \mathbf{F}_{ins} to compute a heatmap and offset vectors for 2D bottom-up instance segmentation. The overall range view network is called RVNet, and its final 3D panoptic segmentation is the result of a k NN-based back-projection.

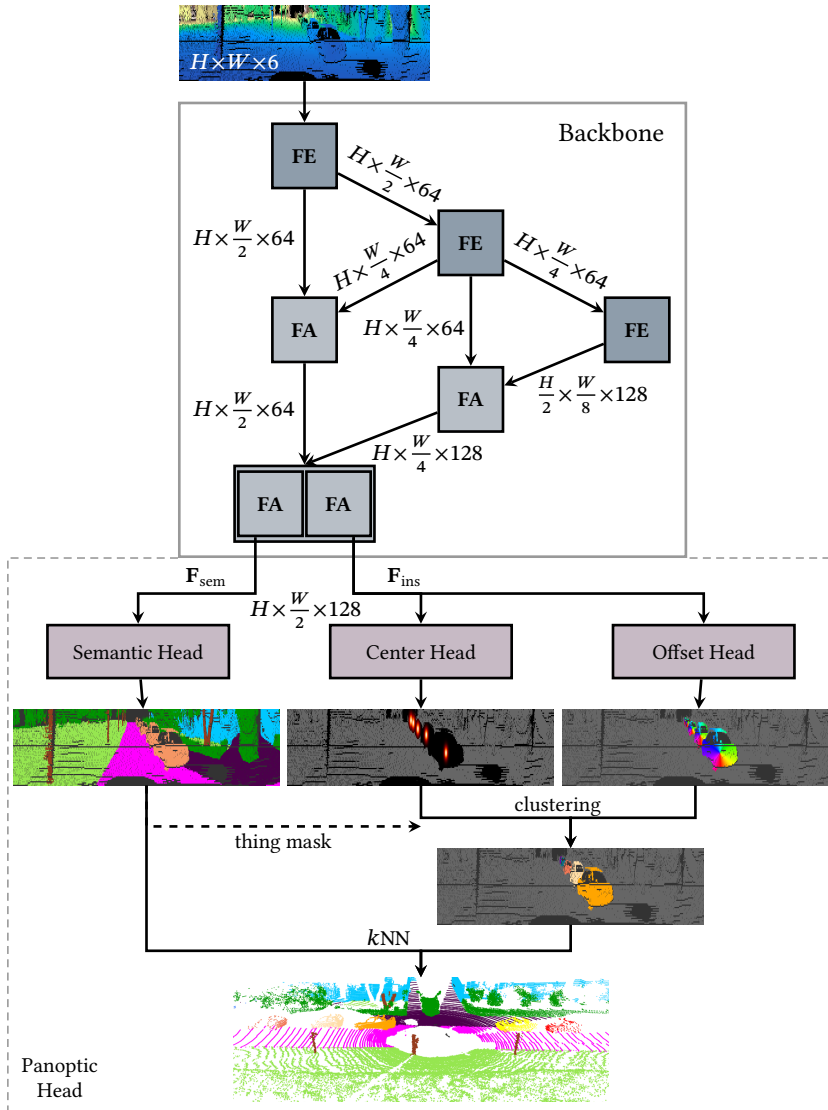


Figure 4.1: The range view network RVNet. It is composed of a 2D fully convolutional backbone based on feature extractors (FEs) and feature aggregators (FAs) [Mey19b], and three different heads. The edges are labeled with the feature maps sizes.

Input Representation

The underlying idea of the range view is to represent point clouds in spherical coordinates and discretize azimuth and polar angle into cell coordinates following Eq. (2.17). Ideally, all 3D points are projected to different grid cells. However, this requires that a point cloud is recorded from a single point of view and measurements are performed in equidistant azimuth and polar intervals. Since these requirements are rarely satisfied for point clouds originating from moving lidar sensors in the considered context, collisions occur when multiple points are projected into the same grid cell. One strategy to address this is the consideration of the intrinsic properties of the deployed lidar to adapt the mapping from angles to cell coordinates accordingly. This can significantly reduce collisions and information loss but omits the underlying regularity of the range view grid, which is an important property for convolution-based processing with CNNs. Hence, further projection strategies are discussed in addition to the regularly-spaced strategy of Eq. (2.17).

As a result of the functional principle of common lidar sensors introduced in Section 2.3, the resulting measurements are usually represented in spherical coordinates. While the distance measurement r depends on the environment, both angles are usually determined directly or indirectly by design. Many rotating 360° lidars use a vertical stack of laser heads rotating around their vertical axis. The position in this stack maps to a fixed polar angle and can be used as row index. On the other hand, the azimuth angle usually depends on the angular velocity and time. In this case, the column can be determined by enumerating the measurements for each laser head. The result is a purely sensor-dependent projection without collisions or information loss and is directly provided by most lidar sensors. However, the cells of the resulting grid are not necessarily equidistant.

For some use cases, such as many public datasets, it is impossible to derive the cell coordinates directly from the sensor since only the Cartesian point clouds are provided. These are often already pre-processed to compensate the ego motion during the continuous scan, which prevents restoring the raw measurements. However, sensor properties can still be taken into account to

optimize the projection. Considering lidar sensors with a grid-like measurement pattern of size (H, W) defined by intrinsic measurement angles $(\hat{\theta}_u, \hat{\phi}_v)$, spherical point clouds can be projected by [Due20a]:

$$\mathbf{u} = \begin{bmatrix} \arg \min_{1 \leq u \leq H} (|\hat{\theta}_u - \theta|) \\ \arg \min_{1 \leq v \leq W} (|\hat{\phi}_v - \phi|) \end{bmatrix}. \quad (4.1)$$

While Eq. (4.1) works for any arbitrary and irregular angle distribution, it has a higher computational complexity than Eq. (2.17). A combination of the presented projection strategies is also possible and applies distinct strategies for the individual dimensions. Some sensors, such as a Velodyne HDL-64E, have an equidistant azimuth but no entirely equidistant polar angle distribution. However, the upper and lower half individually does. In order to reduce the number of collisions and information loss, the computation of the column index follows Eq. (2.17), and the computation of the row index is optimized to:

$$u = \begin{cases} \left\lfloor 0.5 \cdot H \cdot \frac{\theta - \theta_{\text{up}}}{\theta_{\text{mid}} - \theta_{\text{up}}} \right\rfloor & \text{if } \theta < \theta_{\text{mid}} \\ \left\lfloor 0.5 \cdot H \cdot \left(1 + \frac{\theta - \theta_{\text{mid}}}{\theta_{\text{down}} - \theta_{\text{mid}}} \right) \right\rfloor & \text{otherwise} \end{cases}. \quad (4.2)$$

The decision boundary θ_{mid} is an intrinsic property of the sensor and separates the upper and lower half. Independent of the used projection, the result is a range image of size $H \times W \times 6$ with channels r, x, y, z , intensity ι , and an occupancy flag. The latter indicates if at least one point was projected into the respective cell. In case of collisions, the point with the smallest distance r is selected. The resulting tensor is the input to the range view backbone and overall network. The second, third, and fourth channel contain the projected Cartesian points and are used as $\mathbf{P}^{\text{RV}} \in \mathbb{R}^{H \times W \times 3}$ in the following chapters.

Backbone

The backbone is a fully convolutional network depicted in Fig. 4.1 whose architecture is motivated by Deep Layer Aggregation (DLA) [Yu18] and related to the backbone of LaserNet [Mey19b]. It consists of two different building

blocks, feature extractors (FEs) and feature aggregators (FAs) [Mey19b]. An extractor consists of B stacked residual blocks, illustrated in Fig. 4.2. The first convolutional layer of its first block downsamples the input by applying a stride $\mathbf{s} = (s_H, s_W)$ and optionally increases the number of channels. All remaining blocks of a feature extractor keep the resolution and channel size constant. Three feature extractors with $B = 4, 5,$ and 6 build the encoder of the network, see Fig. 4.1, and consecutively downsample the feature maps. The first and second feature extractor apply a stride of $\mathbf{s} = (1, 2)$ and horizontally downsample by a factor of two, whereas the last one with $\mathbf{s} = (2, 2)$ downsamples also vertically. The asymmetrical downsampling is motivated by the intrinsic properties of the deployed lidar sensors. Their horizontal resolution is roughly four times higher than their vertical. With its 30 layers in total, the encoder is comparable to a ResNet-34, however with only three stages.

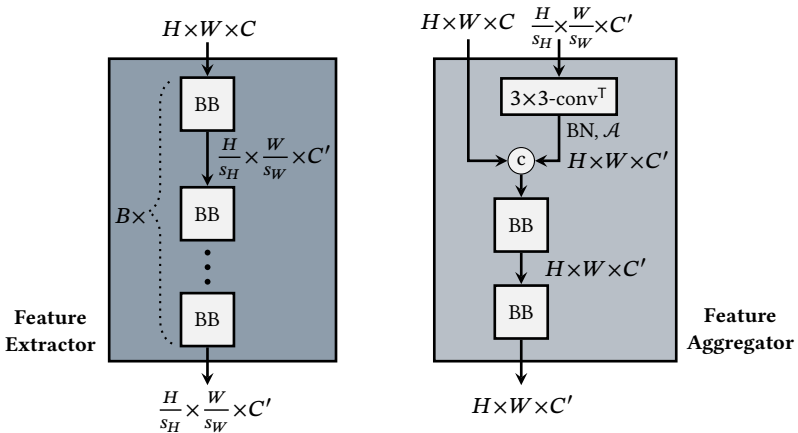


Figure 4.2: The building blocks of the range view backbone based on [Mey19b], which consist of Basic Blocks (BBs) [He16].

Feature aggregators, as the second building block, receive feature maps of two different stages and resolutions. They upsample their lower-resolution input with a transposed convolution, concatenate both inputs and apply two residual blocks. As a result, the upsampling of the lower-resolution feature maps with more context is guided by feature maps from the previous stage

with higher resolution and more spatial information. This strategy improves the combination of aggregated context with fine details. The decoder comprises four feature aggregators, depicted in Fig. 4.1. It uses the multi scale feature maps and context provided by three feature extractors to compute the final feature maps for the heads. As panoptic segmentation requires multiple heads, two feature aggregators are deployed in parallel at the end, one for the semantic head and another for the instance heads. This dual setup showed improved results for existing methods [Che20, Zho21].

Heads and Loss

The range view predictions for semantics, centers, and offsets are computed based on the backbone’s final feature maps and multiple parallel heads. The semantic head is a single 1×1 -convolution that computes the semantic segmentation. Both instance heads for the center and offset predictions have the same architecture consisting of a 3×5 -convolution which reduces the feature channels to 32, followed by BN, LReLU, and a final 1×1 -convolution. All predictions are horizontally upsampled by a factor of two to match the original input resolution. Afterwards, Non-Maximum Suppression (NMS) extracts the k_c range view cells with the highest scores as center candidates $\hat{\mathbf{u}} = (\hat{u}, \hat{v}) \in \mathcal{X}$ based on the center heatmap. In addition, the semantic segmentation determines the range image cells \mathbf{u} belonging to a thing class. Finally, the assignment \mathcal{C} allocates each of these cells to one of the center candidates $\hat{\mathbf{u}}$ based on the predicted offset vectors $\mathbf{O} \in \mathbb{R}^{H \times W \times 2}$:

$$\mathcal{C}(\mathbf{u}) = \arg \min_{\hat{\mathbf{u}} \in \mathcal{X}} \left(\left\| \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} O_{u,v,1} \\ O_{u,v,2} \end{bmatrix} - \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} \right\|_2 \right). \quad (4.3)$$

A k NN-based back-projection strategy [Mil19], introduced in Section 2.3.2, transforms the predicted 2D semantic and instance segmentation back to the 3D point cloud to provide the required 3D panoptic segmentation.

Since all heads are trained simultaneously, a multi task loss is required. The semantic head is trained with CE and Lovász loss [Ber18], the center head with MSE loss, and the offset head with MAE loss, leading to a weighted loss

term with weights λ :

$$L = \lambda_{\text{sem}} \cdot (L_{\text{CE}} + L_{\text{Lovász}}) + \lambda_{\text{ctr}} \cdot L_{\text{MSE}} + \lambda_{\text{off}} \cdot L_{\text{MAE}}. \quad (4.4)$$

4.2 Bird’s Eye View Network

The bird’s eye view is the second view which plays an essential role in the multi view architecture. Similar to the range view, the first step is a transformation of the point clouds into the desired input representation. The bird’s eye view input tensor is then fed to the backbone, which follows the same architecture as the range view backbone, with the only difference of a symmetric feature map downsampling, see Fig. 4.3. The final feature maps of the backbone are again provided to three parallel heads. A sparse semantic head and 3D clustering step provide the final 3D panoptic segmentation. Similar to the range view network, the loss function of Eq. (4.4) is used. The overall bird’s eye view architecture (BEVNet) is depicted in Fig. 4.3.

Input Representation

The point clouds are transformed into the polar bird’s eye view to create the input tensor. The motivation for a polar instead of a Cartesian representation lies in the underlying sensor properties discussed in Section 2.3. Its point density naturally diminishes with distance, and therefore, a polar grid is less sparse and requires a smaller number of cells to cover the required area. Every 3D point is projected to its corresponding bird’s eye view cell following Eq. (2.18). However, multiple points will be projected into most of the cells. While these collisions are undesirable for the range view and can ideally be avoided, it is unavoidable for the bird’s eye view since there is no one-to-one correspondence between cells and measurements. Hence, a learned function based on PointNet transforms the varying number of points inside each cell into a fixed-size feature vector, following the ideas of PointPillars [Lan19b] and PolarNet [Zha20c]. Its lightweight structure proposed in this approach is shown in Fig. 4.4.

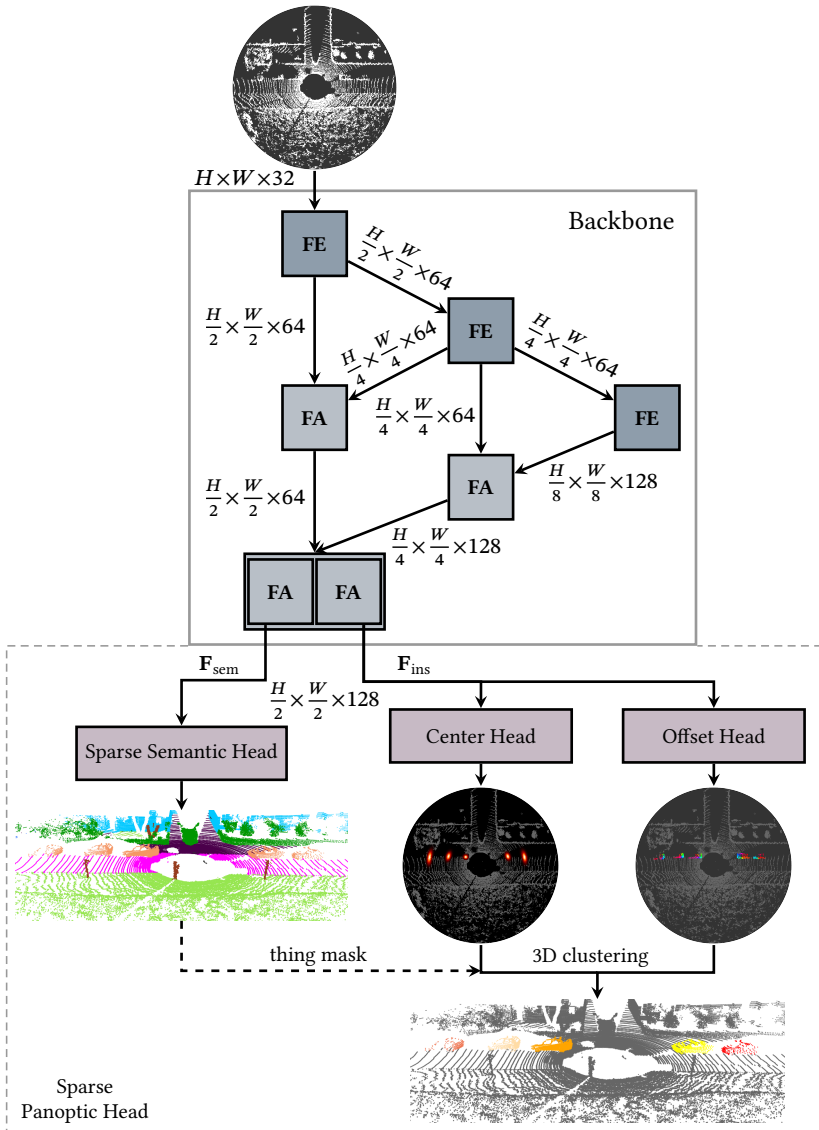


Figure 4.3: The bird's eye view network BEVNet. It is composed of a fully convolutional backbone and three different heads. The edges are labeled with the feature maps sizes.

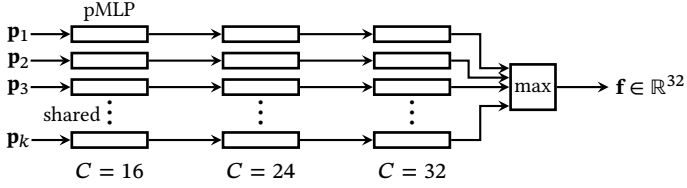


Figure 4.4: PointNet setup for computing a fixed-size feature vector for every grid cell based on the set of points $\{\mathbf{p}_1, \dots, \mathbf{p}_k\}$ which are projected into the considered cell. Every pMLP consists of a single layer and uses BN and LReLU.

Sparse Semantic Head

Current state-of-the-art approaches [Zha20c, Zho21] compute the semantic segmentation in the bird’s eye view, which is then transformed back to the point cloud to obtain a 3D semantic segmentation. Since a 2D bird’s eye view semantic segmentation leads to mediocre results, the classification layer of existing approaches predicts $D \cdot N_{\text{classes}}$ semantic class scores for every bird’s eye view cell, leading to a predicted segmentation $\mathbf{S} \in \mathbb{R}^{H \times W \times D \cdot N_{\text{classes}}}$. These can be interpreted as predictions for a vertical stack of voxels per bird’s eye view cell and results overall in a voxel-based prediction $\mathbf{S}^{\text{VX}} \in \mathbb{R}^{H \times W \times D \times N_{\text{classes}}}$, depicted on the left in Fig. 4.5.

However, this is quite expensive and ineffective due to the sparsity of the underlying voxel grid. Therefore, a novel sparse bird’s eye view head improves this final step by omitting empty cells and only considering cells with at least one point inside:

$$\begin{aligned} \hat{\mathcal{U}}^{\text{occupied}} = \{ \mathbf{u} \mid \mathbf{u} \in \{1, 2, \dots, H\} \times \{1, 2, \dots, W\} \wedge \\ \wedge \exists \mathbf{p} \in \mathcal{P}: (\mathcal{P}^{\text{BEV}} \circ \mathcal{Q}^z)(\mathbf{p}) = \mathbf{u} \}. \end{aligned} \quad (4.5)$$

The set of occupied cells and an arbitrary but fixed ordering defined by the bijective function

$$\mathcal{E} : \hat{\mathcal{U}}^{\text{occupied}} \rightarrow \{1, 2, \dots, N'\}, \quad N' = |\hat{\mathcal{U}}^{\text{occupied}}| \quad (4.6)$$

can be used to transform the 2D semantic feature maps \mathbf{F}_{sem} into a sparse representation. The transformation is based on the spatial feature map transformation \mathcal{S} introduced in Eq. (3.6):

$$\begin{aligned} \mathbf{F}_{\text{sem}}^{\text{sparse}} &= \mathcal{S}(\mathbf{F}_{\text{sem}}, \mathbf{U}^{\text{occupied}}), \\ \mathbf{U}^{\text{occupied}} &= [U_{n',d}] = [\mathcal{E}^{-1}(n')_d] \in \mathbb{N}^{N' \times 2}. \end{aligned} \quad (4.7)$$

Figure 4.5 visualizes the underlying idea and equations. Every row of the matrix $\mathbf{F}_{\text{sem}}^{\text{sparse}}$ contains the feature vector of a non-empty bird's eye view cell. The ordering \mathcal{E} associates the corresponding row to the 3D points when transforming the predictions back to the point cloud. It is worth mentioning that omitting the underlying grid structure is possible because the classification layer individually maps the final feature vectors to class scores without using the grid.

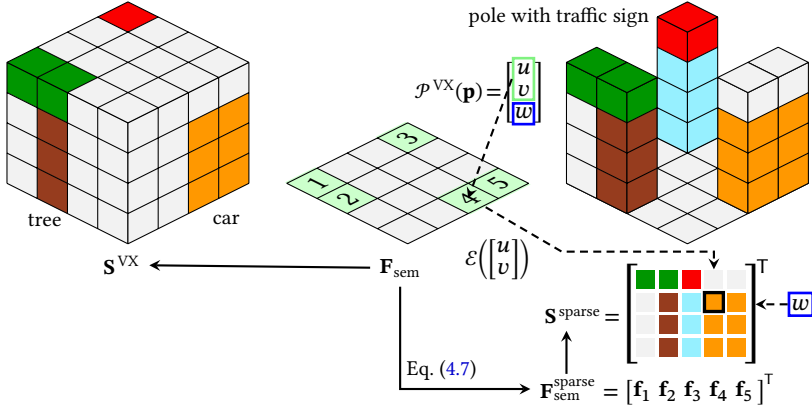


Figure 4.5: Idea of the sparse semantic head. A Cartesian grid instead of the deployed cylindrical grid ensures a clearer visualization. The additional dimension for the class scores is replaced by class colors for visualization. Empty cells or voxels are depicted in gray.

The classification layer of the deployed sparse head computes $D \cdot N_{\text{classes}}$ class scores for every occupied cell based on $\mathbf{F}_{\text{sem}}^{\text{sparse}}$, which provides a vertical stack of predictions $\mathbf{S}^{\text{sparse}} \in \mathbb{R}^{N' \times D \times N_{\text{classes}}}$. These can be transformed into a 3D

point cloud semantic segmentation by:

$$\text{sparse} \mathbf{S}^{\text{PV}} = \mathcal{S}(\mathbf{S}^{\text{sparse}}, \tilde{\mathbf{U}}^{\text{sparse}}) \in \mathbb{R}^{N \times N_{\text{classes}}}. \quad (4.8)$$

The index matrix $\tilde{\mathbf{U}}^{\text{sparse}}$ for the sparse representation is based on voxel coordinates for the cylindrical voxel view

$$\mathbf{U}^{\text{VX}} = [U_{n,d}] = [(\mathcal{P}^{\text{VX}} \circ \mathcal{Q}^{\text{z}})(\mathbf{p}_n)_d] \in \mathbb{N}^{N \times 3}, \quad (4.9)$$

and is computed as follows:

$$\tilde{\mathbf{U}}^{\text{sparse}} = [\tilde{U}_{n,d}], \quad \tilde{U}_{n,1} = \mathcal{E}\left(\begin{bmatrix} U_{n,1} \\ U_{n,2} \end{bmatrix}\right), \quad \tilde{U}_{n,2} = U_{n,3}. \quad (4.10)$$

The ordering \mathcal{E} identifies for every point the row in $\mathbf{S}^{\text{sparse}}$ based on the first two voxel coordinates, as illustrated in Fig. 4.5. These correspond to the bird’s eye view coordinates, which cannot be directly used since the underlying grid structure was abandoned. Simultaneously, the third voxel coordinate $U_{n,3}$ identifies the column and corresponds to the voxel position in the vertical stack. Overall, the sparse segmentation head significantly reduces inference and training time as well as memory demands due to the sparsity of approximately 86 %.

Instance Heads

Both instance heads consist of a 3×3 -convolution which reduces the number of feature channels to 32, followed by BN, LReLU, and a final 1×1 -convolution. In contrast to the semantic head, expanding the predictions to voxel-based predictions is unnecessary since the clustering is based on 2D centers and offsets. Furthermore, the employed instance clustering directly considers and clusters the individual 3D points instead of bird’s eye view cells. As a result, the explicit computation of a 2D instance segmentation can be skipped, which avoids processing the empty cells. The first step of the clustering is again NMS to extract the k_c center candidates $\hat{\mathbf{u}}$ with the highest score. Afterwards, the

assignment \mathcal{C} allocates every 3D point \mathbf{p}_n to one of the extracted center candidates $\hat{\mathbf{u}}$ based on the offset vectors \mathbf{O} and the point’s projection index \mathbf{u}_n :

$$\mathcal{C}(\mathbf{u}_n) = \arg \min_{\hat{\mathbf{u}} \in \mathcal{K}} \left(\left\| \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} O_{u,v,1} \\ O_{u,v,2} \end{bmatrix} - \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} \right\|_2 \right). \quad (4.11)$$

Consequently, the combined predictions of the sparse semantic and the instance heads provide the final 3D panoptic segmentation.

4.3 Multi View Network

The main drawback of the presented approaches is the focus on an individual view. Depending on the chosen view, these approaches suffer from different weaknesses. Consequently, a multi view framework [Due22] is proposed, which combines distinct views and addresses these drawbacks to improve the predicted 3D panoptic segmentation. It builds upon three distinct representations, the 2D range and bird’s eye view combined with the unstructured point view. The backbones of the previously introduced single view networks are two primary components of the multi view network and extract feature maps of the point cloud represented in range and bird’s eye view. Their 2D grid structure allows for efficient feature and context aggregation, and as a result of the different projections, both views contribute valuable features based on different 2D neighborhoods. A novel point view backbone, illustrated in Fig. 4.6, is responsible for the vital combination of features across views. These multi view features are fused at feature level inside the point view backbone, instead of a simple late fusion step, to leverage their full potential. The panoptic head also exploits the multi view setup and provides semantic and instance predictions in different views. On the one hand, it predicts a 3D semantic segmentation based on the pointwise multi view features. On the other hand, it builds upon bird’s eye view feature maps for bottom-up instance segmentation, which is most suitable for a dense heatmap prediction. Furthermore, it allows the range and point view to focus on semantic features. The overall architecture of the multi view approach MVNet is shown in Fig. 4.6.

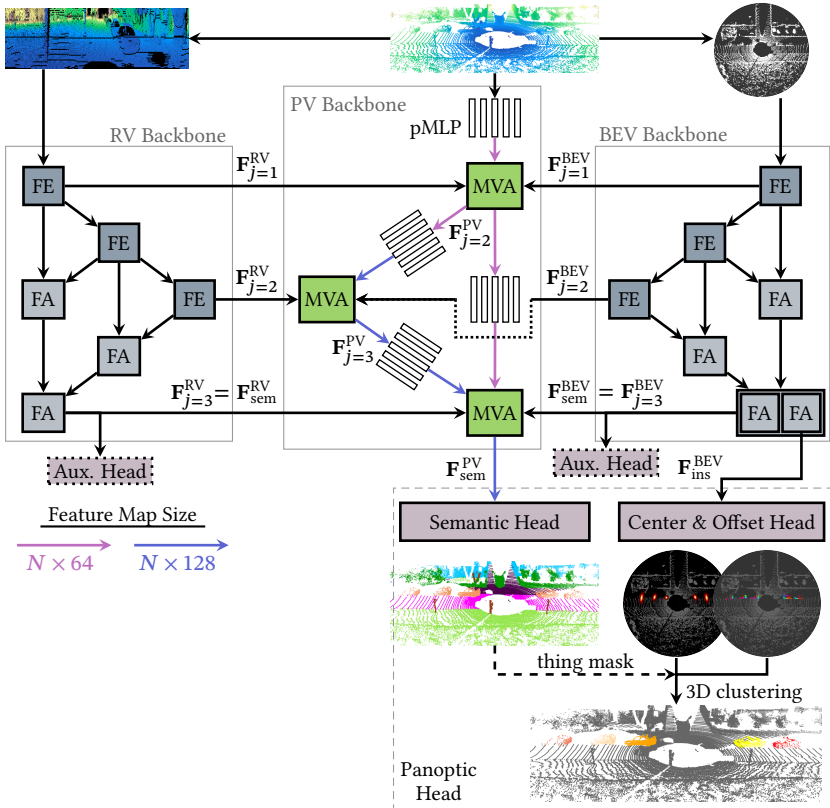


Figure 4.6: Architecture of the multi view framework MVNet based on range view, bird's eye view, and point view.

Point View Backbone

The key element and important link between the range and bird's eye view is the point view backbone. Its overall structure mimics the architecture of the 2D backbones. However, it omits some cross-connections since it deploys no feature aggregators, which require two feature map inputs of different scales. Consequently, the architecture is more related to the U-Net family [Ron15] with a single skip connection. The backbone itself consists of two elements,

pMLPs and Multi View Aggregation (MVA). The single-layer pMLPs refine the pointwise features, while the MVA is the actual link between the backbones, combining and fusing the features across three different views. This architecture is flexible and can be deployed in different configurations to decide for each block of the backbone whether a multi view fusion should be applied. Depending on the choice, an MVA module or pMLP is deployed. The setup shown in Fig. 4.6 deploys multi view aggregation three times. At the beginning, to collect low level features with little context but high spatial resolution, after the last feature extractor to gather features with strong context but reduced spatial resolution, and at the end to exploit the final feature maps, which contain the aggregated context and have a high spatial resolution. Another evaluated configuration is the aggregation after every block, which replaces the remaining three core pMLPs with MVA modules. Independently of the chosen configuration, the last MVA module receives two point view inputs which are concatenated and processed by a pMLP prior to the aggregation.

The multi view aggregation follows a two-step process to aggregate features across three views. The first and most important step is the transformation of feature maps from range and bird’s eye view to point view. This step is required to provide the aggregated context for every 3D point \mathbf{p}_n in the point cloud. Based on Eq. (3.8), the 2D bird’s eye view feature maps are transformed following

$$\begin{aligned} {}^{\text{BEV}}\mathbf{F}_j^{\text{PV}} &= \mathcal{S}(\mathbf{F}_j^{\text{BEV}}, \mathbf{U}_j^{\text{BEV}}) \in \mathbb{R}^{N \times C_{\text{BEV}}}, \\ \mathbf{U}_j^{\text{BEV}} &= [(\mathcal{P}_j^{\text{BEV}} \circ \mathcal{Q}^z)(\mathbf{p}_n)_d]. \end{aligned} \quad (4.12)$$

The range view feature maps are similarly transformed:

$$\begin{aligned} {}^{\text{RV}}\mathbf{F}_j^{\text{PV}} &= \mathcal{S}(\mathbf{F}_j^{\text{RV}}, \mathbf{U}_j^{\text{RV}}) \in \mathbb{R}^{N \times C_{\text{RV}}}, \\ \mathbf{U}_j^{\text{RV}} &= [(\mathcal{P}_j^{\text{RV}} \circ \mathcal{Q}^\theta)(\mathbf{p}_n)_d]. \end{aligned} \quad (4.13)$$

The projections \mathcal{P} depend on the level j due to different feature map sizes. After this step, three feature maps are present in the point view while originating from different views. The second step fuses these into a combined

feature map for the next level $j + 1$:

$$\mathbf{F}_{j+1}^{\text{PV}} = \mathcal{F}_{\text{MV}}(\mathbf{F}_j^{\text{PV}}, {}^{\text{RV}}\mathbf{F}_j^{\text{PV}}, {}^{\text{BEV}}\mathbf{F}_j^{\text{PV}}). \quad (4.14)$$

Four different fusion strategies \mathcal{F}_{MV} with fixed and learnable fusion operations are proposed and investigated. These are addition, elementwise maximum, concatenation followed by a 1×1 -convolution, and a weighted sum with learnable parameters $\mathbf{W}^\nu \in \mathbb{R}^{C_{\text{PV}} \times 3}$ and weight vectors $\boldsymbol{\gamma}^\nu$:

$$\begin{aligned} \mathbf{F}_{j+1}^{\text{PV}} &= \sum_{\nu} \text{diag}(\boldsymbol{\gamma}^\nu) \cdot {}^\nu\mathbf{F}_j^{\text{PV}}, \quad \nu \in \{\text{PV}, \text{RV}, \text{BEV}\}, \\ [\boldsymbol{\gamma}^{\text{PV}} \ \boldsymbol{\gamma}^{\text{RV}} \ \boldsymbol{\gamma}^{\text{BEV}}] &= \sigma_{\text{softmax}}\left(\sum_{\nu} {}^\nu\mathbf{F}_j^{\text{PV}} \cdot \mathbf{W}^\nu\right) \in \mathbb{R}^{N \times 3}. \end{aligned} \quad (4.15)$$

The softmax function is applied row-wise. All strategies apply a final pMLP in the end and require $C_{\text{PV}} = C_{\text{RV}} = C_{\text{BEV}}$ to be applicable, except for the concatenation. The steps and setup of the MVA module are illustrated in Fig. 4.7.

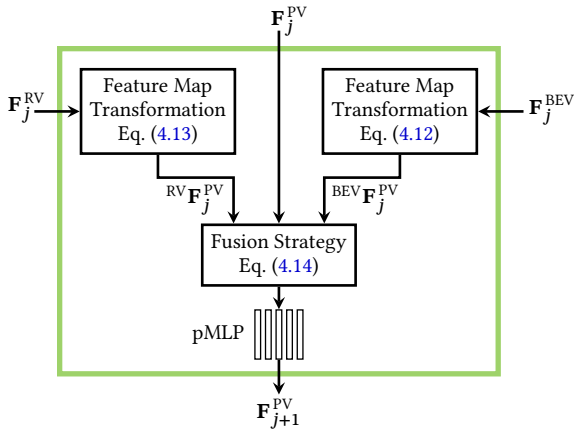


Figure 4.7: Setup and steps of the multi view aggregation module. Range and bird's eye view feature maps are transformed into the point view. Afterwards, the features originating from three different views are fused.

Multi View Panoptic Head and Loss

The semantic head consists of a pMLP, which maps the final pointwise features $\mathbf{F}_{\text{sem}}^{\text{PV}}$ to semantic class scores. On the other hand, the offset and center head are applied to bird’s eye view feature maps $\mathbf{F}_{\text{ins}}^{\text{BEV}}$ and have the same structure as the respective heads of the bird’s eye view network BEVNet. In analogy to the range and bird’s eye view network, MAE loss is applied to the predicted offsets and MSE loss to the predicted center heatmap. The semantic loss is adapted to the multi view setup. Instead of applying the semantic loss $L_{\text{sem}} = L_{\text{CE}} + L_{\text{Lovász}}$ only to the predicted semantic segmentation of the point backbone, auxiliary semantic heads are added to the range and bird’s eye view backbone during training, depicted in Fig. 4.6. These heads are equal to the semantic heads presented for the range and bird’s eye view approaches. As a result, the multi view architecture benefits from the previously presented sparse semantic bird’s eye view head too, which speeds up the multi view training. An auxiliary loss is computed based on their predictions to support both backbones to learn meaningful features. Additionally, the auxiliary loss prevents the bird’s eye view backbone from focusing too much on the instance segmentation. The overall loss used to train the multi view framework is then defined as:

$$L = \lambda_{\text{sem}}^{\text{RV}} \cdot L_{\text{sem}}^{\text{RV}} + \lambda_{\text{sem}}^{\text{BEV}} \cdot L_{\text{sem}}^{\text{BEV}} + \lambda_{\text{sem}}^{\text{PV}} \cdot L_{\text{sem}}^{\text{PV}} + \lambda_{\text{ctr}} \cdot L_{\text{MSE}}^{\text{BEV}} + \lambda_{\text{off}} \cdot L_{\text{MAE}}^{\text{BEV}}. \quad (4.16)$$

Data Augmentation

Pointwise tasks usually suffer from an imbalanced class distribution, since points of classes, such as road or building, occur much more frequently than points of small ones, such as pedestrian or cyclist. One strategy for 3D panoptic segmentation to mitigate this imbalance is the extraction of instances of these classes across the training set to create an instance database. During training, instances from this database are randomly pasted into the 3D point clouds. This data augmentation technique [Zha20c, Xu21a] is called random object augmentation (ROA) in the following and is further improved in this thesis to reduce the domain gap [Due22] as follows:

- Instances are only inserted above their respective ground classes. Four-or-more-wheelers such as cars, trucks, or buses are placed above the road or parking areas, two-wheelers above the road and sidewalk, and pedestrians above the sidewalk.
- Collisions with other points are avoided. Therefore, positions that include other points inside the axis-aligned bounding box of the inserted instance are excluded.
- Instances are inserted close to their original distance and with the same relative orientation to minimize the deviation from the sensor measurement pattern.

The second step is approximated by using a fixed bounding box size for each class, which allows pre-computing valid positions for each point cloud and class. To account for the class imbalance, the probability of a class being chosen is inversely proportional to its point frequency.

5 Temporal Panoptic Segmentation of 3D Point Clouds

The multi view approach presented in the previous chapter focuses on improving 3D panoptic segmentation based on the point cloud information of the current time step. The temporal framework [Due20a] proposed in this chapter goes one step further and exploits temporal information and dependencies in point cloud sequences. These are the result of the consecutive and repeated sensor measurements performed by autonomous vehicles or robots. The proposed recurrent fully convolutional approach aggregates and memorizes information over time to improve the predictions for the latest point cloud based on past information. It builds upon a novel recurrent temporal feature fusion for 2D feature maps, which extends range and bird’s eye view approaches with a temporal memory to exploit past feature maps. Subsequently, the multi view architecture is extended by the temporal fusion in range and bird’s eye view to combine multi view and temporal benefits.

5.1 Temporal Range View

Temporal approaches use and exploit previous predictions, feature maps, or input data to improve current predictions, which distinguishes them from single frame approaches considering only the latest frame. In order to develop a beneficial temporal fusion strategy and architecture for the range view, it is essential to consider the requirements of the targeted domain of real-time autonomous systems. In this domain, sensors provide new recordings every Δt and the autonomous system requires new predictions with low latency for

these. The proposed recurrent architecture with recursive feature fusion addresses the drawbacks of overlapping and fixed-size temporal windows discussed in Section 3.2 for recurring recordings. It builds upon RNNs, which compute their output based on the input and previous outputs and resemble infinite impulse response filters. As a result, the underlying idea is based on two elements, a single frame backbone and a hidden state or temporal memory. The single frame backbone was presented in Section 4.1 and computes feature maps \mathbf{F}_t^{RV} for the latest input range image to provide the information extracted from the current point cloud. The temporal memory \mathbf{H}_t , on the other hand, contains the temporal feature maps recursively aggregated over time. The recursive update step \mathcal{H} combines both elements and updates the temporal memory with the latest feature maps provided by the single frame backbone. It can generally be formulated as:

$$\mathbf{H}_t = \mathcal{H}(\mathbf{F}_t^{\text{RV}}, \mathbf{H}_{t-1}) = \mathcal{H}(\mathbf{F}_t^{\text{RV}}, \mathcal{H}(\mathbf{F}_{t-1}^{\text{RV}}, \mathbf{H}_{t-2})) = \dots \quad (5.1)$$

One significant advantage of the recursive update is the reuse of all previous feature computations. When a new recording arrives, the backbone computes \mathbf{F}_t^{RV} , analogous to the single frame approach. The temporal memory \mathbf{H}_{t-1} has already been computed in the previous time step and is reused. Therefore, the computational effort is only increased by the temporal update \mathcal{H} , which is performed once in every time step. This increase is independent of the processed sequence length, and no fixed temporal window size or trade-off between exploited past frames and computational effort is necessary. The number of considered past frames is potentially unlimited since their information is aggregated in the temporal memory. The network learns to integrate or forget information as part of the training. One remaining challenge is the alignment of feature maps between two time steps. The backbone extracts the feature maps \mathbf{F}_t^{RV} in the range view defined relative to the current ego position. However, the latter constantly changes due to ego motion. Consequently, a spatial transformation has to recursively transform the temporal memory \mathbf{H}_{t-1} from the range view of the last to the range view of the current ego position.

The proposed recurrent fully convolutional architecture T-RVNet is depicted in Fig. 5.1, where the temporal alignment is applied prior to the update step. Different alignment and update strategies are proposed and discussed in the following. Since the single frame backbone returns semantic and instance feature maps, the temporal pipeline and memory are required twice, illustrated in Fig. 5.1. The panoptic head and loss are inherited from the range view network of Section 4.1. The proposed temporal training strategy trains the temporal framework on short data sequences, comprising several tens of frames. This strategy ensures the presence of temporal dependencies and simultaneously retains a significant variation in the data.

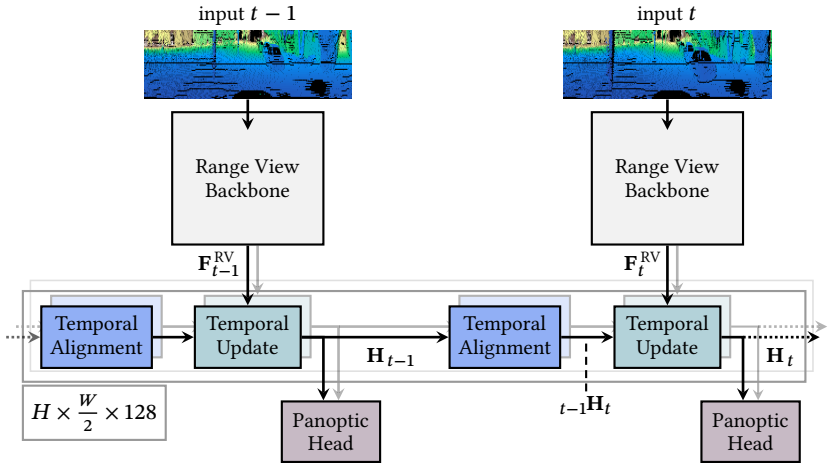


Figure 5.1: The recurrent temporal architecture T-RVNet with its components and unrolled for two time steps. The temporal memory \mathbf{H} has the same spatial dimensions and channel size as the backbone feature maps \mathbf{F}^{RV} . The temporal pipeline is required twice and applied to the semantic features maps $\mathbf{F}_{\text{sem}}^{RV}$ and instance feature maps $\mathbf{F}_{\text{ins}}^{RV}$.

5.1.1 Temporal Memory Alignment

The temporal memory alignment enables the recurrent architecture and recursive memory update depicted in Fig. 5.1. It addresses the underlying challenge

of transforming the temporal memory \mathbf{H}_{t-1} into the current range view, defined relative to the current ego pose, prior to the memory update. The transformation into the latest range view is based on Eqs. (3.9) and (3.11):

$${}_{t-1}\mathbf{H}_t = \hat{\mathcal{S}}(\mathbf{H}_{t-1}, \mathbf{V}_{t-1}). \quad (5.2)$$

Afterwards, the transformed memory ${}_{t-1}\mathbf{H}_t$ is spatially aligned with \mathbf{F}_t^{RV} . Two strategies for the computation of the index tensor \mathbf{V}_{t-1} are introduced, the backward (bwd) strategy

$$\begin{aligned} \mathbf{V}_{t-1}^{\text{bwd}} &= [V_{u,v,d}] = [\mathcal{J}_{t \rightarrow (t-1)}^{\text{RV}}(\mathbf{g}_{u,v})_d], \quad \mathbf{G} = \mathbf{P}_t^{\text{RV}}, \\ \mathcal{J}_{t \rightarrow (t-1)}^{\text{RV}}(\mathbf{g}) &= (\mathcal{P}^{\text{RV}} \circ \mathcal{Q}^\theta)(\mathbf{T}_{t \rightarrow (t-1)} \cdot \mathbf{g}) \end{aligned} \quad (5.3)$$

and the forward (fwd) strategy:

$$\begin{aligned} \mathbf{V}_{t-1}^{\text{fwd}} &= \text{reverse}(\mathbf{V}'_t), \\ \mathbf{V}'_t &= [V'_{u,v,d}] = [\mathcal{J}_{(t-1) \rightarrow t}^{\text{RV}}(\mathbf{g}_{u,v})_d], \quad \mathbf{G} = \mathbf{P}_{t-1}^{\text{RV}}, \\ \mathcal{J}_{(t-1) \rightarrow t}^{\text{RV}}(\mathbf{g}) &= (\mathcal{P}^{\text{RV}} \circ \mathcal{Q}^\theta)(\mathbf{T}_{(t-1) \rightarrow t} \cdot \mathbf{g}). \end{aligned} \quad (5.4)$$

The transformation $\hat{\mathcal{S}}$ used with the forward strategy natively transforms from t to $t-1$. However, the temporal memory must be transformed from $t-1$ to t . Therefore, the native index tensor \mathbf{V}'_t , which contains index \mathbf{u}' in cell \mathbf{u}^* , must be reversed, see Fig. 5.2. Afterwards, the new index tensor contains index \mathbf{u}^* in cell \mathbf{u}' , which allows $\hat{\mathcal{S}}$ to access the previous temporal memory \mathbf{H}_{t-1} at \mathbf{u}^* and moves its content to cell \mathbf{u}' in the temporally aligned memory ${}_{t-1}\mathbf{H}_t$.

At first glance, both strategies look rather similar. However, there is an important difference in the index tensors \mathbf{V}^{bwd} and \mathbf{V}^{fwd} . On the one hand, the backward strategy computes for every cell of the *current* time step where its contained 3D point would have been in the last time step, illustrated in Fig. 5.2. It transforms the 3D points \mathbf{P}_t^{RV} from the current to the previous ego pose, followed by a range view projection. On the other hand, the forward strategy considers for every cell of the *previous* time step where its content would be in the current time step. It transforms the points $\mathbf{P}_{t-1}^{\text{RV}}$ of the previous time

step to the current ego pose, followed by a range view projection. The underlying assumption in both cases is that the cell \mathbf{u}^* in the previous and the cell \mathbf{u} in the current range view contain spatially close measurements. The notable difference between the backward and forward strategy is that the former uses \mathbf{P}_t^{RV} and the latter $\mathbf{P}_{t-1}^{\text{RV}}$ to associate the cells \mathbf{u}^* and \mathbf{u} . Consequently, forward and backward association is sometimes asymmetric:

$$\exists \mathbf{u} : \mathcal{J}_{t \rightarrow (t-1)}^{\text{RV}}(\mathbf{p}_{\mathbf{u}, t}) = \mathbf{u}^* \wedge \mathcal{J}_{(t-1) \rightarrow t}^{\text{RV}}(\mathbf{p}_{\mathbf{u}^*, t-1}) = \mathbf{u}' \neq \mathbf{u}, \quad (5.5)$$

where $\mathbf{p}_{\mathbf{u}, t}$ is the 3D point that was projected into cell \mathbf{u} at time t and $\mathbf{p}_{\mathbf{u}^*, t-1}$ was projected into cell \mathbf{u}^* at time $t-1$. The underlying reason for the asymmetry is that $\mathbf{p}_{\mathbf{u}, t} \neq \mathbf{p}_{\mathbf{u}^*, t-1}$. In general, this is expected since a lidar sensor never records the exact 3D point in the current and last time step. Most of the time, these points are spatially close, and \mathbf{u}' equals \mathbf{u} or is an adjacent cell, depending on the quantization. In this case, the spatial deviations are small, and the temporal alignment works well without considerable differences between both strategies.

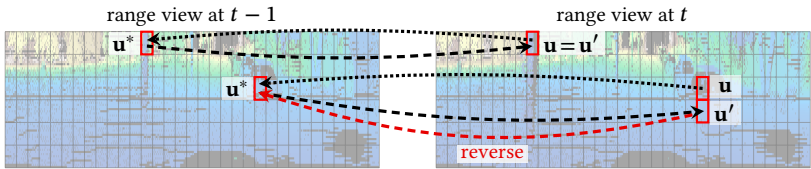


Figure 5.2: Association of range view cells across time, based on the backward (dotted) and forward strategy (dashed). In the first example, both strategies are symmetric, while in the second, they provide different pairs of associated cells.

However, there are additional and more severe reasons for $\mathbf{p}_{\mathbf{u}, t} \neq \mathbf{p}_{\mathbf{u}^*, t-1}$, shadowing and moving objects. They have different causes but a similar impact, with an example for shadowing shown in Fig. 5.3. In scenario (a), the point \mathbf{p}_t of a pole is measured. This point is projected into cell \mathbf{u} of the latest range view and into cell \mathbf{u}^* when transformed and projected into the range view of the previous ego pose and time step. However, this pole was invisible in the previous time step since a building was in the line of sight. Hence,

the point $\mathbf{p}_{\mathbf{u}^*, t-1}$, which was actually recorded and projected into cell \mathbf{u}^* , lies on the corner of the building, with a significant spatial distance to $\mathbf{p}_{\mathbf{u}, t}$. Scenario (b) shows the opposite setup, where the corner of the building recorded in the previous time step is no longer visible because the pole shadows it. While moving objects cause similar effects, the reason there is predominantly the movement of the objects themselves and not the change of sensor view-point induced by ego motion.

The forward and backward strategies are prone to different errors in these scenarios. The forward strategy generates wrong associations in scenarios similar to (b), where an obstacle or moving object hides previously recorded areas. Since it transforms the hidden point \mathbf{p}_{t-1} , it wrongly matches \mathbf{u} and \mathbf{u}^* . The backward strategy, on the other hand, handles these scenarios well and only associates cell \mathbf{u} with \mathbf{u}' since it relies on \mathbf{p}_t . However, it struggles in scenarios where previously hidden areas become visible, such as scenario (a), and incorrectly associates \mathbf{u} and \mathbf{u}^* . As a counterpart, the forward strategy can be applied successfully in these scenarios.

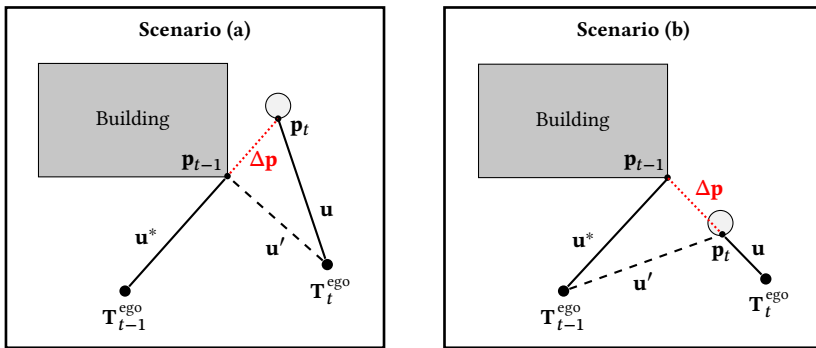


Figure 5.3: Two different scenarios illustrate a wrong association for either the backward (a) or forward (b) strategy.

The examples and discussion above show that scenarios exist for both strategies where wrong cells get associated across time. While it is possible to leave it up to the network to learn to handle this kind of error, further investigation

is beneficial. Therefore, a detection mechanism is proposed to detect wrong associations based on the spatial distance $\Delta \mathbf{p}$ of expected and measured 3D points, illustrated in Fig. 5.3. Based on the decision criterion $\Delta \mathbf{p} \leq \delta_{\text{range}}$, the affected cells of the temporal memory can be discounted with $\xi \in [0, 1]$ or explicitly deactivated. Based on this mechanism, combining both strategies and choosing the strategy with the smallest $\Delta \mathbf{p}$ for each cell is also possible.

5.1.2 Temporal Memory Update

The temporal memory update fuses the latest feature maps with the aligned temporal memory. Initially, RNNs in neural language processing used LSTMs or GRUs to combine the temporal memory, or hidden state, with the current input. Both approaches emerged to address the vanishing or exploding gradient problem and have also been adapted to the 2D image domain by replacing fully connected layers with convolutional layers [Shi15, Sia17]. Although LSTMs have more gates and thus more parameters, GRUs showed similar performance [Chu14], and no clear advantage of any of the approaches could be demonstrated. Hence, the investigated baseline fusion strategy is based on a ConvGRU, introduced in Section 2.1.3.

However, one drawback of the established ConvGRU is that it considers only a small spatial context based on a single convolutional layer. In fact, ConvGRUs aggregate no context at all if a 1×1 -convolution is used. This section proposes improved strategies to address this drawback. These can generally be grouped into two categories, gated strategies based on extended ConvGRUs and residual non-gated strategies. The gated update strategy is an enhanced ConvGRU, called ContextGRU and illustrated in Fig. 5.4. It aggregates sophisticated context based on the temporal memory and current feature maps. A small residual network is applied to the concatenation of the temporal memory ${}_{t-1}\mathbf{H}_t$ and the latest feature maps \mathbf{F}_t^{RV} to aggregate combined context for a significantly improved candidate memory $\tilde{\mathbf{H}}_t$. The chosen location also ensures that the gradient flow through time stays untouched to prevent reintroducing exploding or vanishing gradients.

The residual update strategies use no gating mechanisms but rely on a concatenation followed by a residual network. At first glance, these approaches are prone to the vanishing or exploding gradient problem. The number of layers the gradient has to pass increases with every additional time step the gradient is backpropagated. Nevertheless, a residual strategy is promising since residual networks were designed as very deep networks with many stacked layers. Additionally, gradients will usually be backpropagated only a few time steps in the considered context. The reasons are twofold. First, the last few frames contain the most valuable information, which naturally diminishes with temporal distance. Therefore, backpropagating the gradient dozens or even hundreds of time steps adds no significant benefit. Secondly, computational and especially memory demands allow only a few steps and up to ten in this thesis. Based on this discussion, the maximum depth for the backpropagation can be computed. Considering ten steps in time and a residual network consisting of four BBs for the update, the maximum number of layers from output to input are 118 convolutional layers. Since residual networks with up to 152 layers are frequently used [He16, Zha17, Yu18], the residual update strategy is well-suited and promising.

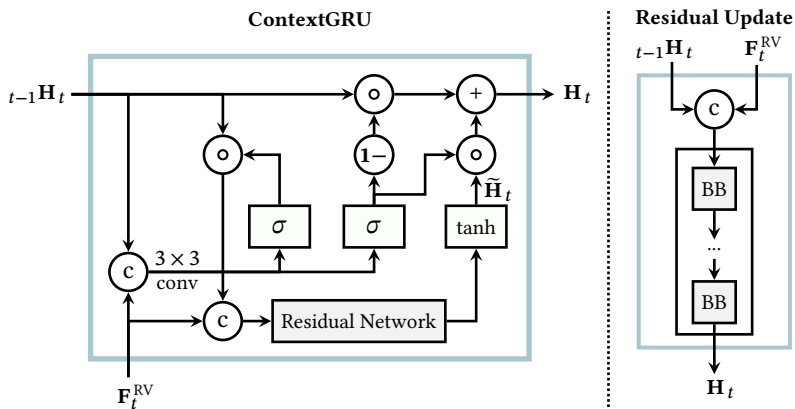


Figure 5.4: Update strategies for the temporal memory based on a gated ContextGRU and a residual network. The ContextGRU extends the original GRU, illustrated in Fig. 2.7.

5.1.3 Temporal Training

The underlying training strategy for recurrent architectures differs significantly since temporal dependencies only exist for a sequence of consecutive point clouds. On the one hand, this requires datasets that contain sequential data. On the other hand, training with randomly drawn samples from the dataset, which is a common strategy, no longer works. A straightforward strategy provides the data sequentially based on the native sequences in the underlying dataset. However, this strategy has several drawbacks:

- Native sequences often contain very similar data, e.g., driving for several minutes on a highway. Training continuously on long sequences results in updating the network repeatedly and for many iterations on similar data.
- Variation between epochs is relatively small since only the native sequences can be shuffled, which is especially critical for datasets with few but long sequences.

The proposed sequence-based training strategy addresses these challenges by training the temporal approaches with subsequences of ζ point clouds. Different sequence lengths $\zeta \in \{25, 50, 100\}$ are investigated. This strategy significantly improves the variety because every ζ frames a new and, in most cases, distinct subsequence is randomly drawn. Additionally, the variation between epochs is considerably improved since the number of sequences to shuffle is usually two or three orders of magnitude higher with short subsequences than with native sequences. The restriction of the sequence length is a negligible limitation since the most valuable temporal dependencies are short-term. Point clouds which are minutes or hundreds of meters away contain little relevant information for the current frame.

Alongside the training strategy, also the gradient flow differs from optimizing a single frame network. Predictions not only depend on the current but also on previous input and computations. As a result, error and gradients can be back-propagated through time. Thereby, the single frame backbone learns to compute valuable feature maps for the current *and* future frames. Additionally,

the memory update learns to combine current and past information. Truncated Backpropagation Through Time (TBPTT) [Wil90] makes gradient propagation through time computationally manageable by truncating it after κ_2 steps. While the training with subsequences already increases the training variation, they still contain similar data. Therefore, the proposed strategy updates the weights only every κ_1 steps to reduce the number of weight updates on strongly related data. These sparse updates also enable training on datasets, where only every κ_1 -th point cloud has labels. Since training on a subsequence starts with a zero-filled memory, the first weight update is delayed for κ_3 steps. The delay allows the memory to aggregate meaningful temporal information prior to the first weight update. The zero-filled memory is also the initial state for inference. An overview of the presented temporal training configuration is shown in Fig. 5.5.

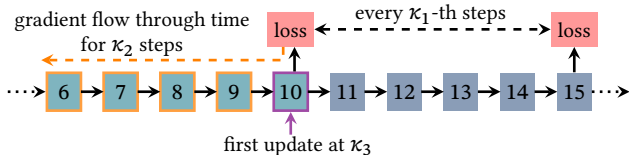


Figure 5.5: The proposed temporal training strategy [Due20a]. After a warm-up phase of κ_3 steps to fill the initially empty memory, loss and weight updates are computed every κ_1 steps. The gradient is propagated κ_2 past frames back through time before it is truncated.

5.2 Temporal Bird’s Eye View

The presented recurrent temporal fusion with recursive feature map transformation is not restricted to the range view. It can extend arbitrary approaches relying on grid-based views as long as a unique 3D position can be assigned to the cells of the respective view. Therefore, this concept can also be applied to the bird’s eye view to create a similar recurrent temporal architecture, depicted in Fig. 5.6 and called T-BEVNet. It extends the bird’s eye view network introduced in Section 4.2, which consists of the presented bird’s eye

view backbone and the sparse panoptic head. The latter comprises the introduced sparse semantic, offset, and center head. The notable difference to the temporal range view approach concerns the temporal alignment step, which needs to transform polar bird's eye view cells across time instead of range view cells. The memory update and temporal training strategies are equal to the strategies presented in the previous section.

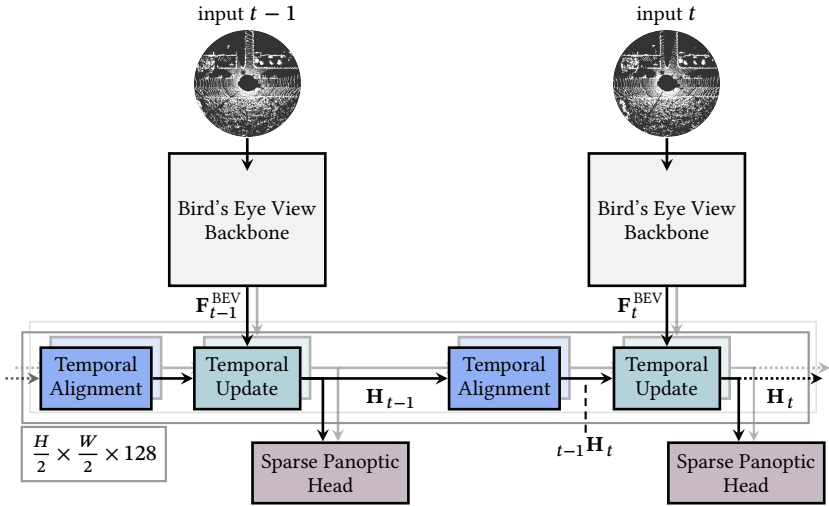


Figure 5.6: The recurrent temporal bird's eye view T-BEVNet architecture unrolled for two time steps. The temporal pipeline is required twice and applied to the semantic feature maps $\mathbf{F}_{\text{sem}}^{\text{BEV}}$ and instance feature maps $\mathbf{F}_{\text{ins}}^{\text{BEV}}$.

Temporal Memory Alignment

In general, the temporal alignment for the bird's eye view feature maps follows Eq. (5.2). The difference between the range and bird's eye view alignment originates from the definition of the 3D cell positions. Bird's eye view cells usually contain multiple points and simultaneously have a unique 3D position, independently of the contained data. These properties motivate the choice of the cell center as 3D position. On the other hand, a range view cell has no 3D position without considering the data since empty cells do not have

an assigned distance. Therefore, the backward strategy

$$\begin{aligned} \mathbf{V}_{t-1}^{\text{bwd}} &= [V_{u,v,d}] = [\mathcal{J}_{t \rightarrow (t-1)}^{\text{BEV}}(\mathbf{g}_{u,v})_d], \quad \mathbf{G} = \mathbf{G}^{\text{BEV}}, \\ \mathcal{J}_{t \rightarrow (t-1)}^{\text{BEV}}(\mathbf{g}) &= (\mathcal{P}^{\text{BEV}} \circ \mathcal{Q}^z)(\mathbf{T}_{t \rightarrow (t-1)} \cdot \mathbf{g}) \end{aligned} \quad (5.6)$$

and forward strategy

$$\begin{aligned} \mathbf{V}_{t-1}^{\text{fwd}} &= \text{reverse}(\mathbf{V}'_t), \\ \mathbf{V}'_t &= [V'_{u,v,d}] = [\mathcal{J}_{(t-1) \rightarrow t}^{\text{BEV}}(\mathbf{g}_{u,v})_d], \quad \mathbf{G} = \mathbf{G}^{\text{BEV}}, \\ \mathcal{J}_{(t-1) \rightarrow t}^{\text{BEV}}(\mathbf{g}) &= (\mathcal{P}^{\text{BEV}} \circ \mathcal{Q}^z)(\mathbf{T}_{(t-1) \rightarrow t} \cdot \mathbf{g}) \end{aligned} \quad (5.7)$$

rely on the 3D positions $\mathbf{G}^{\text{BEV}} = [G_{u,v,d}]$ of the bird's eye view cells. These are defined by their centers:

$$\begin{aligned} G_{u,v,1} &= r_{u,v} \cdot \cos(\phi_{u,v}), \quad G_{u,v,2} = r_{u,v} \cdot \sin(\phi_{u,v}), \quad G_{u,v,3} = 0, \\ r_{u,v} &= r_{\min} + \frac{u - 0.5}{H} \cdot r_{\text{fov}}, \quad \phi_{u,v} = \phi_{\min} + \frac{v - 0.5}{W} \cdot \phi_{\text{fov}}. \end{aligned} \quad (5.8)$$

The grid is fixed in the ego coordinate system and as such independent of any input data and time. The data independence also eliminates alignment errors induced by shadowing. Therefore, moving objects are the only cause of alignment errors in the bird's eye view. Figure 5.7 depicts both strategies.

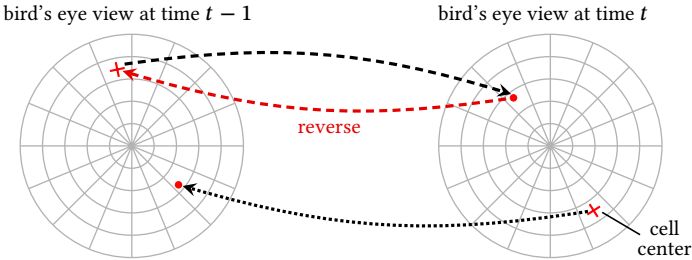


Figure 5.7: Association of bird's eye view cells across time with the backward (dotted) and forward strategy (dashed).

5.3 Temporal Multi View Network

The next important step is the successful combination of the multi view and temporal framework to combine multi view and temporal benefits. However, two challenges arise when attempting to add temporal capabilities to the proposed multi view approach. First, temporal fusion is required for the semantic and instance branch to fully benefit from temporal information. The semantic branch is based on the point backbone’s multi view features, and the instance branch relies on bird’s eye view feature maps. Second, the multi view features of the point backbone are pointwise features and challenging to associate with aggregated temporal information from the previous time step. Spatio-temporal proximity in the point view requires k NN search, which is computationally expensive for large point clouds and even more for combined point clouds across time. The presented temporal framework for 2D approaches in range and bird’s eye view is the foundation to address these challenges. It integrates temporal fusion into the range and bird’s eye view branch.

The combined architecture T-MVNet is depicted in Fig. 5.8. It extends the range view branch with the temporal range view fusion presented in Section 5.1. Since the backbone, as part of the multi view architecture, returns only semantic feature maps, a single temporal pipeline is sufficient. On the other hand, the bird’s eye view branch is enhanced with the temporal bird’s eye view fusion proposed in Section 5.2, which requires the temporal pipeline twice. The architecture of the point backbone is unchanged. However, the last multi view aggregation step no longer uses the final feature maps of the backbones but the aggregated temporal memories of both 2D views. Therefore, it aggregates temporal range and bird’s eye view features to compute temporal multi view features, which improve the 3D semantic segmentation. In parallel, the second temporal bird’s eye view pipeline computes the temporal memory for the instance feature maps provided to the offset and center head. Hence, the offset vectors and center heatmap are also temporally enhanced and, thereby, the instance segmentation. As a result, both subtasks of panoptic segmentation benefit from the proposed temporal multi view approach.

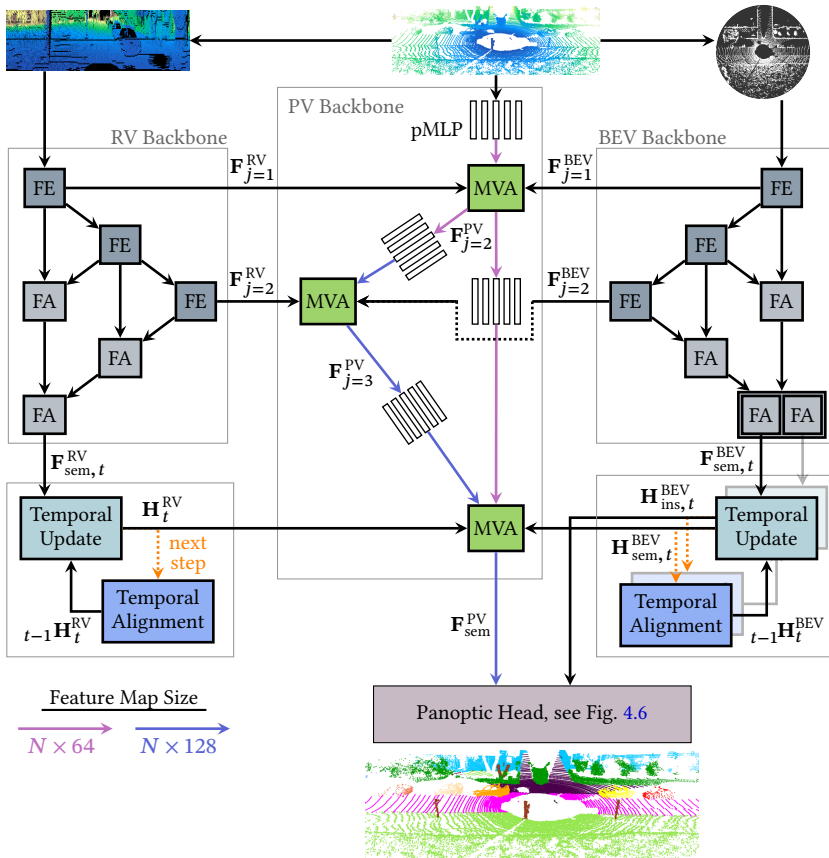


Figure 5.8: Temporal multi view architecture T-MVNet, which combines multi view and temporal benefits. One temporal memory is added for the semantic range view features map F_{sem}^{RV} and two memories for the semantic and instance bird's eye view features maps F_{sem}^{BEV} and F_{ins}^{BEV} , respectively.

6 Multimodal Panoptic Segmentation of 3D Point Clouds

The proposed temporal multi view framework focuses on the lidar sensor as a single sensor modality combined with temporal information. The multimodal multi view framework introduced in this chapter goes one step further and exploits the camera as an additional sensor modality. Since sensor fusion is combined with the multi view architecture, different lidar views are available for fusion. A promising combination is based on camera images and the lidar range view, as motivated in Section 3.3. Consequently, this chapter investigates the fusion of camera and lidar and presents a novel multi scale deep fusion network [Due20b, Due21, Sch22a] to fuse lidar range view feature maps with camera feature maps. Based on the resulting range fusion backbone, the temporal multi view approach is enhanced to a multimodal framework, which combines multi view, temporal *and*, multi sensor benefits.

6.1 Multi Sensor Range View

The sensor fusion range view network SF-RVNet is designed to combine and fuse lidar and camera information in the range view. It addresses two main drawbacks of the mentioned existing approaches in Section 2.5. First, the novel multi scale fusion provides considerably improved multi sensor features to enhance both semantic *and* panoptic segmentation. Second, the architecture and training strategy decouple both sensor backbones to keep them independent for increased robustness against sensor failure. A major drawback of many fusion approaches [Mey19a, Kri20, Zhu21c] is their dependency on

both sensors. Without the proposed design, a fusion approach will considerably degrade when a sensor fails or provides invalid output.

The suggested fusion architecture depicted in Fig. 6.1 relies on lidar and camera backbones, which are connected by the fusion branch. The lidar range view backbone presented in Section 4.1 is used again and computes range view feature maps at multiple stages and scales. The exchangeable camera backbone has the same task and provides multi scale camera feature maps. At least three different scales are required for the proposed and deployed fusion strategies. In general, an arbitrary image network can be used as long as it provides the three different scales of camera feature maps, which is fulfilled by most established architectures. Since ResNets are predominantly used as feature extractors in state-of-the-art image networks, ResNet-50 and ResNet-101 are chosen exemplarily in this approach. The feature maps at the end of each stage are potential candidates for fusion, illustrated in Fig. 6.1, and identified by the respective block based on the official naming convention [He16]. The final block of the fourth stage differs between ResNet-50 and ResNet-101.

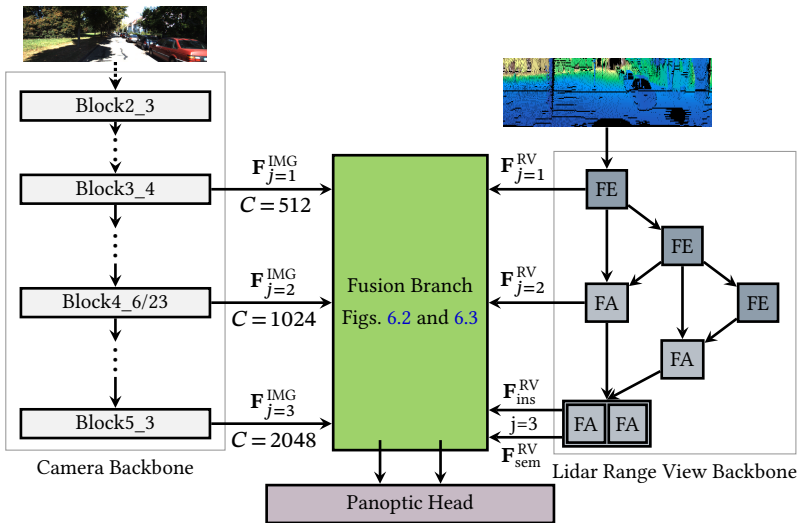


Figure 6.1: Range fusion network SF-RVNet based on lidar range view and camera backbone. A fusion branch connects the backbones by fusing the feature maps of both sensors.

The fusion branch is the vital link between both backbones, which is responsible for the fusion of lidar and camera features. It transforms camera feature maps from camera image to range view, followed by their fusion with different strategies introduced in the following. The fused semantic and instance feature maps are finally provided to the standard panoptic head, introduced in Section 4.1. Furthermore, to train the overall range fusion framework successfully, different training strategies are investigated regarding panoptic results and the impact of sensor failure.

6.1.1 Sensor Fusion

The core element of the fusion network is the fusion branch, which combines lidar range view and camera image feature maps to provide improved features containing the information of both sensors. Following the concept discussed in Section 3.3, deep feature fusion is chosen as the overarching fusion strategy, motivated by the discussed advantages over early and late fusion. In contrast to the latter, deep fusion is a generic strategy because the high number of intermediate feature maps offers countless possibilities for their combination and fusion. Hence, specific deep fusion strategies or architectures have to define several steps. First, feature maps from both sensors and different stages must be selected for fusion. In the second step, a common representation is required for both sensor modalities and a spatial feature transformation into the chosen representation. Finally, a multi scale aggregation strategy for the fused feature maps is required to provide the final multi sensor feature maps. Two novel deep fusion strategies [Due21, Sch22a] are proposed in the following, which address the discussed challenges while proposing different aggregation strategies.

In order to achieve a multi scale fusion, both strategies rely on multiple lidar and camera feature maps, which originate from different scales. The lidar backbone computes feature maps at three different scales, depicted in Fig. 6.1. Therefore, the fusion strategies combine these three scales with camera feature maps at the same number of distinct scales to fully exploit the multi scale potential. The ResNet camera backbone offers up to five different scales. Since

there is no obvious most promising combination, the most beneficial one is evaluated by experiments. In the next step, the camera feature maps are transformed into the lidar range view since the target is a lidar panoptic segmentation. This transformation is based on Eq. (3.9) and the following equation:

$$\begin{aligned} \text{IMG} \mathbf{F}_j^{\text{RV}} &= \hat{\mathcal{S}}(\mathbf{F}_j^{\text{IMG}}, \mathbf{V}_j^{\text{IMG}}), \\ \mathbf{V}_j^{\text{IMG}} &= [V_{u,v,d}] = [\mathcal{J}_j^{\text{IMG}}(\mathbf{g}_{u,v})_d], \quad \mathbf{G} = \mathbf{P}^{\text{RV}}, \\ \mathcal{J}_j^{\text{IMG}}(\mathbf{g}) &= \mathcal{P}_j^{\text{IMG}}(\mathbf{T}^{\text{li} \rightarrow \text{cam}} \cdot \mathbf{g}). \end{aligned} \quad (6.1)$$

The projection depends on the stage j due to the different camera feature map sizes. Initially, projected camera features $\text{IMG} \mathbf{F}_j^{\text{RV}}$ of stage j in the range view have still the range view’s original resolution. After a bilinear down-sampling step to match the size of the lidar feature maps, both sensors can be fused. The last step aggregates the three stages to combine and exploit features at different scales.

The first strategy follows an iterative or hierarchical pattern [Due21] and is illustrated in Fig. 6.2. The main component is a fusion module, which transforms the camera feature maps into the range view following Eq. (6.1) and reduces their large feature channel size to $C_1^{\text{IMG}} = C_2^{\text{IMG}} = 64$, and $C_3^{\text{IMG}} = 128$. This reduction ensures equal channel sizes for lidar and camera feature maps and equal influence of both sensors. Afterwards, they are concatenated and processed by a residual block to provide fused feature maps with $C'_1 = C'_2 = 96$ and $C'_3 = 192$. In the second step, the fused features from the previous stage and scale are combined with the current stage by concatenation and two additional residual blocks. By splitting every module in a sensor fusion and fusion refinement step, the network can focus on a beneficial sensor fusion first, then on combining multi scale multi sensor features. The refinement step is omitted for the first stage since there is no preceding one. By stacking three fusion modules, all three scales of lidar and camera backbone are iteratively exploited and aggregated for the final multi sensor feature maps. Two parallel modules are deployed in the last step to fuse and provide semantic and instance feature maps to the panoptic head.

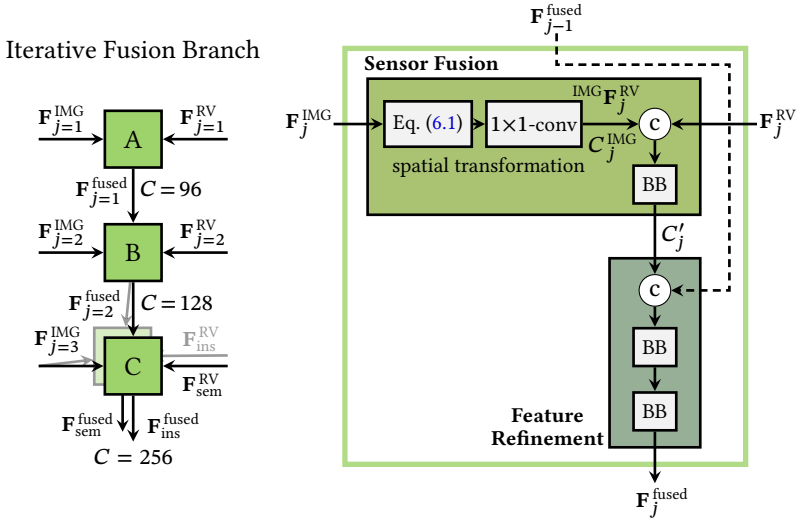


Figure 6.2: Iterative fusion strategy based on alternating sensor fusion and feature refinement steps.

The second strategy [Sch22a] builds upon the idea of FPNs and is depicted in Fig. 6.3. Lidar range view and camera feature maps are combined by the previously presented sensor fusion step. The three resulting multi sensor feature maps are then simultaneously aggregated in a bottom-up and top-down feature pyramid to compute multi scale features. On the one side, the top-down pyramid (Dn) aggregates multi sensor features starting with fine details and incorporates more and more context. On the other side, the bottom-up pyramid (Up) starts with aggregated context and adds more and more details. Both pyramids build upon the feature refinement step depicted in Fig. 6.2. An ablation study investigates the replacement of this module by a simple concatenation and 3×3 -convolution as a more lightweight alternative. The outputs of both pyramids are combined by a convolutional layer, BN, and optionally LReLU (Py). While the iterative strategy doubles only the last module for parallel semantic and instance features, this is impossible for the pyramid fusion. The reason is that the parallel semantic and instance feature maps from the range view are not fused in the last but in the first step of the bottom-up

pyramid. Therefore, two parallel pyramid branches are deployed to provide semantic and instance features.

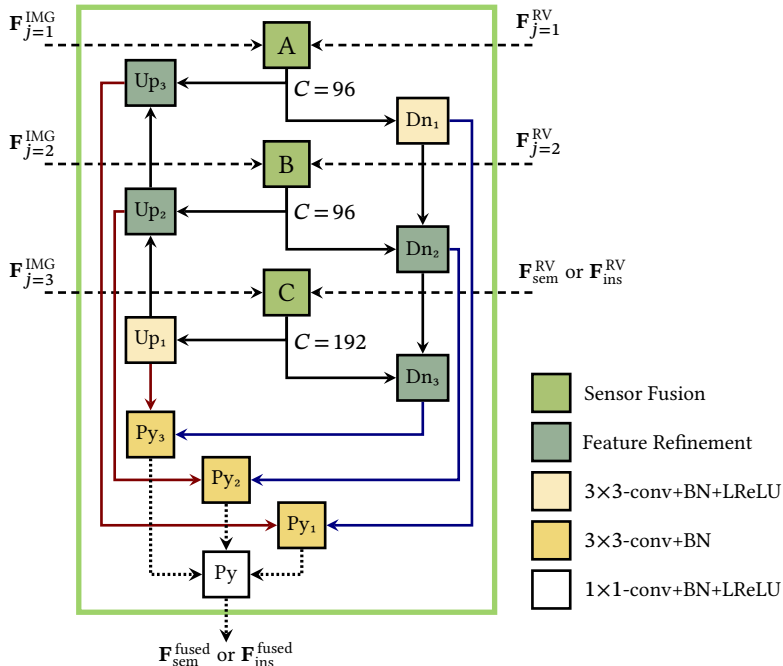


Figure 6.3: Pyramid fusion based on two parallel FPNs. The channel size of the feature maps for all non-labeled edges is $C = 256$.

6.1.2 Training Strategy

The training strategy plays an important role when training multi sensor approaches, especially for the presented range fusion approach. Its decoupled camera and lidar backbone offer the possibility to pre-train them individually on lidar and camera data. Depending on the existing data and sensor setup, this has potentially two advantages. First, no combined data is necessary for this step, which allows using data where only one sensor modality is present,

potentially increasing the amount of training data. Second, the training is not restricted to the overlapping field of view of both sensors. Again, this can significantly increase the amount of training data, e.g., when considering a 360° lidar and front camera.

During the training of the overall architecture, the pre-trained backbones can be further trained or kept unchanged. While fixed backbones might negatively impact predictions, it offers a major advantage in terms of redundancy. In case of sensor failure or unavailability, the backbones can still compute their single sensor predictions as a fallback. This requires as little overhead as applying the single sensor head to the last feature maps of the respective backbone, which ensures a low latency. The required head is needed for and optimized during pre-training in any case. As a result, the following two-step training strategy is deployed:

1. Train both backbones individually on their respective data to fully exploit existing data. The camera backbone can also be trained on other tasks, such as object detection.
2. Train the overall fusion approach with both backbones frozen. This enables the fusion branch to learn a beneficial fusion of lidar and camera without constantly changing backbone features.

6.2 Multimodal Multi View Network

The combination of the previously presented temporal multi view approach of Section 5.3 and the proposed range fusion network is the last step of this thesis and provides the overall multimodal multi view framework. The combined architecture TSF-MVNet is able to simultaneously leverage the potential of a multi view, temporal, and multi sensor architecture.

As a result of the chosen design, the range fusion backbone seamlessly replaces the range view backbone of the temporal multi view architecture, as illustrated in Fig. 6.4. The temporal memory of the range view is now provided with the fused camera and lidar features and aggregates multi sensor features

over time. Consequently, the point view backbone receives multi sensor features at the first level ($j = 1$) and temporally fused multi sensor features at the last level ($j = 3$). The intermediate level ($j = 2$) provides lidar features since no sensor fusion is performed at this stage. In addition, the range fusion can also be combined individually with the multi view and temporal framework, called SF-MVNet and TSF-RVNet, respectively. SF-MVNet replaces the range view backbone of MVNet, and TSF-RVNet the range view backbone of T-RVNet, with the range fusion backbone.

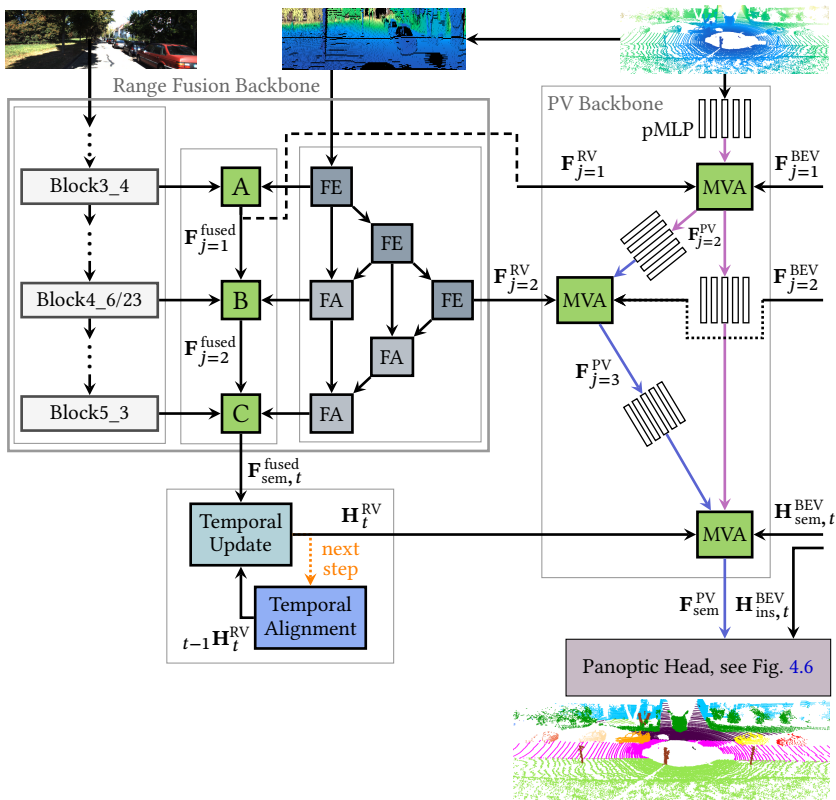


Figure 6.4: The multimodal multi view architecture TSF-MVNet, which combines multi view, temporal, and sensor fusion benefits. The bird’s eye view branch equals the one depicted in Fig. 5.8.

7 Evaluation

The following chapter thoroughly evaluates the individual contributions of this thesis. The first section introduces the datasets and metrics used for the evaluation. Afterwards, the three main contributions of this thesis are evaluated, starting with the multi view framework presented in chapter 4, followed by the temporal framework introduced in chapter 5, and finally, the sensor fusion approach proposed in chapter 6. In addition, another set of experiments investigates the benefits of combining the individual contributions of this work. The respective temporal and multimodal multi view frameworks have been presented in Sections 5.3 and 6.2. All contributions are first evaluated and analyzed by extensive ablation studies, followed by a comparison to state-of-the-art approaches.

7.1 Experimental Setup

The main elements of the experimental setup for evaluation are the selected datasets and metrics. First, the two chosen public, large scale, and challenging datasets from the driving domain are presented and analyzed in the next section. Afterwards, the metrics for semantic and panoptic segmentation are introduced. The reliance on established and frequently used metrics ensures comparability to other state-of-the-art methods.

7.1.1 Datasets

A considerable number of point cloud datasets have been published over the last years, see Section 2.3.2. However, only a few have the necessary properties for the training and evaluation of the proposed multimodal framework, which requires sequential lidar scans and camera images. Additionally, the considered task of panoptic segmentation needs pointwise semantic and instance labels for supervision. These requirements reduce the set of eligible datasets to SemanticKITTI [Beh19] and nuScenes [Cae20], two large scale and distinct outdoor datasets, which are also the predominant choices of other state-of-the-art methods.

SemanticKITTI is a multimodal outdoor dataset recorded in the city of Karlsruhe in Germany and provides pointwise semantic and instance labels based on the KITTI Odometry Benchmark [Gei12]. The 360° lidar scans with up to $64 \cdot 2,083 = 133,312$ points originate from a Velodyne HDL-64E and are recorded with 10 Hz. Two front-facing cameras are triggered by the lidar and provide camera images with a resolution of approximately $1,245 \times 375$ after rectification. The dataset is divided into 22 individual sequences, which are officially grouped into training and test split. Sequences 0–10 with 23,201 frames serve for training and validation, while the remaining sequences 11–21 with 20,351 frames serve for testing. The evaluation on the test set is only possible on the official benchmark server since no labels have been published. To prevent optimizing on the test set, an overall maximum of ten submissions per account is possible. Sequence 8 with 4,071 frames is used for validation throughout the experiments.

The semantic labels of the official benchmark contain 19 distinct classes with eight thing and eleven stuff classes. The classes motorcycle (mcycle), motorcyclist (mclist), other-vehicle (vehicle), and other-ground (ground) are abbreviated throughout the evaluation to increase the readability of tables and plots. An overview of the thing classes is shown in Fig. 7.1. It illustrates the frequency of semantic class labels and the number of total and unique instances for the train and validation set. While most classes are self-explanatory, the difference between bicycle and bicyclist is not obvious. The former refers to a

bicycle without a rider, whereas the latter describes bicycles with a rider and includes both. The same applies to motorcycle and motorcyclist. In general, the thing classes represent only a small proportion of the overall point labels except for the more common car class. The total instance count ranges from 750 to approximately 15,000, which are different recordings of 26–193 unique instances. The exception is again the class car with about 2,200 unique and a total of over 200,000 instances. The distribution of the stuff classes with a combined share of nearly 95% is depicted in Fig. 7.2.

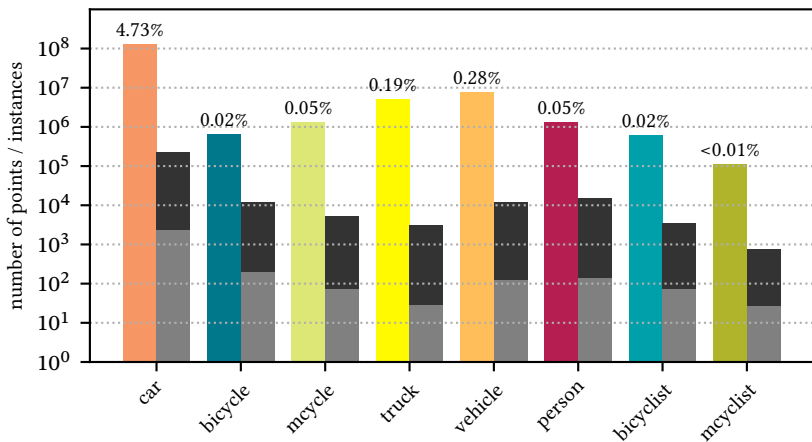


Figure 7.1: Overview of the thing classes of SemanticKITTI with the absolute number and percentage of pointwise semantic labels. Additionally, the number of total and unique instances is shown. The colors correspond to the class label visualization in figures.

SemanticKITTI offers additional tasks alongside the introduced semantic and panoptic benchmarks. The multi scan or dynamic semantic segmentation task further distinguishes between moving and non-moving for the classes car, truck, vehicle, person, bicyclist, and motorcyclist. Consequently, it contains

25 distinct semantic classes and is predominantly used to evaluate temporal approaches. Furthermore, the binary task moving object segmentation requires approaches to classify each point as moving or non-moving.

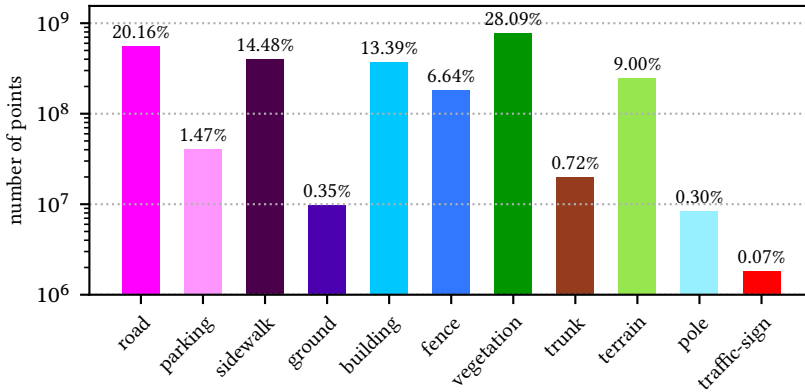


Figure 7.2: Overview of the stuff classes of SemanticKITTI with the absolute number and percentage of pointwise semantic labels.

NuScenes is also a large scale multimodal outdoor dataset with pointwise semantic and instance labels and has been recorded in Boston and Singapore. The 360° lidar scans from a Velodyne HDL-32 were recorded at 20 Hz and contain up to $32 \cdot 1,084 = 34,688$ points per scan. Additionally, six cameras mounted around the car provide camera images of the car’s 360° environment with a resolution of $1,600 \times 900$ each. The dataset is divided into 1,000 individual sequences, each approximately 20 s in length. The official split assigns 700, 150, and 150 sequences or 28,130, 6,019, and 6,008 frames for training, validation, and testing. Similar to SemanticKITTI, the results for the test set can only be evaluated on the official benchmark server, which restricts to three evaluation runs per year. Labels only exist for keyframes sampled across the sensor modalities at 2 Hz, which results in a predominant number of unlabeled intermediate frames. These intermediate frames are required for

temporal training, in contrast to the non-temporal case. Since two consecutive frames are very similar due to the high frame rate of 20 Hz, every other frame is omitted. This speeds up training time without impacting the results.

The semantic segmentation task of nuScenes contains 16 classes, which can be divided into ten thing and six stuff classes. Similar to SemanticKITTI, motorcycle (mcycle) and construction-vehicle (con-vehicle) are abbreviated in the following. Figure 7.4 illustrates the frequency of thing classes for the train and validation set. These classes represent less than 9% of the overall point labels, and most of them individually represent less than 1%. However, trucks and especially cars are more common. The total instance count for the rarer classes lies between approximately 10,000 and 20,000, which corresponds to 600–1000 unique instances. More common classes have more than 70,000 total and 4,000 unique instances up to around 360 000 total and 20,000 unique car instances. The distribution of the stuff classes with a combined share of nearly 92% is depicted in Fig. 7.4.

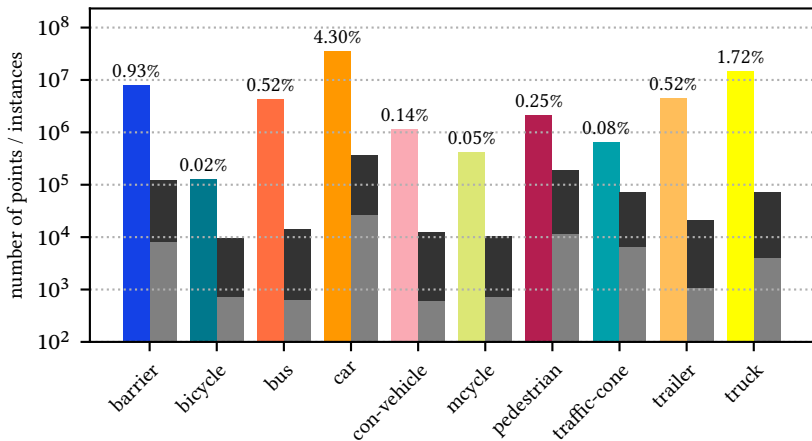


Figure 7.3: Overview of the thing classes of nuScenes with the absolute number and percentage of pointwise semantic labels. Additionally, the number of total and unique instances is shown.

The distinct properties of the datasets motivate the evaluation on both of them. They have been recorded in different countries and use different lidar sensors and cameras. Therefore, the point clouds are much sparser for nuScenes with only one-fourth of the points. SemanticKITTI, on the other hand, has less traffic with only 23 000 moving instances compared to 300 000 of nuScenes. Additionally, the thing classes significantly differ between both datasets, and SemanticKITTI requires a more detailed differentiation of stuff classes.

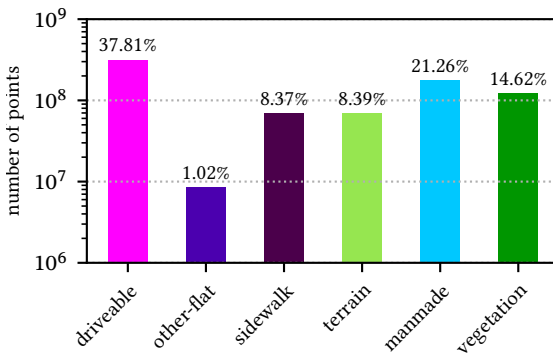


Figure 7.4: Overview of the stuff classes of nuScenes with the absolute number and percentage of pointwise semantic labels.

7.1.2 Metrics

Different metrics have been established to evaluate semantic and panoptic segmentation approaches, which are introduced in the following. Important concepts for classification tasks are true positives, false positives, and false negatives. Applied to semantic segmentation and a semantic class $cls \in \mathcal{C}$, the set of true positives TP_{cls} are all pixels or points which are correctly classified as class cls . The false positives FP_{cls} are classified as class cls but belong to another class, and the false negatives FN_{cls} belong to class cls but have been wrongly classified otherwise, see Fig. 7.5. The visualizations of the concepts in this section rely on image pixels instead of 3D points for a clearer illustration, while the concepts equally apply to 3D points.

The first metric for **semantic segmentation** based on these concepts is the accuracy acc , which is defined by the number of correctly classified points divided by the total number of points N_{total} :

$$acc = \frac{\sum_{cls \in \mathcal{C}} |TP_{cls}|}{N_{\text{total}}}. \quad (7.1)$$

However, accuracy favors dominating classes. For example, an algorithm can achieve an accuracy greater than 99% on SemanticKITTI and still ignore all thing classes except car due to their small proportion. Hence, the prevailing metric to assess semantic segmentation is the mean of the class-wise intersection-over-union IoU_{cls} :

$$mIoU = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} IoU_{cls} = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} \frac{|TP_{cls}|}{|TP_{cls}| + |FP_{cls}| + |FN_{cls}|}. \quad (7.2)$$

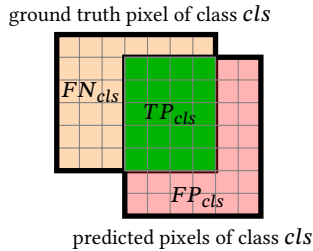


Figure 7.5: Visualization of true positives, false positives, and false negatives on pixel level.

In case of **panoptic segmentation**, the sets of true positives TPI_{cls} , false positives FPI_{cls} , and false negatives FNI_{cls} are defined on instance level. Ground truth and predicted instances I_{gt} and I_{pred} establish a match $(I_{\text{gt}}, I_{\text{pred}}) \in TPI_{cls}$ if their IoU_{match} is greater than 0.5. This threshold ensures that ground truth or predicted instances are only matched once. An unmatched ground truth instance is considered a false negative, and an unmatched predicted instance is a false positive. Predicted instances are required to have a uniform class, which is why the standard evaluation procedure splits not only based on the

predicted instance but also based on semantics. Consequently, points with the same instance label but different semantic classes are considered different instances. Alternatively, a uniform semantic class can be explicitly computed upfront. An exemplary matching is shown in Fig. 7.6.

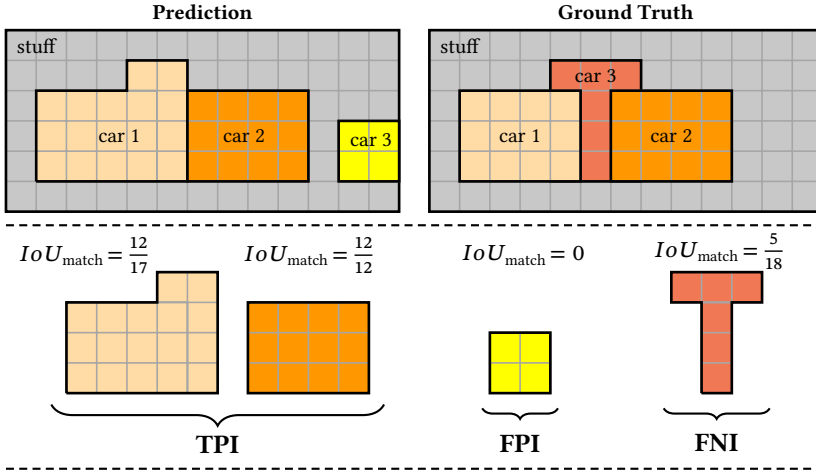


Figure 7.6: The matching of predicted and ground truth instances results in the illustrated sets of true positives (TPI), false positives (FPI), and false negatives (FNI).

The instance detection performance is assessed based on these sets by the mean recognition quality mRQ , defined by

$$mRQ = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} RQ_{cls} = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} \frac{|TPI_{cls}|}{|TPI_{cls}| + 0.5 \cdot (|FPI_{cls}| + |FNI_{cls}|)}, \quad (7.3)$$

and equals the well-known F_1 -score commonly used for object detection. Additionally, the mean segmentation quality mSQ assesses the instance segmentation over TPI_{cls} and measures its quality:

$$mSQ = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} SQ_{cls} = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} \frac{\sum_{\text{match} \in TPI_{cls}} IoU_{\text{match}}}{|TPI_{cls}|}. \quad (7.4)$$

Finally, both metrics are combined into the unified panoptic quality [Kir19]:

$$mPQ = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} PQ_{cls} = \frac{1}{|\mathcal{C}|} \sum_{cls \in \mathcal{C}} SQ_{cls} \cdot RQ_{cls}. \quad (7.5)$$

These metrics also include the stuff classes, which are considered one-instance classes. All ground truth and predicted points of a stuff class belong to *one* ground truth and *one* predicted instance, respectively.

Statistical significance in machine learning is commonly demonstrated with k -fold cross-validation. However, applying this method to approaches based on deep learning is often challenging and rarely seen in literature. The underlying reason is that training a single deep learning approach on one of the folds may take days, and training on all folds can take weeks or even a month. The average training time of the experiments in this thesis was about 2.6 days, which results in 26 days for a 10-fold cross-validation for a single experiment. Since this is infeasible, the statistical significance is shown by a randomization test [Smu07], which tests the likelihood that one approach is truly better than another instead of achieving better results by coincidence. A randomization test checks for paired probes of both compared approaches if they originate from the same underlying distribution, without making any assumptions about this distribution.

The test computes a metric m_q for two approaches and Q predictions or sets of predictions, which results in Q paired probes. The test's null hypothesis assumes that both approaches are equally good and, therefore, originate from the same underlying distribution \mathcal{M}_q : $m_{q,0} \sim \mathcal{M}_q$ and $m_{q,1} \sim \mathcal{M}_q$. Under this assumption, the assignment of $m_{q,0}$ and $m_{q,1}$ to their respective approach is irrelevant for the paired probes, and switching would not influence the measured results. The test creates a large amount of these permutations and computes the test metric, such as the mean \bar{m} , for both approaches to verify or reject the null hypothesis. Consequently, the p -value is the fraction of permutations with an equal or higher difference $|\bar{m}_0 - \bar{m}_1|$ than the originally measured difference. The null hypothesis is rejected if the p -value is smaller than a selected significance level α . The following evaluation aims for a confidence of about two standard deviations and selects $\alpha = 0.05$.

The natural way to apply a randomization test to the setup of this thesis is to consider the results for each frame as paired probes. However, $mIoU$ and mPQ are not computed for each frame individually. After aggregating true positives, false positives, and false negatives over all frames, they are computed over the entire validation set. As a result, the individual frames have a different influence depending on the occurring classes and the frequency of their points or instances. This inequality prevents the application of the test for $mIoU$ and mPQ on frame-level. One possibility to address this is considering each 3D point and its prediction as a probe. However, this requires comparing all point predictions and computing the metrics over the entire validation set with hundreds of millions of points from scratch for each permutation. Consequently, it would take days just to compare two approaches when a meaningful number of 100 000 or more permutations are used. In order to make this test computationally feasible, a subset of five million points is randomly drawn for every permutation. The original and permuted difference of the two approaches is computed on these subsets and both allow the computation of the p -value. While these computations are still too expensive for the mPQ , they allow showing the statistical significance for the $mIoU$. An underlined $mIoU$ in the following tables indicates a significant difference to the previous underlined value, or the first line. The row showing a significant difference to all other lines is underlined twice.

7.1.3 General Implementation Details

All experiments are implemented based on PyTorch¹ and use distributed parallel training in mixed precision mode on up to eight NVIDIA V100 GPUs. Adam optimizer [Kin15] is used across all experiments with a weight decay of 0.0005 and optimizes the networks for up to 100 000 iterations. The initial learning rate λ_0 is exponentially reduced during training following:

$$\lambda_i = \lambda_0 \cdot e^{-5 \cdot 10^{-5} \cdot i}. \quad (7.6)$$

¹ <https://pytorch.org/>

All non-temporal experiments are trained with a batch size of 32. Due to the high memory demands of TBPTT, and to ensure a constant batch size across all temporal experiments, the batch size is reduced to 16 for temporal trainings. All non-temporal lidar experiments are pre-trained on the semantic task with an initial learning rate of $\lambda_0 = 0.001$, followed by the panoptic training with a smaller initial learning rate of $\lambda_0 = 0.0001$. The same initial learning rate is also used for temporal and sensor fusion experiments, which use pre-trained single frame or single sensor backbones.

Data augmentation is important to reduce overfitting and improve the results. All trainings randomly flip the point clouds along the x - and y -axis with a probability of 0.5 for every dimension. Additionally, the point clouds are rotated for a random angle around the z -axis. Finally, random 180° crops of the 360° scans are used for training. Purely range view-based experiments deviate slightly from this crop and use 2D crops of size 64×1024 . Finally, up to 10 instances are pasted randomly into the scene. The augmentations are applied temporally consistent for temporal trainings to retain the temporal dependencies and spatial consistency across time. The range images' size is 64×2083 for SemanticKITTI and 32×1084 for nuScenes. Furthermore, a bird's eye view grid of size 480×360 is employed in both cases, with $r_{\min} = 2$ m, $r_{\max} = 50$ m, and covering the entire 360° . Points outside are mapped to the closest cell.

To generate the heatmap and offset vector targets for the instance clustering, the instance centers are determined by the mean over all instance points. An unnormalized Gaussian kernel with $\sigma = (2, 8)$ for range view and $\sigma = (4, 4)$ for bird's eye view is added at this position to the ground truth heatmap. NMS suppresses a 5×5 -area in bird's eye view and a 3×7 -area in range view when the best 50 candidates are extracted from the predicted heatmap. The default loss weights are $\lambda_{\text{sem}} = 1$, $\lambda_{\text{ctr}} = 100$, and $\lambda_{\text{off}} = 0.1$.

Runtime measurements are performed in single precision and are reported as mean and with standard deviation (std) over the validation set. One related challenge is the usage of different GPUs across state-of-the-art approaches. If an official repository exists, the inference time is remeasured on a V100, indicated by (*). Otherwise, and on condition that the used GPU is specified, the

values are estimated (\approx) for the V100 based on inference benchmark results published online¹ and shown in Table 7.1.

Table 7.1: Relative inference performance of different GPUs.

GPU	Relative Performance
RTX 2080 Ti	0.89
V100	1.00
RTX 3090	1.28

7.2 Multi View Panoptic Segmentation

The evaluation of this thesis starts with the multi view panoptic segmentation approach as the first contribution. It is thoroughly evaluated by an extensive number of experiments, starting with the evaluation of the range and bird’s eye view network and the benefits of the proposed improvements. Follow-up experiments assess the multi view architecture and fusion. If not stated otherwise, the default multi view architecture presented in Fig. 4.6 is used with concatenation as fusion strategy and trained with $\lambda_{\text{sem}}^{\text{RV}} = \lambda_{\text{sem}}^{\text{BEV}} = \lambda_{\text{sem}}^{\text{PV}} = 1$.

7.2.1 Range View Experiments

The network RVNet presented in Section 4.1 is a single view approach, and its backbone is one of the core elements of the multi view framework. It is based on DLA from the image domain and related to LaserNet, which successfully applies this architecture to lidar data. However, the initial results of this architecture for the task of 3D panoptic segmentation are mediocre, as shown in the first line of Table 7.2, despite the usage of an established architecture from the image domain. For that reason, Section 4.1 proposed different improvements for the network, which are investigated in the following. Additionally, an improved data augmentation strategy has been presented in Section 4.3.

¹ <https://mtli.github.io/gpubench/>

The proposed approach considers the sensor properties by deploying the improved range view projection and asymmetric stride, motivated in Section 4.1. This sensor-aware architecture (SAA) reduces the amount of lost information during the projection based on the improved range view projection. Its asymmetric stride considers the asymmetric distance between range view cells. Next, and building upon the findings of [Aks20], Lovász loss (LV) is added to the loss and applied equally weighted with CE loss. It is robust to imbalanced class distributions and an improved surrogate for optimizing the $mIoU$. Finally, random objects are pasted into the range images (ROA) to increase the occurrence frequency of rare thing classes. All three proposed and discussed enhancements significantly improve the panoptic results and achieve a large overall improvement for both metrics, as illustrated in Table 7.2. Additionally, LV and ROA only influence the training procedure but not the inference time. Solely the asymmetric stride increases the latter.

Table 7.2: Improvements achieved by sensor-aware architecture (SAA), Lovász loss (LV), and random object augmentation (ROA) on the validation set of SemanticKITTI. Bold metrics indicate the best values of each column, and underlining indicates significance, see Section 7.1.2.

SAA	LV	ROA	mPQ	$mIoU$	$t_{inf} \pm \text{std}$ in ms
			0.401	0.487	35.3 ± 1.6
✓			0.427	<u>0.496</u>	46.4 ± 1.4
✓	✓		0.486	<u>0.535</u>	46.4 ± 1.4
✓	✓	✓	0.519	<u>0.597</u>	47.0 ± 1.4

Another experiment evaluates the choice of the general backbone architecture compared to other established architectures. Therefore, two representatives of common 2D architectural families are evaluated with the same set of improvements. DeepLabV3 has been chosen as a representative for ResNet-based networks with a pyramid pooling module, which has already been successfully applied for panoptic segmentation in the image domain [Che20]. The corresponding experiment uses the official architecture of Panoptic-DeepLab with ResNet-101 and additionally applies the proposed improvements. However, the results and runtime depicted in Table 7.3 are significantly worse than the

results and runtime of the DLA backbone. The second architecture, U-Net, has been chosen since it is frequently used for lidar segmentation tasks and was also one milestone in the image segmentation domain. While it achieves better results than the DeepLab backbone and is slightly faster than the DLA backbone, it lacks segmentation quality. Consequently, the chosen architecture is the strongest and most promising backbone choice for the single view and also the multi view approach.

Table 7.3: The achieved panoptic results and number of trainable parameters of different backbone architectures on the validation set of SemanticKITTI.

Backbone	mPQ	$mIoU$	Parameters	$t_{\text{inf}} \pm \text{std}$ in ms
DeepLab	0.456	0.556	74.3 M	135.0 ± 1.6
U-Net	0.490	<u>0.575</u>	43.2 M	44.1 ± 1.4
DLA	0.519	<u>0.597</u>	5.2 M	47.0 ± 1.4

7.2.2 Bird’s Eye View Experiments

The bird’s eye view network BEVNet presented in Section 4.2 is the second single view approach, and its backbone is the second core element of the multi view approach. While it follows the same backbone architecture as the range view network, its initial results are considerably better, see Table 7.4. Nevertheless, combining CE and Lovász loss significantly improves the results, similar to the range view network. The same holds for adding random object augmentation to insert rare instances into the point clouds. The combination of both enhancements achieves a considerable improvement of the semantic and panoptic segmentation, measured by $mIoU$ and mPQ , respectively.

The bird’s eye view network is further enhanced with the proposed sparse semantic head (SH), which restricts the convolution operations in the semantic head to occupied cells. This restriction avoids the expensive expansion of the 2D bird’s eye view feature maps to 3D voxel predictions used by existing approaches [Zha20c, Zho21]. Table 7.4 underlines the benefits regarding inference time. In addition, the training time is also reduced by approximately

30%. Simultaneously, the prediction quality remains unaffected, which makes the sparse head a valuable addition.

Table 7.4: Panoptic improvements achieved by Lovász loss (LV) and random object augmentation (ROA) on the validation set of SemanticKITTI. The sparse semantic head (SH) improves the inference time.

SH	LV	ROA	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
✓			0.491	0.559	36.4 ± 1.4
✓	✓		0.523	<u>0.570</u>	35.8 ± 1.3
✓	✓	✓	0.552	<u>0.611</u>	36.2 ± 1.4
	✓	✓	0.554	0.613	40.7 ± 1.4

The chosen DLA backbone architecture is also compared to the already motivated representatives DeepLab and U-Net for the bird’s eye view. While DeepLab achieves better results in the bird’s eye view than in the range view, it still achieves the lowest values for the considered metrics and has the highest inference time. The choice of U-Net is additionally motivated by PolarNet [Zha20c], which successfully uses a U-Net backbone in its bird’s eye view network. While it achieves the highest $mIoU$, it is outperformed for panoptic segmentation by the DLA backbone, which also has a considerably lower inference time. The general differences in the runtime compared to the range view originate from a higher resolution and symmetric strides. The former impacts the inference time negatively, the latter positively. Depending on the architecture, this leads to faster or slower inference times. Overall, the chosen DLA architecture is the most promising choice for panoptic segmentation.

Table 7.5: The achieved panoptic results and number of trainable parameters of different backbone architectures on the validation set of SemanticKITTI.

Backbone	mPQ	$mIoU$	Parameters	$t_{\text{inf}} \pm \text{std}$ in ms
DeepLab	0.495	0.572	74.8 M	70.4 ± 1.7
U-Net	0.544	0.617	43.3 M	43.9 ± 1.0
DLA	0.552	0.611	5.3 M	36.2 ± 1.4

7.2.3 Multi View Experiments

The multi view approach proposed in this thesis combines the range, bird’s eye, and point view to exploit the strengths of the individual views and compensate for weaknesses. An initial set of multi view experiments analyzes the effects of different view combinations, and the corresponding results are depicted in Table 7.6. Both previously evaluated single view approaches are the initial baselines and provide the first important insights. The results of BEVNet are superior to RVNet, especially when considering the panoptic metrics. This superiority supports the decision of this thesis to choose the superior bird’s eye view for instance clustering. In the next step, two out of three views are combined to investigate the influence of the individual views and to reveal the combination with the most potential. All these experiments follow the exact same architecture proposed in Section 4.3 to ensure a fair comparison, except that one of the three views is removed. The first experiment RVNet+PV combines range and point view and shows that this setup achieves no improvements. Instance clustering is still performed in range view, without any influence of the point view, which is challenging because of instances occluding each other. As a result, instance centers are close together and the clustering is prone to errors in the offset predictions. One minor advantage is the direct prediction of a 3D semantic segmentation without the necessity of a back-projection. RVNet+PV achieves the same panoptic results as RVNet with the kNN-based back-projection strategy. The combination of bird’s eye and point view BEVNet+PV also does not improve the panoptic segmentation, which is the consequence of the point view only affecting the predicted 3D semantic segmentation. Since the sparse head of the bird’s eye view already predicts high resolution 3D voxel semantics, the benefits of pointwise features are negligible. The following experiments combine range and bird’s eye view, initially with simpler fusion strategies and not yet with the proposed point view network. It is the first setup that predicts the center heatmap and offset vectors in the bird’s eye view, while the 3D semantic segmentation benefits from range and bird’s eye view features. The first and simplest fusion of both views is based on the final predictions, which are fused by computing the geometric mean. Even this simple fusion strategy significantly improves

the panoptic and semantic segmentation and confirms the value of combining range and bird’s eye view due to their distinct underlying projections. The panoptic results are further improved if the geometric fusion is replaced by a learned late fusion of the final pointwise range and bird’s eye view features.

Table 7.6: Impact of different view combinations on panoptic and semantic results on the validation set of SemanticKITTI and nuScenes.

	Approach	RV	BEV	PV	mPQ	$mIoU$	$t_{inf} \pm \text{std}$ in ms
SemanticKITTI	RVNet	✓			0.519	0.597	47.0 ± 1.4
	BEVNet		✓		0.552	<u>0.611</u>	36.2 ± 1.4
	RVNet+PV	✓		✓	0.519	<u>0.597</u>	57.8 ± 1.0
	BEVNet+PV		✓	✓	0.547	<u>0.613</u>	48.2 ± 1.5
	Geom. Fusion	✓	✓		0.568	<u>0.632</u>	56.9 ± 5.2
	Late Fusion	✓	✓		0.574	0.638	58.7 ± 4.7
	MVNet	✓	✓	✓	0.592	0.657	76.0 ± 1.7
	MVNetAll	✓	✓	✓	0.591	0.657	85.8 ± 1.5
	Approach	RV	BEV	PV	mPQ	$mIoU$	$t_{inf} \pm \text{std}$ in ms
nuScenes	RVNet	✓			0.614	0.704	32.1 ± 2.3
	BEVNet		✓		0.603	<u>0.680</u>	35.7 ± 1.6
	MVNet	✓	✓	✓	0.651	<u>0.756</u>	57.6 ± 1.5

While these results are already promising, this simple multi view architecture still lacks in leveraging the full multi view potential. Therefore, the approach presented in Section 4.3 deploys a third backbone for the point view, which repeatedly fuses the features at multiple scales from range and bird’s eye view to refine and enhance the pointwise features. As a result, and shown in Table 7.6, the proposed MVNet improves the panoptic segmentation even further and significantly outperforms every combination of two views. Since MVNet aggregates multi view features after selected feature extractor and aggregator blocks, see Fig. 4.6, an additional experiment, MVNetAll, evaluates the potential of fusing after every block. However, the higher number of aggregation steps provides no enhancements. MVNet achieves similar improvements on

nuScenes and outperforms both single view approaches by a large margin, depicted in the lower part of Table 7.6. One notable difference is the, in relation, worse performing BEVNet, which results from nuScenes’ sparser point clouds. With only a quarter of SemanticKITTI’s point cloud size, the bird’s eye view gets increasingly sparse, which negatively impacts its results.

To investigate the multi view benefits more closely, the class-wise outcomes for thing and stuff classes are presented in Table 7.7 and Table 7.8, respectively. The overall improvements of the $mIoU$ are directly reflected in the individual class IoU with better results for six out of eight thing classes. The results of the other two classes are similar to BEVNet or RVNet. Enhancements of the semantic segmentation for stuff classes are even more pronounced, and all classes but two benefit.

Table 7.7: Comparison of the class-wise results for thing classes on the validation set of SemanticKITTI. Due to space limitations, leading zeros are omitted in this and the following class tables.




Approach	Metric	car	bicycle	mcycle	truck	vehicle	person	bicyclist	mcyclist	mean things
RVNet	IoU	.900	.514	.669	.683	.414	.651	.818	.008	.582
BEVNet		.947	.483	.720	.702	.524	.629	.794	.042	.605
MVNet		.955	.541	.777	.929	.516	.715	.819	.077	.666
RVNet	PQ	.816	.492	.512	.297	.484	.728	.842	.013	.523
BEVNet		.910	.516	.649	.494	.491	.725	.833	.059	.585
MVNet		.914	.550	.664	.629	.504	.764	.860	.101	.623

When comparing the PQ of thing classes, MVNet predominantly outperforms RVNet due to the superior instance predictions in the bird’s eye view. Compared to BEVNet, MVNet again enhances all classes but by a smaller margin. This is expected since the instance clustering is provided by the bird’s eye view part of MVNet and equals BEVNet. In this case, the main improvements originate from the semantic segmentation, which is confirmed by the fact that

almost all thing or stuff classes with improved PQ also have improved IoU . Especially for stuff classes as one-instance classes, improving the semantic segmentation is the only possibility to improve the PQ . Consequently, the stuff classes show similar enhancements for the PQ as for the IoU . Overall, the predominantly improved class metrics emphasize the value of combined range and bird’s eye view features and the proposed approach’s capability to successfully combine and exploit them.

Table 7.8: Comparison of the class-wise results for stuff classes on the validation set of SemanticKITTI.

Approach	Metric	road	parking	sidewalk	ground	building	fence	vegetation	trunk	terrain	pole	traffic sign	mean stuff
RVNet	IoU	.938	.362	.804	.020	.851	.493	.848	.597	.706	.592	.470	.607
BEVNet		.938	.449	.793	.003	.891	.499	.857	.595	.709	.584	.448	.615
MVNet		.950	.471	.824	.011	.906	.597	.859	.670	.697	.651	.523	.651
RVNet	PQ	.940	.181	.767	.000	.802	.186	.830	.402	.518	.525	.530	.516
BEVNet		.935	.252	.758	.000	.870	.180	.836	.417	.529	.514	.522	.528
MVNet		.949	.297	.792	.005	.882	.247	.840	.507	.539	.608	.605	.570

These findings are illustrated by the selected semantic and instance examples shown in Fig. 7.7. In the semantic example, BEVNet fails to accurately segment the static environment and misses a pole  entirely (a). In addition, it classifies parts of the yard as terrain , which is considered sidewalk  in the ground truth (b). On the other hand, MVNet correctly segments the pole and yard area as a result of the multi view architecture. The second example shows instance segmentation results and two major errors of RVNet. It is unable to correctly separate the highlighted parked cars (c) and (d), whereas MVNet provides the correct instance segmentation. This example underlines the advantage of relying on instance segmentation based on the bird’s eye view.

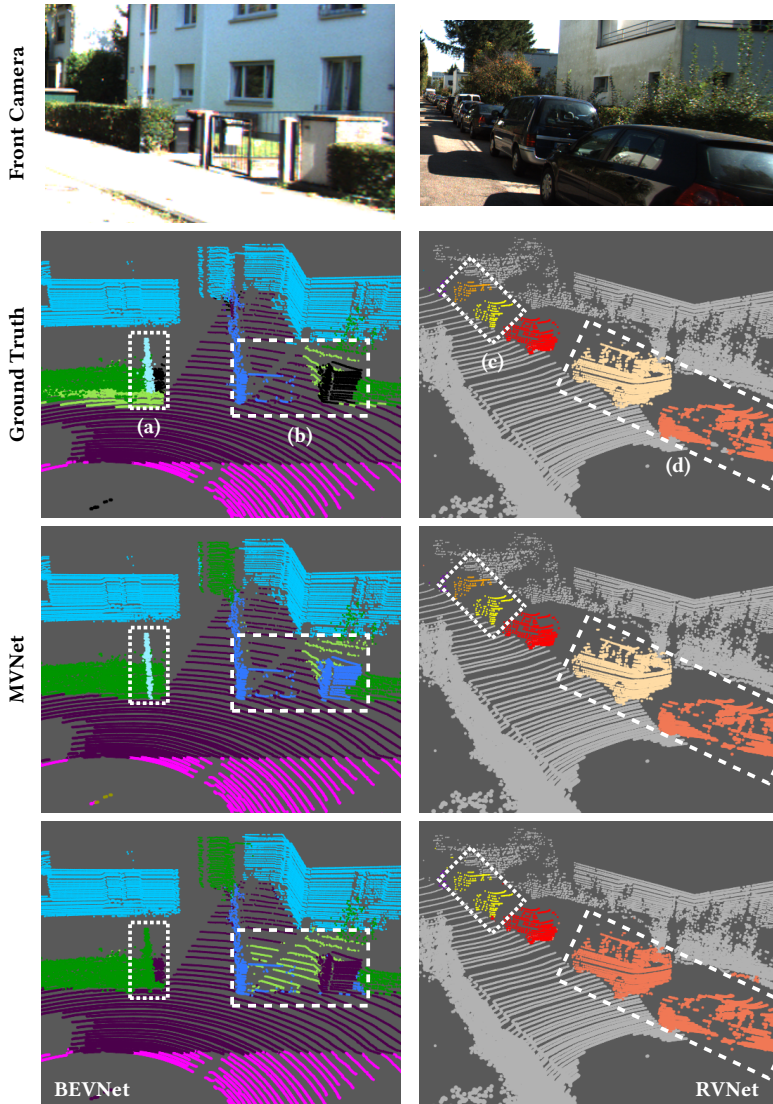


Figure 7.7: Two selected examples showing the superiority of MVNet over the single view approaches BEVNet and RVNet. The left example shows semantic and the right example instance segmentation. Black points in the ground truth indicate unlabeled points.

Extended Ablation Studies

After the general multi view setup has been evaluated, the pointwise fusion of range, bird’s eye, and point view features is investigated more closely. The different strategies proposed in Section 4.3 are addition, element-wise maximum, concatenation (concat), and learned weighted sum (lws). The corresponding results are depicted in Table 7.9 and show that all fusion strategies achieve a similar panoptic quality. Concatenation and addition achieve slightly better semantic segmentation than the other strategies and are therefore the favorable choices.

Table 7.9: Panoptic results and inference times of different multi view fusion strategies on the validation set of SemanticKITTI.

Fusion Strategy	<i>mPQ</i>	<i>mIoU</i>	$t_{\text{inf}} \pm \text{std}$ in ms
concat	0.592	0.657	76.0 ± 1.7
max	0.592	<u>0.647</u>	79.4 ± 1.8
lws	0.593	0.651	81.2 ± 1.6
add	0.595	0.654	74.0 ± 1.7

The proposed loss for the multi view framework comprises several weighted components related to the semantic or instance task, see Eq. (4.16). Crucial components are the auxiliary semantic losses for range and bird’s eye view, which improve the semantic results significantly, illustrated in Table 7.10. This first set of experiments evaluates the influence of different semantic weights by training the multi view approach solely for semantic segmentation with different weight combinations. The best semantic segmentation is achieved with equally weighted views, whereas preferring individual views negatively impacts the results. The semantic loss is completed by the center and offset losses for the instance task, and all three are of different magnitudes. Therefore, the primary goal of the respective weights $\lambda_{\text{sem}} = \lambda_{\text{sem}}^{\text{RV}} + \lambda_{\text{sem}}^{\text{BEV}} + \lambda_{\text{sem}}^{\text{PV}}$, λ_{off} , and λ_{ctr} is to equalize their magnitudes. Table 7.10 presents the results of additional experiments, which evaluate permutations of weights used in existing works [Yan19c, Che20, Zho21]. However, no significant best combination is observable across the evaluated permutations.

Table 7.10: Results of different weight combinations for the semantic and instance loss on the validation set of SemanticKITTI.

$\lambda_{\text{sem}}^{\text{RV}}$	$\lambda_{\text{sem}}^{\text{BEV}}$	$\lambda_{\text{sem}}^{\text{PV}}$	$mIoU$	λ_{sem}	λ_{off}	λ_{ctr}	mPQ	$mIoU$
0	0	1	0.626	1.0	0.1	100.0	0.588	0.652
1/4	1/4	1/2	<u>0.648</u>	3.0	0.1	100.0	0.592	0.657
1/3	1/3	1/3	<u>0.660</u>	1.0	0.1	200.0	0.590	0.652
3/8	3/8	2/8	<u>0.654</u>	3.0	0.1	200.0	0.587	0.655

Overall, the presented results motivate the choice of the architecture of Fig. 4.6 with three MVA modules and concatenation-based fusion as the final multi view approach. Furthermore, the best loss weights are $\lambda_{\text{sem}} = 3$, with equally weighted views, $\lambda_{\text{off}} = 0.1$, and $\lambda_{\text{ctr}} = 100$. This setup is used for all subsequent multi view experiments and its comparison to state-of-the-art. Since the comparison equals the one of the final multimodal multi view approach, it is jointly presented later in this chapter. In this case, the panoptic head unifies the semantic class of each instance to the dominating class, improving the mPQ to 0.604.

7.3 Temporal Panoptic Segmentation

As the second contribution, the temporal panoptic segmentation approaches are extensively evaluated in the next step, starting with the evaluation of the temporal range and bird’s eye view networks. The conducted experiments analyze the benefits of the proposed recurrent temporal architecture in both views, and several ablation studies investigate the influence of the proposed components. Finally, the presented methods are compared to other state-of-the-art temporal approaches.

If not stated otherwise, the memory update is based on the residual strategy with four BBs. The default temporal training strategy updates the weights every $\kappa_1 = 5$ steps, propagates the gradient $\kappa_2 = 4$ steps back in time, and performs the first update after $\kappa_3 = 10$ warm up steps. The sequence length

defaults to $\zeta = 50$, and the pre-trained backbone of RVNet or BEVNet is used as initialization instead of training from scratch.

7.3.1 Temporal Range View Experiments

The temporal range view network T-RVNet introduced in Section 5.1 builds upon a recurrent architecture to recursively aggregate and propagate features and information through time. T-RVNet builds upon several individual components and a temporal training strategy to exploit temporal information. The first experiments investigate the contribution of the individual components to the improved panoptic results, with an overview depicted in Table 7.11. One of these components is the backbone of RVNet as single frame backbone (SFB). Consequently, RVNet is the appropriate single frame baseline shown in the first row of Table 7.11, allowing the evaluation of temporal improvements.

Table 7.11: Influence of the individual components of the temporal architecture. Starting with the single frame backbone (SFB) as a baseline and consecutively adding the temporal memory (TM), alignment (TMA) and Truncated Backpropagation Through Time (TBPTT) [Wil90].

	SFB	TM	TBPTT	TMA	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
SemanticKITTI	✓				0.519	0.597	47.0 ± 1.4
	✓	✓			0.559	<u>0.615</u>	66.6 ± 1.4
	✓	✓	✓		0.556	0.615	66.5 ± 1.5
	✓	✓		✓	0.573	<u>0.640</u>	66.7 ± 1.3
	✓	✓	✓	✓	0.575	<u>0.658</u>	66.9 ± 1.3
nuScenes	SFB	TM	TBPTT	TMA	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
	✓				0.614	0.704	32.1 ± 2.3
	✓	✓	✓	✓	0.669	<u>0.744</u>	41.3 ± 2.1

The first step towards the proposed temporal architecture is the addition of the temporal memory (TM), which significantly improves the panoptic results. Although no temporal alignment and training has been applied yet, the temporal network is already able to benefit from temporal information.

Adding TBPTT only improves the panoptic segmentation in conjunction with the temporal alignment. The alignment is crucial to ensure the correct association of past and current features by compensating ego motion in the range view. Consequently, aligning the memory based on the backward strategy further improves the results and additionally allows the architecture to benefit from TBPTT. With the alignment, the errors are correctly propagated through time, and the semantic segmentation is further enhanced. Overall, T-RVNet achieves major improvements over the single frame network RVNet with a significantly higher mPQ and $mIoU$. Equally convincing improvements on nuScenes are depicted in Table 7.11. The increased runtime is a consequence of the temporal memory added to the semantic and instance branch of RVNet.


To further investigate the temporal benefits, the individual class results are considered and compared with the non-temporal approach RVNet. The semantic and panoptic segmentation is improved for every thing class, as illustrated in Table 7.12. The strong benefits for these seem counterintuitive since the temporal approach disregards their movement. However, a considerable number of the respective instances are stationary, such as parked cars or bicycles, or move slowly like pedestrians. In these cases, no alignment errors exist or are negligible. In the case of higher motion velocity, the resulting error depends on the relative movement direction and is also small in many scenarios. Therefore, thing classes strongly benefit from temporal information in most situations. In addition, stuff classes similarly benefit, as illustrated in Table 7.13. The highest absolute improvements of all classes are revealed for the classes truck and vehicle. Both classes are often confused due to the vehicle class containing object types with a partially similar shape to trucks, such as buses and trailers. Since individual instances are only partially observed, this leads to shape ambiguities and confusion. Temporal information reduces this confusion based on the aggregated information, which is reflected in a considerable reduction of instances with classifications alternating between truck and vehicle over time. Overall, T-RVNet achieves predominantly enhanced results for class-wise IoU and PQ , as well as significant improvements of the means over thing and stuff classes.

Table 7.12: Comparison of the class-wise panoptic results for thing classes on the validation set of SemanticKITTI.

Approach	Metric	car	bicycle	mcycle	truck	vehicle	person	bicyclist	mcyclist	mean things
RVNet	<i>IoU</i>	.900	.514	.669	.683	.414	.651	.818	.008	.582
T-RVNet		.952	.538	.699	.906	.758	.700	.833	.017	.675
RVNet	<i>PQ</i>	.816	.492	.512	.297	.484	.728	.842	.013	.523
T-RVNet		.839	.540	.536	.641	.596	.796	.865	.070	.610

Table 7.13: Comparison of the class-wise panoptic results for stuff classes on the validation set of SemanticKITTI.

Approach	Metric	road	parking	sidewalk	ground	building	fence	vegetation	trunk	terrain	pole	traffic sign	mean stuff
RVNet	<i>IoU</i>	.938	.362	.804	.020	.851	.493	.848	.597	.706	.592	.470	.607
T-RVNet		.949	.531	.821	.038	.877	.606	.847	.624	.698	.612	.498	.645
RVNet	<i>PQ</i>	.940	.181	.767	.000	.802	.186	.830	.402	.518	.525	.530	.516
T-RVNet		.951	.314	.782	.000	.833	.222	.823	.459	.519	.562	.571	.549

These findings are illustrated by the selected example shown in Fig. 7.8. Despite the correct segmentation of the parking area  in the first frame, RVNet fails to accurately segment the parking spot in the following frames. On the other hand, T-RVNet correctly predicts this area across all frames due to the exploitation of temporal information and provides improved and temporarily consistent predictions.

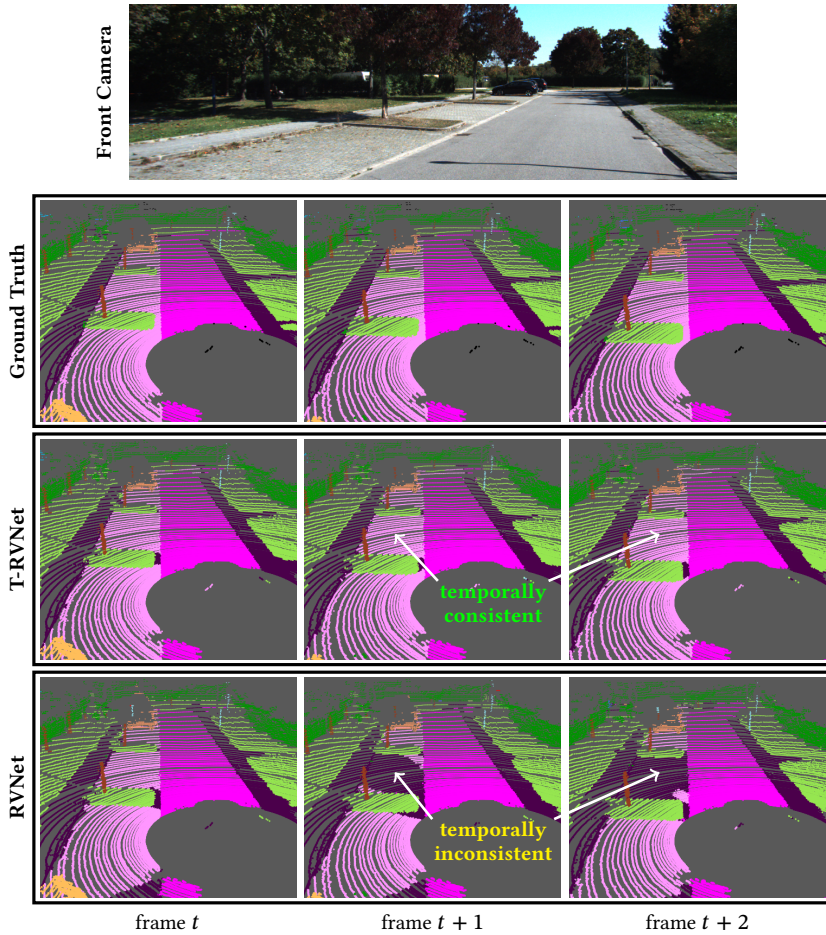


Figure 7.8: Example for temporally robust and improved results of T-RVNet.

Extended Ablation Studies

In the next step, the discussed components are evaluated individually. The core element of the temporal architecture is the temporal memory, which recursively fuses temporally aggregated feature maps from the past with the

latest feature maps. Table 7.14 illustrates the results of the different fusion strategies proposed in Section 5.1 and provides several key insights. Gating mechanisms commonly used for RNNs provide no advantages over a residual-based fusion. This finding confirms the previous claim that a residual strategy does not suffer from exploding or vanishing gradients. The underlying reasons are that the gradient is backpropagated only for a few steps and that residual networks were designed as very deep networks, see Section 5.1 for a detailed discussion. On the other hand, the native ConvGRU suffers from the discussed limited or missing spatial aggregation after the feature fusion. The spatial aggregation is especially beneficial in the considered setup since it adds the capability to compensate for small errors in the alignment step. Therefore, the native ConvGRU achieves the worst mPQ and $mIoU$, barely improving over RVNet. The importance of sophisticated context aggregation is additionally confirmed by the results of the proposed ContextGRUs, which integrate a residual network into their candidate branch. The provided context aggregation by two or four BBs significantly improves the panoptic segmentation, while two blocks are already sufficient. However, and despite the established gating mechanism, directly applying the residual networks as residual strategy without gating achieves a better panoptic segmentation. Hence, the best strategy and preferred choice is a residual update based on four BBs.

Table 7.14: Results of different fusion strategies for the memory update on the validation set of SemanticKITTI.

Category	Strategy	mPQ	$mIoU$	$t_{inf} \pm \text{std}$ in ms
residual	$2 \times \text{BB}$	0.568	0.646	58.5 ± 1.4
	$4 \times \text{BB}$	0.575	<u>0.658</u>	66.9 ± 1.3
	BB + BoB + BB	0.567	<u>0.644</u>	61.6 ± 1.3
gated	ConvGRU	0.533	<u>0.605</u>	52.9 ± 1.5
	ContextGRU $2 \times \text{BB}$	0.564	<u>0.642</u>	67.0 ± 1.3
	ContextGRU $4 \times \text{BB}$	0.566	0.644	75.4 ± 1.5

The second investigated component is the temporal memory alignment. Two distinct alignment strategies and a range gating mechanism have been proposed to compensate ego motion and disable wrong feature alignments caused by moving objects or occlusion. The overview in Table 7.15 reveals that both strategies perform similarly well, with a minor advantage for the backward strategy. Consequently, the backward strategy is preferable since it is faster and provides better values. Independent of the strategy, the temporal method requires no explicit alignment error detection provided by the proposed range threshold because the panoptic results remain unchanged. These errors either have no negative impact on the memory update, or their frequency is too low to influence the considered metrics. On the other hand, if the range gate is too restrictive, e.g., 2 cm, which is the standard deviation of the sensor’s measurement error, a considerable amount of valuable information is ignored. Consequently, the results are negatively influenced.

Table 7.15: Results of the two alignment strategies and different range gate thresholds on the validation set of SemanticKITTI.

Strategy	δ_{range}	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std in ms}$
forward	0.02m	0.548	0.635	70.5 ± 1.5
	0.1m	0.570	<u>0.655</u>	70.4 ± 1.5
	none	0.568	0.653	69.6 ± 1.4
backward	0.02m	0.555	<u>0.638</u>	67.6 ± 1.4
	0.1m	0.568	<u>0.655</u>	67.8 ± 1.4
	none	0.575	0.658	66.9 ± 1.3

Another set of experiments investigates the influence of the temporal training parameters sequence length and temporal backpropagation steps. These parameters influence the training procedure proposed in Section 5.1.3 instead of the model architecture. The results in Table 7.16 show that the proposed approach is insensitive to changes in these parameters. Especially the mPQ is very similar across all parameter combinations, whereas the differences in the $mIoU$ are more prominent. Overall, no clear tendency is visible that a

particular sequence length or number of backpropagation steps achieves superior results. Hence, no sequence lengths longer than 100 have been evaluated. The maximum number of backpropagation steps was limited by memory requirements, which increases nearly linearly with this number. Another key insight provided by Table 7.16 is the importance of training with artificial subsequences. Training with the nine native sequences of SemanticKITTI fails and cannot leverage any temporal potential for the reasons discussed in Section 5.1.3. It achieves no improvements over the non-temporal baseline approach RVNet.

Table 7.16: Influence of the temporal training parameters sequence length ζ and backpropagation steps κ_2 evaluated on the validation set of SemanticKITTI.

	$\zeta = 25$		$\zeta = 50$		$\zeta = 100$		$\zeta = \text{native}$	
	<i>mPQ</i>	<i>mIoU</i>	<i>mPQ</i>	<i>mIoU</i>	<i>mPQ</i>	<i>mIoU</i>	<i>mPQ</i>	<i>mIoU</i>
$\kappa_2 = 1$	0.573	0.652	0.574	0.649	0.569	0.640		
$\kappa_2 = 4$	0.570	0.643	0.575	0.658	0.576	0.656	0.515	0.597
$\kappa_2 = 8$	-	-	0.576	0.656	0.573	0.647		

The recursive feature aggregation over time is the core idea of the proposed temporal approach with the important and novel benefit of an unrestricted temporal window without influence on the runtime. An unrestricted temporal memory raises the question of how many past frames actually contribute to the improvements of the latest frame. To answer this question, experiments with restricted temporal memory are conducted. While trained with the default temporal setup, the number of past frames aggregated in the temporal memory is restricted during evaluation. The corresponding results are plotted in Fig. 7.9. The first insight is the expected importance of past information, confirmed by the considerable degradation of the *mPQ* if the temporal memory is disabled during evaluation ($\zeta_{\text{mem}} = 0$). In this case, even the single frame baseline achieves a better panoptic segmentation. When the number of considered past frames increases, the panoptic results considerably improve up to ten frames. Afterwards, the improvements slow down, and with $\zeta_{\text{mem}} = 50$, the results of the unbounded memory are nearly achieved. These

findings underline that a double-digit number of past frames is required to fully exploit the potential of temporal information and emphasize the value of the runtime independence.

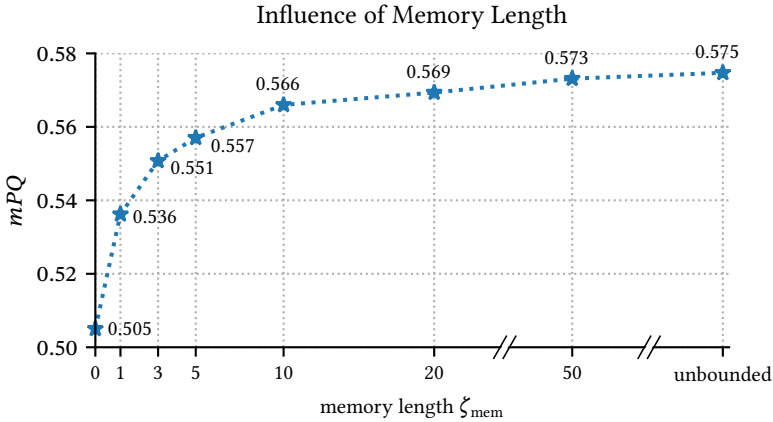


Figure 7.9: Influence of the number of considered past frames, or memory length, on the results.

Furthermore, an important factor for temporal fusion is the ego velocity and time step size. The velocity influences the change of the static environment between frames and the step size the distance traveled by moving objects. Hence, another experiment investigates the influence of increasing the velocity and time step size on the results, which are depicted in Fig. 7.10. For this experiment, the evaluation skips frames to simulate higher velocities and larger time steps. Both factors are coupled for existing datasets and cannot be simulated independently. The first data point for both datasets are the default result at the given mean ego velocity over the evaluation set. The second entry corresponds to aggregating only every other frame into the temporal memory. This setup simulates a velocity or time step size twice as high. Therefore, the evaluation starts with the first frame and skips every other frame, which is repeated starting with the second frame to provide predictions for every frame. The third entry corresponds to skipping two frames, tripling velocity or time

step size, and so on. Naturally, the improvements achieved by temporal fusion decrease with increasing velocity since the jointly observed area in both time steps decreases. Consequently, the area increases where temporal fusion provides no benefits. Additionally, the errors induced by disregarding object movement increase with larger time step sizes. Nevertheless, the proposed temporal approach achieves considerable improvements up to very high velocities occurring, e.g., on the highway. The enhancements are higher for SemanticKITTI because its lidar provides point clouds with a higher density, compensating for larger distances between frames to some extent. Furthermore, nuScenes suffers more from the higher time step size due to a higher number of moving objects.

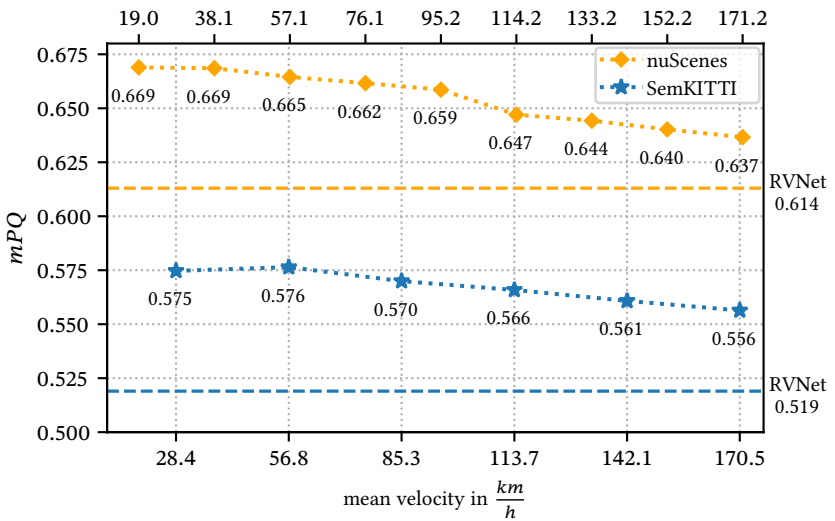


Figure 7.10: Influence of the ego velocity and time between frames on the results. The time step size is 0.1 s for the first entry and increases by 0.1 s with each entry to the right.

7.3.2 Temporal Bird’s Eye View Experiments

The temporal bird’s eye view network T-BEVNet builds upon the same concepts as T-RVNet. Therefore, a reduced subset of the previous experiments is conducted to investigate and confirm the temporal benefits in the bird’s eye view. The influence of the individual components on the results is shown in Table 7.17. Since T-BEVNet builds upon the backbone of BEVNet as a single frame backbone, BEVNet is the respective non-temporal baseline shown in the first row of Table 7.17. The first added component is again the temporal memory, which considerably improves the panoptic results. The improvements are slightly lower compared to the temporal range view, which can be explained by the superior panoptic segmentation of BEVNet compared to RVNet. Again, temporal alignment and training are not required in order to benefit from temporal information. However, this reduced setup lacks in exploiting its full potential. The following lines confirm that the temporal memory alignment is crucial to further improve the panoptic results and to leverage the full potential of TBPTT. Similar to T-RVNet, major overall improvements over the single frame network BEVNet are achieved with significantly higher *mPQ* and *mIoU*. The enhanced results on nuScenes depicted in Table 7.17 confirm the convincing outcomes provided by the temporal approach.

Table 7.17: Benefits provided by the individual components. Starting with the single frame backbone (SFB) as baseline and consecutively adding the temporal memory (TM), temporal alignment (TMA) and Truncated Backpropagation Through Time (TBPTT).

	SFB	TM	TBPTT	TMA	<i>mPQ</i>	<i>mIoU</i>	$t_{\text{inf}} \pm \text{std}$ in ms
SemanticKITTI	✓				0.552	0.611	36.2 ± 1.4
	✓	✓			0.573	<u>0.624</u>	49.1 ± 1.3
	✓	✓	✓		0.585	0.621	49.1 ± 1.2
	✓	✓		✓	0.593	<u>0.648</u>	53.4 ± 1.2
	✓	✓	✓	✓	0.609	<u><u>0.670</u></u>	53.5 ± 1.2
nuScenes	SFB				<i>mPQ</i>	<i>mIoU</i>	$t_{\text{inf}} \pm \text{std}$ in ms
	✓				0.603	0.680	35.7 ± 1.6
	✓	✓	✓	✓	0.686	<u><u>0.732</u></u>	45.8 ± 3.1

The evaluation of the memory update in the bird’s eye view focuses on the most promising strategies revealed in the previous chapter. Table 7.18 shows the results for the two best residual strategies and ContextGRU as the best gated strategy. Again, the gating mechanism provides no benefits and is outperformed by both residual strategies. While the residual network built from two BBs achieves a similar mPQ with less runtime, four BBs achieve a significantly higher $mIoU$.

Table 7.18: Results of different fusion strategies for the memory update on the validation set of SemanticKITTI.

Category	Strategy	mPQ	$mIoU$	$t_{inf} \pm \text{std}$ in ms
residual	2 × BB	0.610	0.656	47.5 ± 1.3
	4 × BB	0.609	0.670	53.5 ± 1.2
gated	ContextGRU 2 × BB	0.596	<u>0.653</u>	59.2 ± 1.2

The investigation of the temporal training parameters sequence length ζ and backpropagation steps through time κ_2 depicted in Table 7.19 show similar results as for T-RVNet. However, the configuration with a sequence length of 50 and 4 backpropagation steps is the best strategy for temporal bird’s eye view training when considering the semantic results.

Table 7.19: Influence of the temporal training parameters sequence length ζ and backpropagation steps κ_2 evaluated on the validation set of SemanticKITTI.

	$\zeta = 25$		$\zeta = 50$		$\zeta = 100$	
	mPQ	$mIoU$	mPQ	$mIoU$	mPQ	$mIoU$
$\kappa_2 = 1$	0.594	0.658	0.606	0.658	0.602	0.660
$\kappa_2 = 4$	0.605	0.651	0.609	0.670	0.603	0.648
$\kappa_2 = 8$	-	-	0.611	0.654	0.609	0.641

Overall, the best temporal range and bird’s eye view results are achieved with the backward strategy for temporal alignment without range gate and a temporal memory update based on four residual BBs. The best training strategy

uses subsequences with a length of 50 and propagates the gradient four steps back in time.

7.3.3 Comparison to State-of-the-Art

A comparison of the proposed temporal framework to other temporal methods follows the detailed ablation studies of the previous sections. Therefore, it is compared to the approaches presented in Section 2.4 across various segmentation tasks. As introduced in Section 7.1.3, the inference time of existing methods that used different GPUs is reproduced (*) if the code was released or converted (\approx) based on public benchmarks otherwise.

Table 7.20: Panoptic and semantic segmentation results of temporal approaches on the test set of SemanticKITTI. Additionally, the achieved temporal improvements over the respective non-temporal baseline on the validation set are reported. The inference times for T-RVNet and T-BEVNet are for the panoptic and semantic setup, respectively.

Approach	Test Set		Validation Set		t_{inf} in ms
	mPQ	$mIoU$	ΔmPQ	$\Delta mIoU$	
Wang et al. [Wan22b]	0.546	0.606	0.009	-0.003	$\approx 70 / -$
SpSequenceNet [Shi20]	-	0.571	-	0.027	- / 647*
MetaRange [Wan22a]	-	0.610	-	0.023	- / 44*
<i>T-RVNet (Ours)</i>	0.546	0.614	0.056	0.061	66.9 / 36.3
<i>T-BEVNet (Ours)</i>	0.554	0.629	0.055	0.057	53.5 / 28.9

In the first step, the results and relative improvements for semantic and panoptic segmentation are considered, which only a few existing approaches explicitly address. The overall results are reported on the test set. On the other hand, the improvements are reported on the validation set because the ablation studies, which reveal the temporal benefits, are performed thereon. The overview in Table 7.20 shows the superiority of the proposed temporal framework, achieving with both representations the best results and the lowest inference time. The approach of Wang et al. [Wan22b] uses a sophisticated baseline which provides good results. However, their aggregation of input points

for thing classes as temporal fusion strategy achieves only minor improvements. SpSequenceNet [Shi20] only considers one previous frame, which considerably limits the leveraged temporal potential. Consequently, it is clearly outperformed by the proposed method in overall results and achieved improvements. Finally, MetaRange [Wan22a] relies on input level fusion based on residual images. Their semantic results show that this strategy cannot compete with the proposed feature-based strategy.

Table 7.21: Results of temporal approaches for the task of dynamic semantic segmentation on the test set of SemanticKITTI.

Approach	$mIoU$	t_{inf} in ms
SpSequenceNet [Shi20]	0.431	647*
TemporalLattice [Sch22b]	0.471	≈ 199
MetaRange [Wan22a]	0.495	44*
TVSN [Han22]	0.525	–
Wang et al. [Wan22b]	0.529	≈ 70
<i>T-RVNet (Ours)</i>	0.530	36.3
<i>T-BEVNet (Ours)</i>	0.530	28.9

Next, the results for the dynamic semantic segmentation task are compared and depicted in Table 7.21. The presented temporal framework achieves once more the best results with the lowest runtime. Some already discussed approaches also tackle this task but again cannot compete with the proposed approach for the reasons mentioned. The exception is Wang et al. providing similar results, which indicates that their aggregation of instance points at the input level provides valuable information for identifying moving points. The recurrent approach of TemporalLattice [Sch22b] achieves considerably worse results while having a higher computational complexity than T-RVNet and T-BEVNet. These findings show that a careful design of RNN-based architectures is required to exploit its potential with low computational complexity.

Finally, the temporal framework of this thesis is compared to other approaches for the task of moving object segmentation, and the results are illustrated in Table 7.22. These approaches predominantly rely on residual images and

are designed specifically for this task. They exploit the temporal information solely to identify moving points and not to improve learned features in general. Nevertheless, the proposed temporal approach achieves the second-best results while being the fastest.

Table 7.22: Moving object segmentation results on the test set of SemanticKITTI.

Approach	$mIoU$	t_{inf} in ms
SpSequenceNet [Shi20]	0.399	647*
LMNet [Che21b]	0.625	35*
4DMOS [Mer22]	0.652	146*
Sun et al. [Sun22]	0.702	168*
RVMOS [Kim22]	0.747	≈ 37
<i>T-RVNet (Ours)</i>	0.710	36.3
<i>T-BEVNet (Ours)</i>	0.712	28.9

To summarize, the temporal framework shows excellent performance and improvements across four different tasks and outperforms all other temporal approaches on three of them. These convincing results underline the benefits of the proposed temporal feature fusion and confirm its capabilities of exploiting these. None of the existing approaches is capable of achieving convincing results across all these tasks simultaneously. Furthermore, the presented approach achieves the best inference time, which shows the value of the recursive aggregation and reuse of previously computed feature maps. Overall, the presented temporal framework achieves excellent results with low computational complexity.

7.4 Multi Sensor Panoptic Segmentation

The next step of the evaluation investigates the proposed sensor fusion of lidar and camera for panoptic segmentation, which is the third contribution. Various experiments thoroughly evaluate the presented multi sensor fusion

architecture and training strategy to analyze the benefits of the proposed approach. Finally, it is compared to other state-of-the-art sensor fusion methods.



In contrast to previous sections, the following experiments are conducted on nuScenes, if not stated otherwise. Unlike SemanticKITTI, it provides camera images for the entire 360° environment and allows using the entire lidar scan for fusion instead of a small overlapping field of view in the front. The ResNet-50 of BEVDepth [Li22d] pre-trained on the nuScenes object detection task is used as the camera backbone for these experiments. On the other hand, a PSPNet with default architecture is used for the experiments on SemanticKITTI, which is pre-trained on the semantic segmentation provided by the KITTI-STEP dataset [Web21]. Furthermore, the pre-trained backbone of RVNet is used as initialization for the lidar backbone. If not stated otherwise, the camera feature maps of the second, fourth, and fifth stage are fused by the iterative architecture, and both sensor backbones are frozen during the fusion training. The inference time reported for sensor fusion approaches excludes the runtime of the camera backbone since it is not the focus of this thesis.

7.4.1 Range View Fusion Experiments

The sensor fusion network SF-RVNet introduced in Section 6.1 relies on a multi scale feature fusion for lidar and camera feature maps to improve panoptic segmentation based on multi sensor features. The first multi sensor experiments investigate the impact of the individual iterative fusion components on the results, which are depicted in Table 7.23. The baseline is RVNet since its backbone is one of the components of the fusion approach. Instead of adding the entire fusion branch at once, its modules A, B, and C are added individually to investigate the influence of each stage. The first experiment solely uses the last fusion module, which already improves the results by a considerable margin. This finding underlines the effectiveness of the proposed fusion module and the value of camera features in general. The *mPQ* is further enhanced when the second and first fusion modules are added. This leads to the conclusion that a multi scale fusion is beneficial, as well as the iterative refinement of the fused features.

Table 7.23: Influence of the individual fusion stages on the results.

	Baseline	+C	+BC	+ABC	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
nuScenes	✓				0.614	0.704	32.1 ± 2.3
	✓	✓			0.680	<u>0.739</u>	41.9 ± 0.7
	✓	✓	✓		0.705	0.741	44.5 ± 0.8
	✓	✓	✓	✓	0.707	0.743	47.0 ± 0.8
SemKITTI	Baseline	+C	+BC	+ABC	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
	✓				0.559	0.617	-
	✓	✓	✓	✓	0.577	<u><u>0.642</u></u>	-

These findings are illustrated by selected semantic and instance examples depicted in Fig. 7.11. In the former, RVNet fails to classify the bus  correctly and confuses it with a truck , which has a similar shape. In contrast, SF-RVNet exploits camera information to resolve this confusion and correctly detects the bus. The second example shows instance results and the advantage of the high camera resolution. SF-RVNet provides a more accurate instance segmentation on the border of two trailers. This example also shows errors in the ground truth, one of the challenges mentioned in the introduction.

Extended Ablation Studies

Additional experiments investigate the revealed benefits of SF-RVNet over the baseline RVNet more closely. One possible and unwanted source of improvement is the increased model capacity due to the added fusion branch. Therefore, SF-RVNet is trained with empty camera features to exclude this possibility. Consequently, it uses the increased model capacity but cannot exploit camera features. The results in Table 7.24 show only a small improvement for mPQ and no improvements for $mIoU$, which eliminates the increased model capacity as the main source of enhancement. The second experiment examines the dependence of the fusion approach on camera features during inference. Hence, SF-RVNet is trained with camera features but receives no camera features during inference. The huge quality drop depicted in Table 7.24 is another strong evidence that the proposed fusion approach intensively exploits

camera features. This simulated camera failure can be mitigated by applying the lidar head deployed during pretraining to the lidar backbone feature maps to achieve the panoptic results of RVNet instead of the degraded ones.

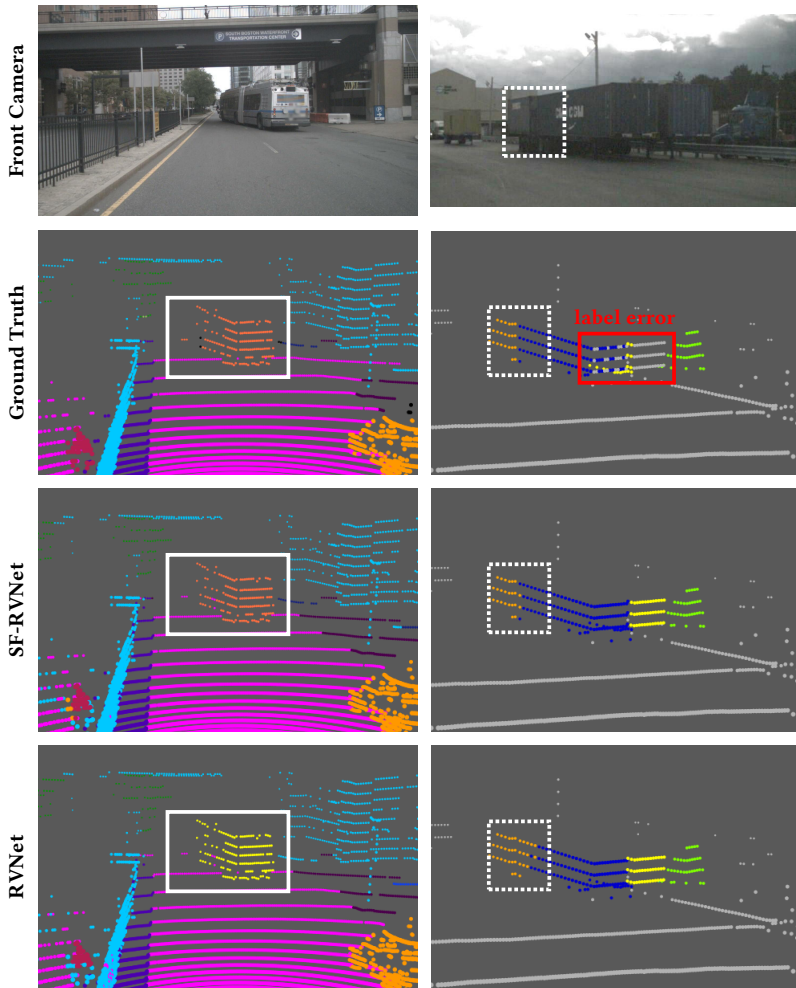


Figure 7.11: Improved semantic and instance segmentation of SF-RVNet due to the fusion of lidar and camera information.

Table 7.24: Relevance of camera information for SF-RVNet during training and inference, evaluated on the validation set of nuScenes.

Baseline	<i>mPQ</i>	<i>mIoU</i>
RVNet	0.614	0.704
SF-RVNet, trained without camera	0.639	0.701
SF-RVNet, inference without camera	0.600	<u>0.683</u>
SF-RVNet	0.707	<u>0.743</u>

The next step investigates the impact of the selected camera feature maps and fusion strategy. The ResNet architecture provides feature maps at five different stages. However, the first stage is not further considered since it consists solely of an initial 3×3 -conv and max pooling, which provides only shallow features. All combinations of the other four stages are evaluated, and all perform equally well, with the results shown in Table 7.25.

Table 7.25: Influence of the chosen ResNet stages for extracting intermediate camera feature maps, evaluated on the validation set of nuScenes.

ResNet Stages	<i>mPQ</i>	<i>mIoU</i>	$t_{\text{inf}} \pm \text{std in ms}$
2, 3, 4	0.707	0.742	48.6 ± 0.8
2, 3, 5	0.706	0.738	47.9 ± 0.8
2, 4, 5	0.707	0.743	47.0 ± 0.8
3, 4, 5	0.703	0.742	47.9 ± 0.8

In contrast, the chosen fusion architecture significantly impacts the results. Table 7.26 depicts the outcomes of the strategies proposed in Section 6.1.1. The pyramid strategy achieves the best results at the cost of a high computational complexity. The latter is mainly caused by the requirement to deploy two entire pyramid modules in parallel, doubling the fusion branch’s computational complexity. Additionally, the pyramid fusion itself is more complex than the iterative fusion. While iterative fusion achieves worse results, it still provides excellent improvements and has a significantly lower inference time.

Table 7.26: Results of the different fusion architectures on the validation set of nuScenes.

Fusion Architecture	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
iterative	0.707	0.743	47.0 ± 0.8
pyramid light	0.712	<u>0.757</u>	64.3 ± 3.3
pyramid	0.719	0.754	91.2 ± 3.7
pyramid extended	0.729	<u>0.768</u>	243.9 ± 3.2

The pyramid fusion is evaluated in three different configurations, starting with the default and light pyramid strategies, which achieve similar panoptic results. The light variant replaces the feature refinement modules with a single convolutional layer, see Section 6.1.1. In contrast, the extended pyramid fusion doubles the projected camera and fusion channel sizes in the second and third sensor fusion step to maximize the exploited camera information. The larger channel sizes further increase the mPQ and $mIoU$, as well as the computational complexity.

One advantage of the proposed architecture is the potential independence of lidar and camera backbone. Both backbones are pre-trained on their respective data and frozen during fusion training to achieve independence. This procedure raises the general question of how the training strategy influences the panoptic results. Table 7.27 shows the results of different strategies for freezing or further optimizing the individual backbones during fusion training. If the lidar backbone is further optimized when training the overall fusion approach, SF-RVNet performs worse considering the $mIoU$. It is likely harder for the fusion branch to learn a high quality feature fusion of constantly changing backbone features. Combined with the major advantage of the backbones acting as a fallback for sensor failure, discussed in Section 6.1, the preferable training strategy keeps both backbones unchanged.

Overall, both sensor fusion strategies provide excellent results, with the iterative strategy being faster and the pyramid strategy providing the best results. The stages the camera features are chosen from play no significant role. Freezing the backbones during fusion training provides the best results and ensures

that the fusion approach does not degrade below the lidar baseline, see Table 7.24, in case of camera failure. Therefore, the final range fusion network uses the extended pyramid fusion strategy with camera stages two, four, and five. However, the iterative strategy is chosen for the deployment in the multimodal multi view architecture. As one of many components, computational complexity is an important property in this setup.

Table 7.27: Impact of optimizing the backbones during training of the fusion approach. The experiments were conducted on the validation set of nuScenes.

Optimize		<i>mPQ</i>	<i>mIoU</i>
Lidar Backbone	Camera Backbone		
		0.707	0.743
	✓	0.707	0.742
✓		0.714	<u>0.718</u>
✓	✓	0.709	<u>0.693</u>

7.4.2 Comparison to State-of-the-Art

In addition to the presented extensive ablation studies, the proposed sensor fusion approach is further compared to other fusion approaches. As discussed in Section 2.5, only a few approaches for semantic segmentation have been proposed, and to the best of the author’s knowledge, no approach tackles the related task of panoptic segmentation.

Since SemanticKITTI only has front-facing cameras, the evaluation for sensor fusion must be restricted to the overlapping field of view (FoV) of the lidar and camera, which covers approximately 1/6 of the overall point cloud. Consequently, the following results are reported on the validation set because the official test server supports no restriction to the overlapping FoV. The outcomes for different methods are depicted in Table 7.28. The first finding is the superiority of SF-RVNet over the single modality range view approaches. This confirms the capabilities of the proposed approach to improve semantic segmentation based on sensor fusion. It is worth mentioning that further approaches with better results than the listed methods exist but do not provide their code,

preventing the evaluation on the restricted dataset. Another finding is that simple fusion schemes, such as PointPainting [Vor20] and RGBAL [Elm19], do not achieve state-of-the-art results. PointPainting has been proposed for 3D object detection and enhances the input point cloud with semantic labels from camera semantic segmentation. However, it achieves no convincing results for 3D semantic segmentation. RGBAL projects the camera image into range view at the input level, which omits a great amount of information due to the lower range view resolution. In contrast to these methods, and similar to the proposed approach, LaserNet++ [Mey19a] and PMF [Zhu21c] deploy a deep fusion of camera and lidar features. LaserNet++ performs a single scale fusion without reaching state-of-the-art semantic segmentation, whereas PMF provides competitive results using multi scale fusion. However, their architecture integrates camera features into their lidar backbone. In case of camera failure, their lidar backbone cannot be used as a fallback and the overall network will severely degrade.

Table 7.28: Semantic results of fusion approaches on the validation set of SemanticKITTI restricted to the overlapping FoV of lidar and camera. († values taken from [Zhu21c], ‡ reproduced results based on own implementation.)

Approach	Modalities	$mIoU$
RangeNet [Mil19]	lidar	0.525
SqueezeSegV3 [Xu20]	lidar	0.556
SalsaNext [Cor20]	lidar	0.614
PointPainting [Vor20]	lidar + camera	0.545 [†]
RGBAL [Elm19]	lidar + camera	0.562 [†]
LaserNet++ [Mey19a]	lidar + camera	0.562 [‡]
PMF [Zhu21c]	lidar + camera	0.639
<i>SF-RVNet (Ours)</i>	lidar + camera	0.642

The previous findings are confirmed by the results on nuScenes depicted in Table 7.29, which allows using the entire scan and official validation set. The validation instead of the test set is chosen because PMF and LIFSeg [Zha21b] report their results only on the former. With LIFSeg, an additional approach is evaluated on nuScenes and achieves the best results. However, this is mainly

caused by their strong voxel-based backbone CylinderNet [Zhu21b]. The fusion strategies of PMF and SF-RVNet achieve larger improvements. When the presented sensor fusion approach is combined with a stronger backbone, such as the proposed multi view approach, it considerably outperforms LIFSeg.

Table 7.29: Semantic results of fusion approaches on the validation set of nuScenes with the achieved improvements over their respective lidar baseline.

Approach	Modalities	$mIoU$	$\Delta mIoU$
RangeNet [Mil19]	lidar	0.655	-
SalsaNext [Cor20]	lidar	0.722	-
PMF [Zhu21c]	lidar + camera	0.769	0.047
LIFSeg [Zha21b]	lidar + camera	0.782	0.039
<i>SF-RVNet (Ours)</i>	lidar + camera	0.768	0.064
<i>SF-MVNet (Ours)</i>	lidar + camera	0.799	0.043

Overall, the proposed sensor fusion architecture achieves state-of-the-art results on both datasets while having the ability to counteract camera failure with its lidar backbone. Additionally, the proposed fusion method can be combined with stronger networks, such as MVNet, which is investigated more closely in the next section.

7.5 Multimodal Multi View Panoptic Segmentation

The main contributions of this thesis have been evaluated individually so far. Another important goal of this thesis is the combination of these contributions into a multimodal multi view approach. Therefore, the next set of experiments investigates the benefits of combining the multi view, temporal, and multi sensor frameworks. In the second step, a comprehensive comparison of the combined frameworks, as the final result of this thesis, to other state-of-the-art approaches is conducted.

7.5.1 Multimodal Experiments

The three frameworks presented in this thesis are combined step by step, starting with the multi view and temporal framework. Table 7.30 shows the results on SemanticKITTI. Adding the temporal memory to the range view branch of the multi view architecture, as presented in Section 5.3, significantly improves the mPQ and $mIoU$ while the latter improvement is more pronounced. This is expected since the temporal range view memory only influences the semantic segmentation and not the center and offset predictions for the clustering. On the other hand, the temporal bird’s eye view memory improves the mPQ by a considerably larger margin since it directly influences the clustering predictions. The best results are achieved when both views are temporally enhanced, which significantly outperforms the non-temporal multi view approach. The large improvement of the panoptic segmentation confirms the value of combining the presented multi view and temporal framework. Another experiment, T-MVNet-Lite, deploys only two BBs for temporal fusion instead of four. This strategy has achieved slightly worse results for the temporal range and bird’s eye view approaches while being faster. It is a more efficient alternative in terms of runtime, which provides the best $mIoU$ and still considerably improves the mPQ .

Table 7.30: Evaluation of the combined temporal multi view architecture on the validation set of SemanticKITTI. Temporal memories are successively added to both views.

Approach	RV	BEV	mPQ	$mIoU$	$t_{\text{inf}} \pm \text{std}$ in ms
MVNet			0.592	0.657	76.0 ± 1.7
MVNet+T-RV	✓		0.624	<u>0.699</u>	85.8 ± 1.8
MVNet+T-BEV		✓	0.649	<u>0.680</u>	92.2 ± 1.7
T-MVNet	✓	✓	0.662	<u>0.702</u>	103.1 ± 1.7
T-MVNet-Lite	✓	✓	0.645	0.705	92.9 ± 1.7

In the next step, the temporal multi view approach is further extended by the sensor fusion approach, see Section 6.2. However, no further improvements are achieved on SemanticKITTI. The main reason is the small overlapping FoV

of lidar and camera, which reduces the point cloud size to approximately $1/6$. While benefits for the range fusion approach can still be shown, the temporal training is unsuccessful on the strongly reduced training data. Since nuScenes provides full camera coverage, a more comprehensive evaluation of all possible combinations is performed thereon.

NuScenes offers a 360° overlapping camera FoV which allows using the entire lidar scan for sensor fusion approaches. Table 7.31 shows the results for all possible combinations, starting with the already presented results for the range view improved with temporal or camera information. Both enhancements achieve similar semantic results, while the sensor fusion improves the panoptic quality by a larger margin. Moreover, their combination provides a multimodal single view approach, improving the panoptic segmentation even further and considerably outperforming the individual enhancements.

Table 7.31: Combination of the contributions of this thesis, which ultimately provides a multimodal multi view framework, and their results on the validation set of nuScenes.

Approach	MV	Temp	SF	mPQ	$mIoU$	$t_{inf} \pm \text{std in ms}$
RVNet				0.614	0.704	32.1 ± 2.3
T-RVNet		✓		0.669	<u>0.744</u>	41.3 ± 2.1
SF-RVNet			✓	0.707	0.743	47.0 ± 0.8
TSF-RVNet		✓	✓	0.733	<u>0.771</u>	52.2 ± 2.1
MVNet	✓			0.651	<u>0.756</u>	57.6 ± 1.5
T-MVNet	✓	✓		0.744	<u>0.791</u>	71.2 ± 3.0
SF-MVNet	✓		✓	0.745	0.799	80.7 ± 1.8
TSF-MVNet	✓	✓	✓	0.784	<u>0.828</u>	102.8 ± 3.8

The lower part of Table 7.31 considers the multi view architecture combined with temporal or sensor fusion, both enhancing the results by a similar and large margin. Consequently, all combinations of two frameworks significantly enhance the semantic and panoptic segmentation, underlining the benefits of the proposed combinations. It is worth mentioning that the range view backbone, as part of the range fusion backbone, is no longer frozen but further optimized in these combined setups. This change is required to successfully train

the range fusion backbone as part of the multi view and temporal framework. The final line of Table 7.31 shows the overall framework of this thesis combining all three contributions and achieving the best results. It provides significantly improved semantic and panoptic segmentation and successfully combines the benefits of the multi view, temporal, and sensor fusion framework.

Next, the individual class results for MVNet, T-MVNet, and TSF-MVNet are compared, representing the incremental combinations of the three contributions. The individual class *IoUs* predominantly benefit from the temporal fusion in the first step and further from combined temporal and sensor fusion in the second step, which reflects the strongly improved *mIoU*. Especially the semantic segmentation results of thing classes depicted in Table 7.32 are considerably enhanced. But also the already convincing results of MVNet for stuff classes are further improved by temporal and sensor fusion, shown in Table 7.33. The findings for the *PQ* on class level are similar, and all classes benefit from the temporal fusion. In addition, every class but one additionally benefits from the combined temporal and sensor fusion, explaining the great improvement of the *mPQ*. As the combination of all proposed contributions, TSF-MVNet provides a significantly enhanced panoptic segmentation for all classes.

Table 7.32: Comparison of the class-wise panoptic results for thing classes on the validation set of nuScenes.

Approach	Metric	barrier	bicycle	bus	car	con-vehicle	mcycle	pedestrian	traffic-cone	trailer	truck	mean things
MVNet	<i>IoU</i>	.800	.335	.875	.924	.501	.804	.779	.656	.680	.811	.716
T-MVNet		.823	.440	.945	.943	.492	.853	.850	.721	.712	.855	.763
TSF-MVNet		.840	.631	.944	.935	.663	.913	.884	.780	.770	.852	.821
MVNet	<i>PQ</i>	.455	.465	.606	.879	.300	.753	.788	.774	.429	.595	.604
T-MVNet		.674	.644	.782	.911	.469	.849	.857	.831	.611	.737	.737
TSF-MVNet		.685	.795	.841	.916	.670	.894	.865	.856	.634	.795	.795

Table 7.33: Comparison of the class-wise panoptic results for stuff classes on the validation set of nuScenes.

Approach	Metric	driveable	other-flat	sidewalk	terrain	manmade	vegetation	mean stuff
MVNet		.966	.701	.748	.748	.894	.870	.821
T-MVNet	<i>IoU</i>	.969	.745	.761	.760	.907	.886	.838
TSF-MVNet		.969	.745	.756	.767	.913	.891	.840
MVNet		.963	.532	.667	.510	.867	.825	.727
T-MVNet	<i>PQ</i>	.966	.577	.698	.552	.888	.850	.755
TSF-MVNet		.966	.595	.705	.564	.895	.872	.766

In the next step, several qualitative examples illustrate the benefits but also different errors of the proposed approaches. The first example is depicted in Fig. 7.12, where instance and semantic segmentation are shown. Two types of errors are observable for MVNet. First, it fails to separate all parking cars ■ into individual instances, tagged with (a) and (b). Since the semantic segmentation is correct for these points, the error has its origin in the center and offset predictions. The temporal information exploited by T-MVNet resolves these errors and accurately splits the cars into individual instances. The second error (c) is the wrong semantic segmentation of the construction vehicle ■ on the right, which is confused with the background classes building ■ and vegetation ■. Even with temporal information, T-MVNet is unable to separate it from the background. Consequently, the entire instance is missed. However, TSF-MVNet succeeds in detecting the construction vehicle based on the additional camera information. Furthermore, it correctly segments the upper part of the hydraulic lift, which is unlabeled ■ in the ground truth. This example underlines that all contributions are valuable and necessary to reduce instance and semantic errors and provide high quality panoptic results.

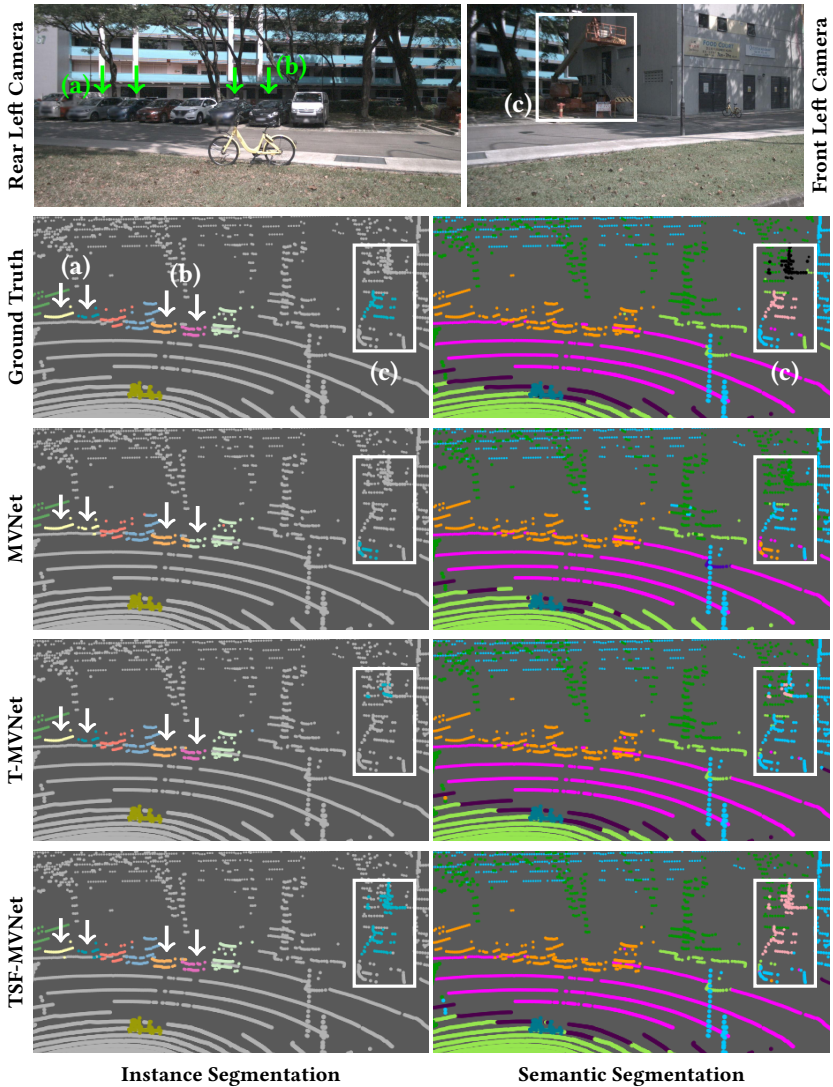













Figure 7.12: Benefits of combining the individual contributions. Temporal and sensor fusion are required to resolve instance (a), (b) and semantic errors (c). An overview of the semantic class colors is depicted in Figs. 7.3 and 7.4.

Since errors in the panoptic segmentation predominantly originate from semantic errors, the next examples focus thereon. Figure 7.13 illustrates the semantic benefits of T-MVNet and its temporal fusion, which correctly classifies the paved center strip . In contrast, MVNet assumes a vegetated center strip which is considered terrain . While MVNet correctly classifies the center strip in some preceding and subsequent frames, the temporal fusion provides a more robust semantic segmentation without these occasional errors. The example in Fig. 7.13 also illustrates the benefits of sensor fusion for distant objects. The parking pickup, which belongs to the truck class  in nuScenes, has only one row of very few measured lidar points. As a result, MVNet and T-MVNet fail to predict the correct class and confuse it with cars . In contrast, the additional information of the higher resolution camera image enables TSF-MVNet to segment the pickup truck successfully, despite the sparse lidar information.

The next scenario, depicted in Fig. 7.14, shows the value of temporal information (a), required to distinguish between buildings  and barriers  successfully. The reason is again the increased temporal robustness, which eliminates occasional errors in individual frames. Overall, T-MVNet considerably reduces the semantic errors in this example. TSF-MVNet further improves the segmentation and is able to segment the sidewalk  more accurately (b). The camera information helps to distinguish between paved and vegetated  flat ground.

The last example shows a failure case of the overall contribution TSF-MVNet. In the scene shown in Fig. 7.15, MVNet provides convincing results but misses the sidewalk between the vegetated terrain . T-MVNet makes similar errors and lacks in improving the semantic segmentation. Even TSF-MVNet fails and, in addition, introduces additional errors by confusing the bus  with a truck . One potential reason is the rainy weather and water on the camera pane. With the limited dataset size, robustness to different weather conditions is difficult to achieve. Consequently, different weather conditions might negatively impact the fusion results in some examples, see Fig. 7.15, whereas in other ones, such as Fig. 7.14, TSF-MVNet provides its full benefits.

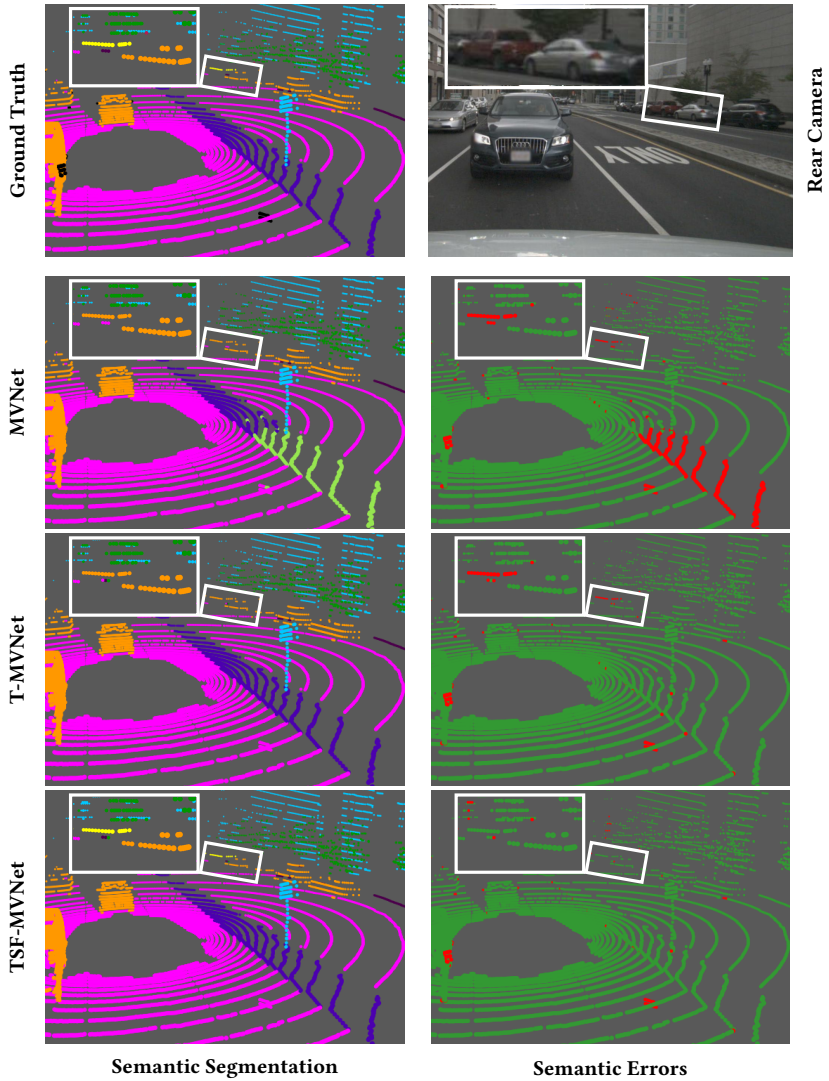


Figure 7.13: Temporal information supports the robust segmentation of the ground classes, and the high resolution camera image helps with distant points. The semantic errors ■ across all approaches are shown on the right.

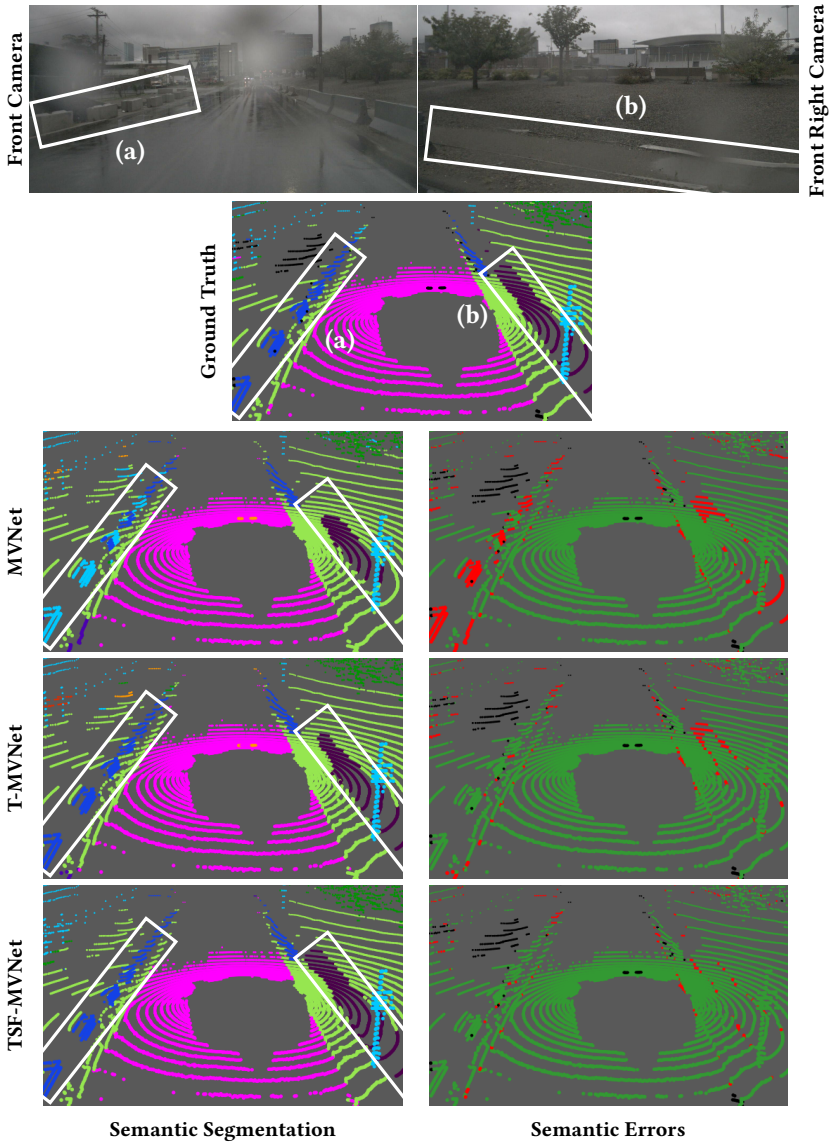


Figure 7.14: Temporal information supports a more stable differentiation of background classes (a). Sensor fusion provides a more accurate differentiation of ground classes (b).

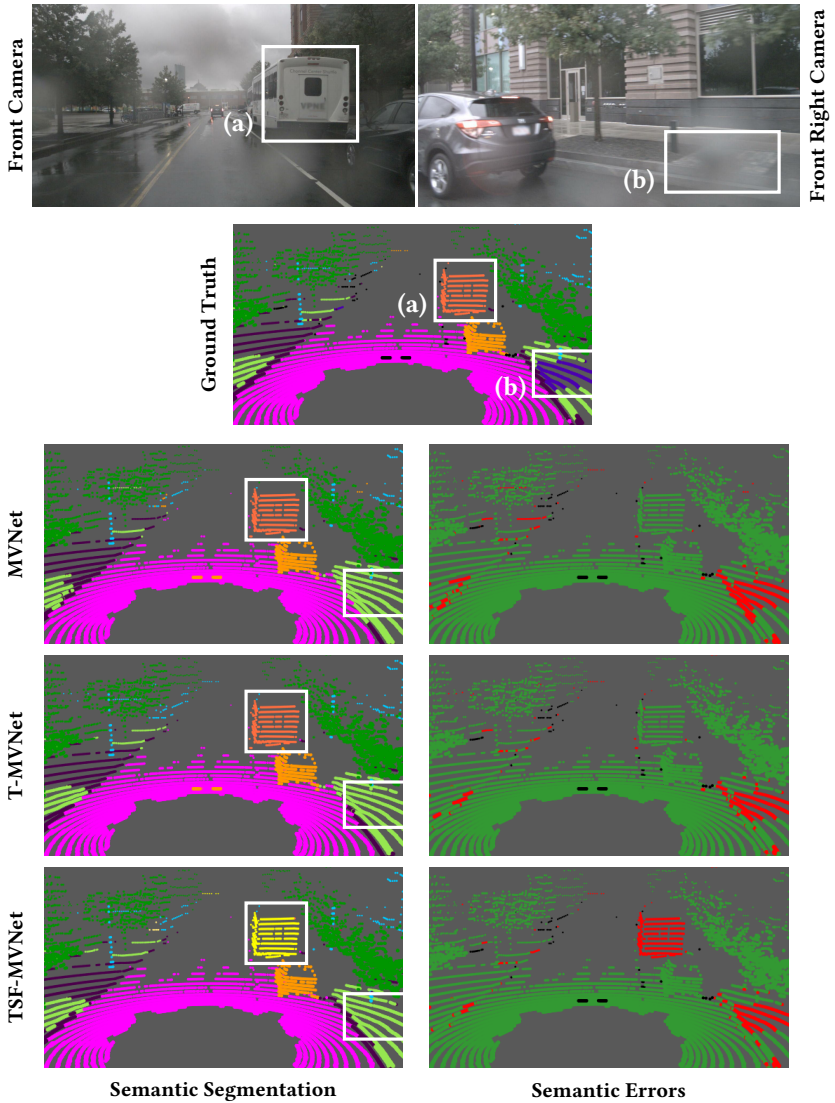


Figure 7.15: Failure case of the proposed contributions. T-MVNet is not able to improve the segmentation over MVNet, and TSF-MVNet introduces additional errors.

Overall, the conducted experiments confirm that the main contributions of this thesis not only provide their benefits individually but that their combination provides even greater improvements. Several examples illustrate these benefits. All proposed combinations are capable of simultaneously exploiting the benefits of the multi view, temporal, or sensor fusion framework, which allows choosing the combination most suitable for a given use case. The overall combination TSF-MVNet provides excellent results for 3D panoptic segmentation and achieves outstanding absolute improvements of +0.170 and +0.124 over the single view approach RVNet for mPQ and $mIoU$, respectively.

7.5.2 Comparison to State-of-the-Art

The temporal and sensor fusion approaches have already been compared to related state-of-the-art approaches. Therefore, the following comprehensive comparison has a broader scope and considers existing 3D panoptic segmentation methods in general, with a focus on the presented multi view framework and combined frameworks. In the first step, the results on the official test set of SemanticKITTI are investigated, which has the highest number of evaluated approaches. Both temporal single view approaches, the multi view approach, and the temporal multi view approach are considered.

The temporal range and bird's eye view frameworks have already been compared to existing temporal approaches for various tasks in Section 7.3.3. In addition, both approaches are more broadly compared to existing panoptic methods in the following. Both achieve convincing results compared to other approaches of their respective view, depicted in Table 7.34. T-RVNet accomplishes compelling results close to the best range view-based approaches while having a significantly lower computational complexity. T-BEVNet provides the best results based on the bird's eye view. As part of the temporal or multimodal multi view architecture, T-RVNet and T-BEVNet are not designed to outperform state-of-the-art approaches standalone. However, the temporal framework can be combined with stronger backbones to achieve this, such as the proposed multi view network MVNet.

Table 7.34: Comparison of multiple contributions to the state-of-the-art panoptic segmentation on the test set of SemanticKITTI. The second best results are highlighted in italic.

View	Approach	mPQ	mRQ	mSQ	$mIoU$	t_{inf} in ms
PV	Panoster [Gas21]	0.527	0.641	0.807	0.599	-
RV	LPSAD [Mil20]	0.380	0.482	0.765	0.509	-
	PanTrackNet [Hur20]	0.431	0.539	0.788	0.526	-
	Panoptic4D [Ayg21]	0.503	0.610	0.816	0.613	-
	CPSeg [Li21]	0.570	0.688	0.822	0.627	94
	EfficientLPS [Sir22]	0.574	0.687	0.830	0.614	217*
	<i>T-RVNet (Ours)</i>	0.546	0.665	0.813	0.614	66.9
BEV	Pan-PolarNet [Zho21]	0.541	0.650	0.814	0.595	57*
	<i>T-BEVNet (Ours)</i>	0.554	0.662	0.826	0.629	53.5
VX	DS-Net [Hon21]	0.559	0.667	0.823	0.616	294 [Li22a]
	GP-S3Net [Raz21a]	0.600	0.721	0.820	0.708	270 [Li22a]
MV	SMAC-Seg [Li22a]	0.561	0.679	0.820	0.633	99
	SCAN [Xu22]	0.615	0.721	0.845	0.677	≈ 100
	Pan-PHNet [Li22b]	0.615	0.721	0.848	0.660	≈ 81
	<i>MVNet (Ours)</i>	0.568	0.678	0.829	0.657	76.0
	<i>T-MVNet (Ours)</i>	0.617	0.726	0.842	0.679	103.1

The multi view network achieves better results than all single view methods, with a few exceptions. EfficientLPS [Sir22] deploys several extensions and additionally uses pseudo labels to improve the panoptic segmentation and compensate for some disadvantages of the range view. Additionally, it relies on object detection for instance segmentation. As a result, it outperforms the proposed multi view approach by a small margin for mPQ . However, it does not achieve the same quality in semantic segmentation, and its computational complexity is nearly three times higher. With additional extensions [Due22], the proposed multi view approach achieves a mPQ of 0.588 and outperforms EfficientLPS also for mPQ . In contrast, the voxel-based GP-S3Net [Raz21a] and the most recent multi view methods SCAN [Xu22] and Pan-PHNet [Li22b] provide better results than MVNet.

Consequently, the three best approaches GP-S3Net, SCAN, and Pan-PHNet are compared more closely to the temporal multi view network, which outperforms all three on panoptic quality. GP-S3Net deploys a complex sparse voxel backbone and graph neural network with a very high computational complexity. Based on its state-of-the-art semantic backbone AF²-S3Net [Che21d], it achieves the best semantic segmentation. Nevertheless, its panoptic segmentation is considerably outperformed by T-MVNet, with approximately one-third of the computational complexity. SCAN combines the stronger but more expensive sparse voxel view with the point view. This combination achieves similar results as the proposed T-MVNet, which is based on the range and bird’s eye view combined with temporal fusion. However, SCAN lacks the same quality of results on nuScenes, see Table 7.36. Especially its panoptic segmentation thereon drops by a large margin. Finally, Pan-PHNet combines voxel and bird’s eye view and proposes an improved clustering for bottom-up panoptic segmentation. The latter significantly contributes to the accomplished panoptic results. These are similar to T-MVNet, which relies on the established weaker clustering but on stronger features, confirmed by its superior semantic segmentation.

The nuScenes panoptic challenge was released recently, and not all approaches evaluated on SemanticKITTI have also been evaluated on nuScenes. Its authors combined all submitted semantic segmentation and object detection approaches resulting in 1,470 possible panoptic segmentation approaches. The best three are reported as baselines and are shown in rows four to six in Table 7.35, outperforming all 2D single view methods by a large margin. While these are overall strong baselines, they use two independent networks instead of one native panoptic segmentation network. Only Pan-PHNet and the proposed T-MVNet and TSF-MVNet achieve competitive or better panoptic results. While TSF-MVNet achieves the best semantic segmentation, Pan-PHNet provides better panoptic segmentation. Interestingly, its *mPQ* on the validation set is considerably lower, see Table 7.36, and is more in line with the findings on SemanticKITTI. Unfortunately, the authors provide no explanation for the outlier results on the test set. The outcomes of other approaches, including T-MVNet and TSF-MVNet, differ only slightly across validation and test set.

Table 7.35: Comparison of the combined contributions to other methods on the **test set** of nuScenes. Some approaches have been evaluated on the test set after publication in [Fon22].

View	Approach	<i>mPQ</i>	<i>mRQ</i>	<i>mSQ</i>	<i>mIoU</i>
RV	PanTrackNet [Mil20]	0.516	0.633	0.804	0.589
	EfficientLPS [Sir22]	0.624	0.741	0.837	0.667
BEV	Pan-PolarNet [Zho21]	0.636	0.751	0.843	0.670
VX	SPVNAS [Tan20] + CP [Yin21]	0.722	0.812	0.885	0.769
	Cylinder3D++[Zhu21b] + CP [Yin21]	0.765	0.850	0.896	0.773
	AF ² -S3Net [Che21d] + CP [Yin21]	0.768	0.854	0.895	0.788
MV	Pan-PHNet [Li22b]	0.801	0.876	0.911	0.802
	<i>T-MVNet (Ours)</i>	0.747	0.835	0.893	0.792
	<i>TSF-MVNet (Ours)</i>	0.787	0.875	0.893	0.811

Table 7.36: Comparison on the **validation set** of nuScenes.

View	Approach	<i>mPQ</i>	<i>mRQ</i>	<i>mSQ</i>	<i>mIoU</i>
MV	SCAN [Xu22]	0.651	0.753	0.857	0.774
	Pan-PHNet [Li22b]	0.747	0.842	0.882	0.797
	<i>T-MVNet (Ours)</i>	0.744	0.836	0.886	0.791
	<i>TSF-MVNet (Ours)</i>	0.784	0.878	0.890	0.828

Overall, the proposed contributions achieve the best panoptic segmentation on SemanticKITTI, even without sensor fusion. TSF-MVNet achieves the best semantic segmentation among the panoptic approaches on nuScenes, the best panoptic segmentation on the validation, and the second-best on the test set. Additionally, it outperforms the best combinations of individual semantic segmentation and object detection networks, combined to panoptic approaches. These findings conclude the excellent results provided by the combined contributions of this thesis.

8 Discussion and Outlook

The overall contribution of this thesis is the proposed multimodal multi view approach for panoptic segmentation of 3D point clouds based on deep learning. It combines the benefits of the three individual contributions, multi view architecture, temporal fusion, and sensor fusion, which improve the panoptic segmentation based on different aspects. Existing approaches only exploit one of these and cannot simultaneously leverage their potential.

The proposed multi view architecture focuses on the lidar sensor to provide a superior panoptic segmentation for unstructured 3D point clouds with CNNs. Features and context are efficiently aggregated in the 2D range and bird's eye view, and provided to a point view backbone. The latter combines the multi view context and maintains a unique feature vector for every 3D point. These enhanced features are the foundation for improved results over single view approaches, which suffer from the drawbacks of the individual views.

The presented temporal framework focuses on temporal information in point cloud sequences instead of considering point clouds individually. It is based on a recursive temporal fusion of feature maps to exploit temporal dependencies. A temporal memory in range or bird's eye view aggregates and propagates information through time. In every time step, the temporal memory of the previous time step is temporally aligned to compensate for ego motion. Afterwards, it is updated with the information extracted from the latest point cloud. In the end, the memory contains temporally fused features of the current and all previous time steps, which are the basis for improved 3D panoptic segmentation over single frame approaches.

The introduced multi sensor approach focuses on exploiting the camera as an additional sensor modality. It fuses feature maps provided by lidar range view

and camera backbones at multiple scales based on two proposed deep fusion strategies. The first one iteratively fuses and refines the multi scale feature maps, whereas the second strategy follows a pyramid-based fusion pattern. The provided multi sensor feature maps considerably improve 3D panoptic segmentation over methods solely relying on lidar.

8.1 Discussion

The **multi view approach** for processing unstructured 3D point clouds with CNNs is the first contribution of this work. Its main novelty is the multi view architecture with a point view backbone connecting 2D backbones for range and bird’s eye view and repeatedly aggregating multi view features. Key properties discussed in the following are the chosen views and the aggregation architecture of the point view backbone.

The first architectural property to discuss is the choice of range and bird’s eye view. This combination can be interpreted as a separation of the 3D voxel view into two orthogonal 2D views. Consequently, their combination preserves a strong representation of 3D information, despite the underlying 2D projection. The conducted experiments also confirm this combination as highly beneficial since it outperforms individual range and bird’s eye view baselines by a large margin, see Table 7.6. In addition, state-of-the-art range or bird’s eye view approaches are also outperformed [Mil20, Hur20, Ayg21, Zho21]. One existing approach [Sir22] achieves a better *mPQ* by 0.006 but a significantly worse semantic segmentation, and its computational complexity is about three times higher. Therefore, the chosen combination of range and bird’s eye view achieves high quality results with low computational complexity and enables efficient temporal and sensor fusion. On the other hand, choosing these views is simultaneously a limitation when focusing on the best results. The 3D voxel view still contains more information than the combined range and bird’s eye views and additionally contains 3D neighborhoods. In contrast, range and bird’s eye view provide distinct and orthogonal 2D neighborhoods, both including points far apart in 3D. While the combination mitigates this drawback

to some extent, the voxel view inherently provides the actual 3D neighborhoods. Consequently, the sparse voxel view is the more powerful representation but has a higher computational complexity. However, most recent multi view approaches [Ye21b, Xu22, Li22b] gradually decreased its high computational complexity with the aid of additional views. Therefore, a promising future strategy to exploit 3D neighborhoods for the presented multi view approach is the deployment of a sparse 3D backbone instead of the proposed point view backbone.

The second architectural property to discuss is the novel point view backbone which connects the range and bird’s eye view backbone. The repeated fusion of multi view features at distinct scales provides better results than different late fusion baselines shown in Table 7.6, existing late fusion panoptic methods [Li22a], or existing early [Ali21, Ger21] and late fusion semantic segmentation approaches [Lio21]. Moreover, the decision against considering neighborhood relations in 3D point clouds based on expensive k NN-search is supported by the improved results over [Lio21, Qiu22]. They rely on this strategy in their final layer and achieve worse results while having a significantly higher computational complexity. One potential reason is that these approaches exploit 3D neighborhoods only at one scale in their final layer. As previously discussed, a more promising strategy to exploit 3D neighborhoods is the sparse voxel view, enabling multi scale 3D feature aggregation.

The second contribution of this thesis is the proposed **temporal approach** for exploiting temporal dependencies to improve 3D panoptic segmentation. Its main novelty is the recursive 2D feature map fusion composed of temporal alignment and update. The resulting novel and unique property is the independence of considered past frames and computational complexity. This property allows the temporal approach to consider past information of a potentially arbitrary number of past frames. Hence, its effectiveness in exploiting temporal information and its computational complexity are discussed in the following.

The conducted experiments underline the effectiveness of the temporal fusion, which considerably improves semantic and panoptic segmentation compared to single frame baselines, see Tables 7.11 and 7.17. Additionally, existing

temporal approaches are outperformed across various tasks, such as semantic segmentation [Shi20, Wan22a], panoptic segmentation [Wan22b], dynamic semantic segmentation [Shi20, Sch22b, Wan22a, Han22, Wan22b], and moving object segmentation [Shi20, Che21b, Mer22, Sun22]. Only one superior approach for the latter task exists [Kim22], explicitly designed and optimized solely for this task. These results underline the value of the generic temporal approach, which is not optimized for one specific task but aims for general feature enhancement. It also confirms the value of exploiting a considerable number of past frames, which is further supported by the dedicated experiments shown in Fig. 7.9. Both properties enable superior results across various tasks and provide excellent improvements over the respective single frame baseline. On the other hand, the lack of explicitly considering the movement of other traffic participants is one of the limitations and offers potential for future improvements. While Table 7.12 reveals that these classes still significantly benefit, it limits the potential in scenarios with dense traffic and high velocities of traffic participants. Considering their motion based on 3D flow is a promising future research direction to address this limitation.

The computational complexity of the temporal fusion strongly benefits from the proposed temporal alignment. It enables the reuse of previously computed feature maps in range or bird's eye view and decouples the number of considered past frames and computational complexity. As a result, the proposed temporal framework has the lowest complexity among existing approaches while simultaneously achieving the discussed superior results across various tasks. Consequently, the decoupling is a precious property.

As a third contribution, the **multi sensor approach** fuses lidar and camera feature maps to exploit camera information for panoptic segmentation of 3D point clouds. Its main novelties are the iterative and pyramid-based fusion architectures for lidar range view and camera feature maps. The following discussion reasons the choice of the range view as fusion view and the proposed fusion strategies.

Choosing the range view as fusion view is motivated by the successful combination of RGB and depth information in the image domain [Sil12]. The excellent improvements over the single sensor range view network verify this

choice. Other approaches [Zhu21c, Zha21b] also successfully fuse camera and lidar information in this view. On the other hand, several methods for 3D object detection [Phi20, Liu22] successfully transform and fuse feature maps of multiple cameras into bird’s eye view. Exploiting this potential for panoptic segmentation would be promising future work, especially considering the multimodal multi view architecture discussed later. Nevertheless, the range view is a convincing choice, confirmed by the achieved enhancements and the low computational complexity of the geometric transformation from camera to lidar range view.

The presented multi scale fusion architectures significantly improve panoptic segmentation compared to the single scale fusion baseline, see Table 7.23, and early fusion [Elm19, Vor20] or single scale deep fusion [Mey19a] strategies deployed by existing semantic segmentation approaches. Consequently, multi scale fusion is important to leverage the full potential of sensor fusion. Employing these strategies inside an independent fusion branch combined with the proposed and evaluated training strategy additionally provides robustness against sensor failure. This crucial property is another advantage over existing approaches. The superior results of the extended pyramid strategy indicate the full potential of sensor fusion at the cost of an over-proportionate computational complexity. Hence, it was not employed in the multimodal multi view architecture due to this limitation. However, these results indicate additional potential, which requires further research for efficient exploitation.

The presented **multimodal multi view** architecture is the overall contribution of this thesis and combines all individual contributions into a unified architecture. A careful and aligned design of the individual approaches and contributions is the foundation of their successful combination. The temporal approach is applicable to 2D lidar views and, thereby, an excellent extension to the multi view framework with its 2D views and backbones. On the other hand, the sensor fusion approach seamlessly replaces the lidar range view backbone. The presented architecture can *simultaneously* exploit all three individual contributions, as shown by the experiments in Table 7.31. Furthermore, it outperforms all existing panoptic approaches on SemanticKITTI, and

all approaches on nuScenes when considering semantic segmentation, see Tables 7.34 and 7.35. To the best of the author’s knowledge, no approaches exist that combine these three key technologies for dense prediction tasks, such as 3D panoptic segmentation. The only other methods combining temporal and sensor fusion, 4D-Net [Pie21] and LIFT [Zen22], tackle the task of 3D object detection. However, both are single view approaches due to the targeted task of object detection, where multi view approaches are less beneficial and rather uncommon. Additional and significant differences are that 4D-Net performs early fusion for temporal fusion, and LIFT relies on attention for sensor and temporal fusion.

One additional advantage of the proposed approach is its low computation complexity. It requires approximately 100 ms to predict a 3D panoptic segmentation for the entire 360° environment with temporal and sensor fusion of *six* cameras. This inference time is considered real-time for a common lidar recording frequency of 10 Hz. Obviously, no server-grade GPU will be available when deployed in an autonomous vehicle or robot. However, the runtime was measured in single precision using float32. Switching to half precision based on float16 halves the inference time without notably affecting the results. Further significant reduction can be achieved with int8 quantization and quantization-aware training. While depending on the available hardware, these optimizations allow embedded real-time deployment.

While the combined architecture provides excellent results with low computational complexity, it also comes with some challenges and limitations. The multi view approach with its three views complicates temporal and sensor fusion since it is challenging to benefit all views. It is convincingly solved for temporal fusion but requires the memory in both 2D views to cover range and bird’s eye view feature maps. On the other hand, sensor fusion is only integrated into the range view branch, which achieves great improvements. However, it does not affect the instance feature maps of the bird’s eye view branch and potentially misses exploiting the full capabilities of sensor fusion. Therefore, fusing the camera and lidar in bird’s eye view, as a replacement or in addition to the range view fusion, might provide a further enhanced 3D

panoptic segmentation. At first glance, performing temporal and sensor fusion once in the point view resolves these drawbacks. However, this strategy would again not improve the bird’s eye view feature maps for the instance predictions. Furthermore, the temporal experiments with GRUs revealed that context aggregation after temporal fusion is crucial, which is missing in the proposed point view backbone. One possibility to address these limitations is a bidirectional information flow, which propagates not only range and bird’s eye view features to the point view, but also the fused multi view features back to the 2D views. As a result, the bird’s eye view and instance feature maps would benefit from sensor fusion in range view and vice versa.

When considering the **overall results** achieved by the multimodal multi view approach, there is still a considerable gap to accomplish a *mIoU* or *mPQ* of 1.0, despite exploiting temporal and sensor fusion. Likewise, other state-of-the-art methods are unable to close this gap. Two underlying reasons can be identified that considerably impact the panoptic results.

The first reason is the imperfection of the datasets, which suffer from label errors and considerable class imbalance. Label errors negatively impact the learning of underlying concepts and lead to errors in the evaluation and metrics. Class imbalance complicates the learning of underlying concepts for rare classes since only a few examples are present in the training set. Furthermore, vaguely defined semantic classes with high intra-class variance, such as other-vehicle or other-ground, are very hard to learn. These factors have a larger impact for large and meaningful test sets with many semantic classes, such as SemanticKITTI. For smaller test sets with fewer semantic classes, such as nuScenes, higher values for *mPQ* and *mIoU* are achieved. Since these factors are independent of the chosen approach, they cannot be significantly mitigated by temporal or sensor fusion.

The second reason is the lidar sensor itself with its varying sparsity. Lidar point clouds get increasingly sparse with growing distance to the sensor. Consequently, distant points have only a few or no other points in their immediate neighborhood. However, classifying an individual point without context is hardly possible. In addition, distant instances of thing classes are often covered by very few points and must be recognized based on these. Temporal and

sensor fusion can mitigate these effects to some extent. However, temporal fusion cannot provide benefits in the distance of the driving direction since no past measurements exist in this area. While cameras with their high resolution are very valuable, their provided information about the environment also decreases with increasing distance.

Overall, large and diverse datasets containing many instances and points for all classes and high quality labels are the most promising step towards considerably higher mPQ and $mIoU$. Furthermore, improved lidar resolution and approaches can further contribute towards metrics closer to the maximum.

8.2 Outlook

Although the proposed multimodal multi view approach achieves excellent results, future research has the potential to further enhance the proposed architecture and achieved panoptic results. Some of the proposed research directions have already been identified in the previous discussion and comprise all three contributions as well as the instance clustering:

- The information flow in the multi view architecture is unidirectional from the 2D backbones to the point view backbone. A promising future enhancement is the propagation of the fused multi view features back to the individual views, similar to [Xu21a]. However, this method considers different views and provides only semantic segmentation. Exploiting bidirectional information flow for the proposed multi view panoptic approach has the additional potential to enhance the instance predictions in bird's eye view based on multi view features.
- The sparse 3D voxel view has been identified as the most powerful individual view in literature. Replacing the point view backbone with an efficient sparse voxel view backbone offers the potential of 3D context aggregation, as addition to the deployed 2D aggregation in range and bird's eye view. Recent approaches [Ye21b, Ye21a] achieve promising semantic segmentation based on small backbones to counteract the

computational complexity drawback of the sparse voxel view. With future research to further reduce the complexity, this replacement has the potential to provide enhanced panoptic segmentation while maintaining a low computational complexity.

- The temporal approach can be further enhanced by explicitly considering the motion of traffic participants. A promising direction is the prediction of 3D flow for every 3D point. This flow information can be projected into both 2D views and extend the temporal alignment to consider the additional flow information. This extension is most beneficial in scenarios with many fast-moving objects. While the velocity-dependent degradation of the temporal approach is small, further reduction is still valuable.
- Recently, multi camera fusion in bird’s eye view for 3D object detection has shown great success. However, it has not yet been investigated for semantic or panoptic segmentation in a multi view setup. Since the deployed range view fusion is independent of the instance branch, sensor fusion in bird’s eye view has the potential to improve the overall panoptic segmentation further since it benefits semantic *and* instance features.
- The clustering of thing points into instances is done greedily in this thesis and was not the focus of its contributions. However, the clustering itself has a significant potential to improve the panoptic segmentation [Li22b]. Nevertheless, existing clustering algorithms rely on several thresholds and parameters. Learning these from the underlying data can potentially enhance the results and omits the necessity to tune these parameters by hand.

When aiming for an improved panoptic segmentation, it is often a trade-off between improved results and computational complexity. If a low computational complexity is important, the research in the mentioned directions can focus on finding efficient extensions. Otherwise, research can focus on fully exploiting the respective potential to achieve greater improvements for 3D panoptic segmentation.

Bibliography

- [Aks20] AKSOY, Eren Erdal; BACI, Saimir and CAVDAR, Selcuk: “SalsaNet: Fast Road and Vehicle Segmentation in LiDAR Point Clouds for Autonomous Driving”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 926–932.
- [Ali21] ALI ALNAGGAR, Yara; AFIFI, Mohamed; AMER, Karim and ELHELW, Mohamed: “Multi Projection Fusion for Real-time Semantic Segmentation of 3D LiDAR Point Clouds”. In: *2021 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1799–1808.
- [Arm16] ARMENI, Iro; SENER, Ozan; ZAMIR, Amir R.; JIANG, Helen; BRILAKIS, Ioannis; FISCHER, Martin and SAVARESE, Silvio: “3D Semantic Parsing of Large-Scale Indoor Spaces”. In: *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1534–1543.
- [Ayg21] AYGÜN, Mehmet; OŠEP, Aljoša; WEBER, Mark; MAXIMOV, Maxim; STACHNISS, Cyrill; BEHLEY, Jens and LEAL-TAIXÉ, Laura: “4D Panoptic LiDAR Segmentation”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 5523–5533.
- [Beh19] BEHLEY, Jens; GARBADE, Martin; MILIOTO, Andres; QUENZEL, Jan; BEHNKE, Sven; STACHNISS, Cyrill and GALL, Jürgen: “SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9296–9306.

- [Ben94] BENGIO, Y.; SIMARD, P. and FRASCONI, P.: “Learning Long-Term Dependencies with Gradient Descent is Difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.
- [Ber18] BERMAN, Maxim; TRIKI, Amal Rannen and BLASCHKO, Matthew B.: “The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4413–4421.
- [Bis06] BISHOP, Christopher M and NASRABADI, Nasser M: *Pattern Recognition and Machine Learning*. Vol. 4. 4. Springer, 2006.
- [Bou20] BOULCH, Alexandre: “ConvPoint: Continuous Convolutions for Point Cloud Processing”. In: *Computers & Graphics* 88 (2020), pp. 24–34.
- [Bro09] BROSTOW, Gabriel J; FAUQUEUR, Julien and CIPOLLA, Roberto: “Semantic Object Classes in Video: A High-Definition Ground Truth Database”. In: *Pattern Recognition Letters* 30.2 (2009), pp. 88–97.
- [Cae20] CAESAR, Holger; BANKITI, Varun; LANG, Alex H.; VORA, Sourabh; LIONG, Venice Erin; XU, Qiang; KRISHNAN, Anush; PAN, Yu; BALDAN, Giancarlo and BEIJBOM, Oscar: “nuScenes: A Multimodal Dataset for Autonomous Driving”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11618–11628.
- [Cas18] CASAS, Sergio; LUO, Wenjie and URTASUN, Raquel: “IntentNet: Learning to Predict Intention from Raw Sensor Data”. In: *Conference on Robot Learning (CoRL)*. 2018, pp. 947–956.
- [Che17a] CHEN, Liang-Chieh; PAPANDREOU, George; SCHROFF, Florian and ADAM, Hartwig: “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).
- [Che17b] CHEN, Xiaozhi; MA, Huimin; WAN, Ji; LI, Bo and XIA, Tian: “Multi-view 3D Object Detection Network for Autonomous Driving”. In: *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6526–6534.

-
- [Che18] CHEN, Liang-Chieh; PAPANDREOU, George; KOKKINOS, Iasonas; MURPHY, Kevin and YUILLE, Alan L.: “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 (2018), pp. 834–848.
- [Che20] CHENG, Bowen; COLLINS, Maxwell D.; ZHU, Yukun; LIU, Ting; HUANG, Thomas S.; ADAM, Hartwig and CHEN, Liang-Chieh: “Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 12472–12482.
- [Che21a] CHEN, Shaoyu; FANG, Jiemin; ZHANG, Qian; LIU, Wenyu and WANG, Xinggong: “Hierarchical Aggregation for 3D Instance Segmentation”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 15447–15456.
- [Che21b] CHEN, Xieyuanli; LI, Shijie; MERSCH, Benedikt; WIESMANN, Louis; GALL, Jürgen; BEHLEY, Jens and STACHNISS, Cyrill: “Moving Object Segmentation in 3D LiDAR Data: A Learning-Based Approach Exploiting Sequential Data”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 6529–6536.
- [Che21c] CHENG, Ran; RAZANI, Ryan; REN, Yuan and BINGBING, Liu: “S3Net: 3D LiDAR Sparse Semantic Segmentation Network”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 14040–14046.
- [Che21d] CHENG, Ran; RAZANI, Ryan; TAGHAVI, Ehsan; LI, Enxu and LIU, Bingbing: “(AF)2-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 12542–12551.
- [Che22] CHEN, Zehui; LI, Zhenyu; ZHANG, Shiquan; FANG, Liangji; JIANG, Qinghong; ZHAO, Feng; ZHOU, Bolei and ZHAO, Hang: “AutoAlign: Pixel-Instance Feature Aggregation for Multi-Modal 3D

- Object Detection”. In: *2022 International Joint Conference on Artificial Intelligence (IJCAI) (2022)*, pp. 827–833.
- [Cho14] CHO, Kyunghyun; MERRIËNBOER, Bart an; BAHDANAU, Dzmitry and BENGIO, Yoshua: “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *Syntax, Semantics and Structure in Statistical Translation Workshop (SSSTW)*. 2014, pp. 103–111.
- [Cho19] CHOY, Christopher; GWAK, JunYoung and SAVARESE, Silvio: “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3070–3079.
- [Chu14] CHUNG, Junyoung; GULCEHRE, Caglar; CHO, KyungHyun and BENGIO, Yoshua: “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).
- [Com02] COMANICIU, D. and MEER, P.: “Mean shift: A Robust Approach Toward Feature Space Analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 603–619.
- [Cor16] CORDTS, Marius; OMRAN, Mohamed; RAMOS, Sebastian; REHFELD, Timo; ENZWEILER, Markus; BENENSON, Rodrigo; FRANKE, Uwe; ROTH, Stefan and SCHIELE, Bernt: “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3213–3223.
- [Cor20] CORTINHAL, Tiago; TZELEPIS, George and ERDAL AKSOY, Eren: “SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds”. In: *International Symposium on Visual Computing (ISVC)*. 2020, pp. 207–222.
- [Dai17a] DAI, Angela; CHANG, Angel X.; SAVVA, Manolis; HALBER, Maciej; FUNKHOUSER, Thomas and NIEBNER, Matthias: “ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes”. In: *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2432–2443.

- [Dai17b] DAI, Jifeng; QI, Haozhi; XIONG, Yuwen; LI, Yi; ZHANG, Guodong; HU, Han and WEI, Yichen: “Deformable Convolutional Networks”. In: *2017 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, pp. 764–773.
- [Den09] DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai and FEI-FEI, Li: “ImageNet: A Large-Scale Hierarchical Image Database”. In: *2009 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255.
- [Dud00] DUDA, Richard O; HART, Peter E and STORK, David G: *Pattern Classification*, 2nd Edition. Wiley Hoboken, 2000.
- [Elm19] EL MADAWI, Khaled; RASHED, Hazem; EL SALLAB, Ahmad; NASR, Omar; KAMEL, Hanan and YOGAMANI, Senthil: “RGB and LiDAR fusion based 3D Semantic Segmentation for Autonomous Driving”. In: *2019 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 7–12.
- [Els18] EL SALLAB, Ahmad; SOBH, Ibrahim; ZIDAN, Mahmoud; ZAHRAN, Mohamed and ABDELKARIM, Sherif: “Yolo4D: A spatio-temporal Approach for Real-time Multi-Object Detection and Classification from LiDAR Point Clouds”. In: *Advances in Neural Information Processing Systems Workshops (NeurIPS W)*. 2018.
- [Eng17] ENGELMANN, Francis; KONTOGIANNI, Theodora; HERMANS, Alexander and LEIBE, Bastian: “Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds”. In: *2017 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 716–724.
- [Eng19] ENGELMANN, Francis; KONTOGIANNI, Theodora; SCHULT, Jonas and LEIBE, Bastian: “Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds”. In: *2019 European Conference on Computer Vision Workshops (ECCVW)*. 2019, pp. 395–409.
- [Eng20a] ENGELMANN, Francis; BOKELOH, Martin; FATHI, Alireza; LEIBE, Bastian and NIEBNER, Matthias: “3D-MPA: Multi-Proposal Aggregation for 3D Semantic Instance Segmentation”. In: *2020 IEEE/CVF*

- Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9028–9037.
- [Eng20b] ENGELMANN, Francis; KONTOGIANNI, Theodora and LEIBE, Bastian: “Dilated Point Convolutions: On the Receptive Field Size of Point Convolutions on 3D Point Clouds”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9463–9469.
- [Est96] ESTER, Martin; KRIEGEL, Hans-Peter; SANDER, Jörg; XU, Xiaowei et al.: “A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *KDD*. Vol. 96. 34. 1996, pp. 226–231.
- [Fan21] FAN, Hehe; YU, Xin; DING, Yuhang; YANG, Yi and KANKANHALLI, Mohan: “PSTNet: Point Spatio-Temporal Convolution on Point Cloud Sequences”. In: *2021 International Conference on Learning Representations (ICLR)*. 2021.
- [Fon22] FONG, Whye Kit; MOHAN, Rohit; HURTADO, Juana Valeria; ZHOU, Lubing; CAESAR, Holger; BEIJBOM, Oscar and VALADA, Abhinav: “Panoptic Nuscenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3795–3802.
- [For12] FORSYTH, David A and PONCE, Jean: *Computer Vision: A Modern Approach*, 2nd Edition. Prentice Hall, 2012.
- [Gao19] GAO, Naiyu; SHAN, Yanhu; WANG, Yupei; ZHAO, Xin; YU, Yanan; YANG, Ming and HUANG, Kaiqi: “SSAP: Single-Shot Instance Segmentation With Affinity Pyramid”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 642–651.
- [Gas21] GASPERINI, Stefano; MAHANI, Mohammad-Ali Nikouei; MARCOS-RAMIRO, Alvaro; NAVAB, Nassir and TOMBARI, Federico: “Panoster: End-to-End Panoptic Segmentation of LiDAR Point Clouds”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3216–3223.

- [Gei12] GEIGER, Andreas; LENZ, Philip and URTASUN, Raquel: “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *2012 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.
- [Ger21] GERDZHEV, Martin; RAZANI, Ryan; TAGHAVI, Ehsan and BING-BING, Liu: “TORNADO-Net: mulTiview tOtal vaRiationN semAntic segmentation with Diamond inceptiOn module”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9543–9549.
- [Gey20] GEYER, Jakob; KASSAHUN, Yohannes; MAHMUDI, Mentar; RICOU, Xavier; DURGESH, Rupesh; CHUNG, Andrew S; HAUSWALD, Lorenz; PHAM, Viet Hoang; MÜHLEGG, Maximilian; DORN, Sebastian et al.: “A2D2: Audi Autonomous Driving Dataset”. In: *arXiv preprint arXiv:2004.06320* (2020).
- [Glo10] GLOROT, Xavier and BENGIO, Yoshua: “Understanding the Difficulty of Training Deep Feedforward Neural Networks”. In: *2010 International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010, pp. 249–256.
- [Goo16] GOODFELLOW, Ian; BENGIO, Yoshua and COURVILLE, Aaron: *Deep Learning*. MIT Press, 2016.
- [Gra15] GRAHAM, Ben: “Sparse 3D Convolutional Neural Networks”. In: *2015 British Machine Vision Conference (BMVC) (2015)*, pp. 150.1–150.9.
- [Gra18] GRAHAM, Benjamin; ENGELCKE, Martin and MAATEN, Laurens van der: “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 9224–9232.
- [Guo21] GUO, Yulan; WANG, Hanyun; HU, Qingyong; LIU, Hao; LIU, Li and BENNAMOUN, Mohammed: “Deep Learning for 3D Point Clouds: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (2021), pp. 4338–4364.

- [Hac17] HACKEL, Timo; SAVINOV, Nikolay; LADICKY, Lubor; WEGNER, Jan D; SCHINDLER, Konrad and POLLEFEYS, Marc: “Semantic3d.net: A new Large-scale Point Cloud Classification Benchmark”. In: *arXiv preprint arXiv:1704.03847* (2017).
- [Han20] HAN, Lei; ZHENG, Tian; XU, Lan and FANG, Lu: “OccuSeg: Occupancy-Aware 3D Instance Segmentation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2937–2946.
- [Han22] HANYU, Shi; JIACHENG, Wei; HAO, Wang; FAYAO, Liu and GUOSHENG, Lin: “Learning Spatial and Temporal Variations for 4D Point Cloud Segmentation”. In: *arXiv preprint arXiv:2207.04673* (2022).
- [He15a] HE, Kaiming and SUN, Jian: “Convolutional Neural Networks at Constrained Time Cost”. In: *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5353–5360.
- [He15b] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing and SUN, Jian: “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *2015 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.
- [He16] HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing and SUN, Jian: “Deep Residual Learning for Image Recognition”. In: *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.
- [He17] HE, Kaiming; GKIOXARI, Georgia; DOLLÁR, Piotr and GIRSHICK, Ross: “Mask R-CNN”. In: *2017 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.
- [Her18] HERRMANN, Christian: Video-to-Video Face Recognition for Low-Quality Surveillance Data. KIT Scientific Publishing, 2018.
- [Hoc97] HOCHREITER, Sepp and SCHMIDHUBER, Jürgen: “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

- [Hon21] HONG, Fangzhou; ZHOU, Hui; ZHU, Xinge; LI, Hongsheng and LIU, Ziwei: “LiDAR-based Panoptic Segmentation via Dynamic Shifting Network”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 13085–13094.
- [Hou19] HOU, Ji; DAI, Angela and NIEBNER, Matthias: “3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4416–4425.
- [Hu20a] HU, Peiyun; ZIGLAR, Jason; HELD, David and RAMANAN, Deva: “What You See is What You Get: Exploiting Visibility for 3D Object Detection”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 10998–11006.
- [Hu20b] HU, Qingyong; YANG, Bo; XIE, Linhai; ROSA, Stefano; GUO, Yulan; WANG, Zhihua; TRIGONI, Niki and MARKHAM, Andrew: “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11105–11114.
- [Hua16] HUANG, Jing and YOU, Suya: “Point Cloud Labeling using 3D Convolutional Neural Network”. In: *2016 International Conference on Pattern Recognition (ICPR)*. 2016, pp. 2670–2675.
- [Hua20a] HUANG, Rui; ZHANG, Wanyue; KUNDU, Abhijit; PANTOFARU, Caroline; ROSS, David A; FUNKHOUSER, Thomas and FATHI, Alireza: “An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 266–282.
- [Hua20b] HUANG, Tengting; LIU, Zhe; CHEN, Xiwu and BAI, Xiang: “EP-Net: Enhancing Point Features with Image Semantics for 3D Object Detection”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 35–52.
- [Hur20] HURTADO, Juana Valeria; MOHAN, Rohit; BURGARD, Wolfram and VALADA, Abhinav: “MOPT: Multi-Object Panoptic Tracking”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020.

- [Ian16] IANDOLA, Forrest N; HAN, Song; MOSKEWICZ, Matthew W; ASHRAF, Khalid; DALLY, William J and KEUTZER, Kurt: “SqueezeNet: AlexNet-level Accuracy with 50x fewer Parameters and <0.5 MB Model Size”. In: *arXiv preprint arXiv:1602.07360* (2016).
- [Iof15] IOFFE, Sergey and SZEGEDY, Christian: “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *2015 International Conference on Machine Learning (ICML)*. 2015, pp. 448–456.
- [Jar09] JARRETT, Kevin; KAVUKCUOGLU, Koray; RANZATO, Marc’Aurelio and LECUN, Yann: “What is the Best Multi-Stage Architecture for Object Recognition?”. In: *2009 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2009, pp. 2146–2153.
- [Jia18] JIANG, Mingyang; WU, Yiran; ZHAO, Tianqi; ZHAO, Zelin and LU, Cewu: “PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation”. In: *arXiv preprint arXiv:1807.00652* (2018).
- [Jia20a] JIANG, Haiyong; YAN, Feilong; CAI, Jianfei; ZHENG, Jianmin and XIAO, Jun: “End-to-End 3D Point Cloud Instance Segmentation Without Detection”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 12793–12802.
- [Jia20b] JIANG, Li; ZHAO, Hengshuang; SHI, Shaoshuai; LIU, Shu; FU, Chi-Wing and JIA, Jiaya: “PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4866–4875.
- [Kim22] KIM, Jaeyeul; Woo, Jungwan and IM, Sunghoon: “RVMOS: Range-View Moving Object Segmentation Leveraged by Semantic and Motion Features”. In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 8044–8051.
- [Kin15] KINGMA, Diederik and BA, Jimmy: “Adam: A Method for Stochastic Optimization”. In: *2015 International Conference on Learning Representations Workshops (ICLRW)* (2015).

- [Kir19] KIRILLOV, Alexander; HE, Kaiming; GIRSHICK, Ross; ROTHER, Carsten and DOLLÁR, Piotr: “Panoptic Segmentation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9396–9405.
- [Kni21] KNIGHTS, Joshua; MOGHADAM, Peyman; FOOKES, Clinton and SRIDHARAN, Sridha: “Point Cloud Segmentation Using Sparse Temporal Local Attention”. In: *arXiv preprint arXiv:2112.00289* (2021).
- [Kri12] KRIZHEVSKY, Alex; SUTSKEVER, Ilya and HINTON, Geoffrey E: “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012, pp. 84–90.
- [Kri20] KRISPEL, Georg; OPITZ, Michael; WALTNER, Georg; POSSEGER, Horst and BISCHOF, Horst: “FuseSeg: LiDAR Point Cloud Segmentation Fusing Multi-Modal Data”. In: *2020 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 1863–1872.
- [Ku18] KU, Jason; MOZIFIAN, Melissa; LEE, Jungwook; HARAKEH, Ali and WASLANDER, Steven L.: “Joint 3D Proposal Generation and Object Detection from View Aggregation”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–8.
- [Kur21] KURUP, Akhil and Bos, Jeremy: “DSOR: A Scalable Statistical Filter for Removing Falling Snow from Lidar Point Clouds in Severe Winter Weather”. In: *arXiv preprint arXiv:2109.07078* (2021).
- [Lah19] LAHOUD, Jean; GHANEM, Bernard; OSWALD, Martin R. and POLLEFEYS, Marc: “3D Instance Segmentation via Multi-Task Metric Learning”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9255–9265.
- [Lan18] LANDRIEU, Loic and SIMONOVSKY, Martin: “Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4558–4567.

- [Lan19a] LANDRIEU, Loic and BOUSSAHA, Mohamed: “Point Cloud Over-segmentation With Graph-Structured Deep Metric Learning”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7432–7441.
- [Lan19b] LANG, Alex H.; VORA, Sourabh; CAESAR, Holger; ZHOU, Lubing; YANG, Jiong and BEJBOM, Oscar: “PointPillars: Fast Encoders for Object Detection From Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12689–12697.
- [Lec98] LECUN, Y.; BOTTOU, L.; BENGIO, Y. and HAFFNER, P.: “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [Li18] LI, Yangyan; BU, Rui; SUN, Mingchao; WU, Wei; DI, Xinhan and CHEN, Baoquan: “PointCNN: Convolution On X-Transformed Points”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 828–838.
- [Li21] LI, Enxu; RAZANI, Ryan; XU, Yixuan and LIU, Bingbing: “CPSeg: Cluster-free Panoptic Segmentation of 3D LiDAR Point Clouds”. In: *arXiv preprint arXiv:2111.01723* (2021).
- [Li22a] LI, Enxu; RAZANI, Ryan; XU, Yixuan and LIU, Bingbing: “SMAC-Seg: LiDAR Panoptic Segmentation via Sparse Multi-directional Attention Clustering”. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 9207–9213.
- [Li22b] LI, Jinke; HE, Xiao; WEN, Yang; GAO, Yuan; CHENG, Xiaoqiang and ZHANG, Dan: “Panoptic-PHNet: Towards Real-Time and High-Precision LiDAR Panoptic Segmentation via Clustering Pseudo Heatmap”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 11809–11818.
- [Li22c] LI, Xiaoyan; ZHANG, Gang; PAN, Hongyu and WANG, Zhenhua: “CPGNet: Cascade Point-Grid Fusion Network for Real-Time LiDAR Semantic Segmentation”. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 11117–11123.

- [Li22d] LI, Yin hao; GE, Zheng; YU, Guanyi; YANG, Jinrong; WANG, Zengran; SHI, Yukang; SUN, Jianjian and LI, Zeming: “BEVDepth: Acquisition of Reliable Depth for Multi-View 3D Object Detection”. In: *arXiv preprint arXiv:2206.10092* (2022).
- [Lia18] LIANG, Ming; YANG, Bin; WANG, Shenlong and URTASUN, Raquel: “Deep Continuous Fusion for Multi-sensor 3D Object Detection”. In: *2018 European Conference on Computer Vision (ECCV)*. 2018, pp. 663–678.
- [Lia19] LIANG, Ming; YANG, Bin; CHEN, Yun; HU, Rui and URTASUN, Raquel: “Multi-Task Multi-Sensor Fusion for 3D Object Detection”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7337–7345.
- [Lia20] LIANG, Zhidong; YANG, Ming; LI, Hao and WANG, Chunxiang: “3D Instance Embedding Learning with a Structure-aware Loss Function for Point Cloud Segmentation”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4915–4922.
- [Lia21] LIANG, Zhihao; LI, Zhihao; XU, Songcen; TAN, Mingkui and JIA, Kui: “Instance Segmentation in 3D Scenes using Semantic Superpoint Tree Networks”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 2763–2772.
- [Lin14] LIN, Tsung-Yi; MAIRE, Michael; BELONGIE, Serge; HAYS, James; PERONA, Pietro; RAMANAN, Deva; DOLLÁR, Piotr and ZITNICK, C Lawrence: “Microsoft COCO: Common Objects in Context”. In: *2014 European Conference on Computer Vision (ECCV)*. 2014, pp. 740–755.
- [Lin17] LIN, Tsung-Yi; DOLLÁR, Piotr; GIRSHICK, Ross; HE, Kaiming; HARIHARAN, Bharath and BELONGIE, Serge: “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944.

- [Lio21] LIONG, Venice Erin; NGUYEN, Thi Ngoc Tho; WIDJAJA, Sergi; SHARMA, Dhananjai and CHONG, Zhuang Jie: “AMVNet: Assertion-based Multi-View Fusion Network for LiDAR Semantic Segmentation”. In: *2021 International Joint Conferences on Artificial Intelligence Workshop (IJCAIW)*. 2021.
- [Liu19a] LIU, Xingyu; YAN, Mengyuan and BOHG, Jeannette: “MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9245–9254.
- [Liu19b] LIU, Zhijian; TANG, Haotian; LIN, Yujun and HAN, Song: “Point-Voxel CNN for Efficient 3D Deep Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019, pp. 963–973.
- [Liu22] LIU, Zhijian; TANG, Haotian; AMINI, Alexander; YANG, Xinyu; MAO, Huizi; RUS, Daniela and HAN, Song: “BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird’s-Eye View Representation”. In: *arXiv preprint arXiv:2205.13542* (2022).
- [Lon15] LONG, Jonathan; SHELHAMER, Evan and DARRELL, Trevor: “Fully Convolutional Networks for Semantic Segmentation”. In: *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.
- [Luo18] LUO, Wenjie; YANG, Bin and URTASUN, Raquel: “Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3569–3577.
- [Maa13] MAAS, Andrew L; HANNUN, Awni Y; NG, Andrew Y et al.: “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *2013 International Conference on Machine Learning (ICML)*. 2013.
- [Mao22] MAO, Jiageng; SHI, Shaoshuai; WANG, Xiaogang and LI, Hongsheng: “3D Object Detection for Autonomous Driving: A Review and New Outlooks”. In: *arXiv preprint arXiv:2206.09474* (2022).

- [Mat15] MATURANA, Daniel and SCHERER, Sebastian: “VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 922–928.
- [McC20] McCRAE, Scott and ZAKHOR, Avideh: “3D Object Detection for Autonomous Driving using Temporal Lidar Data”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 2661–2665.
- [Mer22] MERSCH, Benedikt; CHEN, Xieyuanli; VIZZO, Ignacio; NUNES, Lucas; BEHLEY, Jens and STACHNISS, Cyrill: “Receding Moving Object Segmentation in 3D LiDAR Data Using Sparse 4D Convolutions”. In: *IEEE Robotics and Automation Letters 7.3 (2022)*, pp. 7503–7510.
- [Mey19a] MEYER, Gregory P.; CHARLAND, Jake; HEGDE, Darshan; LADDHA, Ankit and VALLESPI-GONZALEZ, Carlos: “Sensor Fusion for Joint 3D Object Detection and Semantic Segmentation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 1230–1237.
- [Mey19b] MEYER, Gregory P.; LADDHA, Ankit; KEE, Eric; VALLESPI-GONZALEZ, Carlos and WELLINGTON, Carl K.: “LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12669–12678.
- [Mil19] MILIOTO, Andres; VIZZO, Ignacio; BEHLEY, Jens and STACHNISS, Cyrill: “RangeNet++: Fast and Accurate LiDAR Semantic Segmentation”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4213–4220.
- [Mil20] MILIOTO, Andres; BEHLEY, Jens; MCCOOL, Chris and STACHNISS, Cyrill: “LiDAR Panoptic Segmentation for Autonomous Driving”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 8505–8512.

- [Moh20] MOHAN, Rohit and VALADA, Abhinav: “EfficientPS: Efficient Panoptic Segmentation”. In: *International Journal of Computer Vision* 129.5 (2020), pp. 1551–1579.
- [Mot14] MOTTAGHI, Roozbeh; CHEN, Xianjie; LIU, Xiaobai; CHO, Nam-Gyu; LEE, Seong-Whan; FIDLER, Sanja; URTASUN, Raquel and YUILLE, Alan: “The Role of Context for Object Detection and Semantic Segmentation in the Wild”. In: *2014 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 891–898.
- [Nai10] NAIR, Vinod and HINTON, Geoffrey E: “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *2010 International Conference on Learning Representations (ICLR)*. 2010.
- [Pan20a] PAN, Yancheng; GAO, Biao; MEI, Jilin; GENG, Sib0; LI, Chengkun and ZHAO, Huijing: “SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 687–693.
- [Pan20b] PANG, Su; MORRIS, Daniel and RADHA, Hayder: “CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10386–10393.
- [Pan22] PANG, Su; MORRIS, Daniel and RADHA, Hayder: “Fast-CLOCs: Fast Camera-LiDAR Object Candidates Fusion for 3D Object Detection”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 3747–3756.
- [Pha19] PHAM, Quang-Hieu; NGUYEN, Thanh; HUA, Binh-Son; ROIG, Gemma and YEUNG, Sai-Kit: “JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds With Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8819–8828.

- [Phi20] PHILION, Jonah and FIDLER, Sanja: “Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 194–210.
- [Pie21] PIERGIOVANNI, AJ; CASSER, Vincent; RYOO, Michael S. and ANGELOVA, Anelia: “4D-Net for Learned Multi-Modal Alignment”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 15415–15425.
- [Por19] PORZI, Lorenzo; BULÒ, Samuel Rota; COLOVIC, Aleksander and KONTSCIEDER, Peter: “Seamless Scene Segmentation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8269–8278.
- [Qi16] QI, Charles R.; SU, Hao; NIEßNER, Matthias; DAI, Angela; YAN, Mengyuan and GUIBAS, Leonidas J.: “Volumetric and Multi-View CNNs for Object Classification on 3D Data”. In: *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5648–5656.
- [Qi17a] QI, Charles R.; SU, Hao; KAICHUN, Mo and GUIBAS, Leonidas J.: “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 77–85.
- [Qi17b] QI, Charles Ruizhongtai; YI, Li; SU, Hao and GUIBAS, Leonidas J.: “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017, pp. 5099–5108.
- [Qi18] QI, Charles R.; LIU, Wei; WU, Chenxia; SU, Hao and GUIBAS, Leonidas J.: “Frustum PointNets for 3D Object Detection from RGB-D Data”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 918–927.
- [Qiu21] QIU, Shi; ANWAR, Saeed and BARNES, Nick: “Semantic Segmentation for Real Point Cloud Scenes via Bilateral Augmentation and Adaptive Fusion”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 1757–1767.

- [Qiu22] QIU, Haibo; YU, Baosheng and TAO, Dacheng: “GFNet: Geometric Flow Network for 3D Point Cloud Semantic Segmentation”. In: *arXiv preprint arXiv:2207.02605* (2022).
- [Raz21a] RAZANI, Ryan; CHENG, Ran; LI, Enxu; TAGHAVI, Ehsan; REN, Yuan and BINGBING, Liu: “GP-S3Net: Graph-based Panoptic Sparse Semantic Segmentation Network”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16056–16065.
- [Raz21b] RAZANI, Ryan; CHENG, Ran; TAGHAVI, Ehsan and BINGBING, Liu: “Lite-HDseg: LiDAR Semantic Segmentation Using Lite Harmonic Dense Convolutions”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9550–9556.
- [Red18] REDMON, Joseph and FARHADI, Ali: “Yolov3: An Incremental Improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).
- [Ret18] RETHAGE, Dario; WALD, Johanna; STURM, Jürgen; NAVAB, Nassir and TOMBARI, Federico: “Fully-Convolutional Point Networks for Large-Scale Point Clouds”. In: *2018 European Conference on Computer Vision (ECCV)*. 2018, pp. 625–640.
- [Ron15] RONNEBERGER, Olaf; FISCHER, Philipp and BROX, Thomas: “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *2015 International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, pp. 234–241.
- [Ros57] ROSENBLATT, Frank: *The Perceptron, a Perceiving and Recognizing Automaton* (Project Para). Cornell Aeronautical Laboratory, 1957.
- [Roy18] ROYNARD, Xavier; DESCHAUD, Jean-Emmanuel and GOULETTE, François: “Paris-Lille-3D: A Large and High-quality Ground-truth Urban Point Cloud Dataset for Automatic Segmentation and Classification”. In: *International Journal of Robotics Research* 37.6 (2018), pp. 545–557.
- [Rum86] RUMELHART, David; HINTON, Geoffrey and WILLIAMS, Ronald: “Learning Representations by Back-propagating Errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

- [Rus15] RUSSAKOVSKY, Olga; DENG, Jia; SU, Hao; KRAUSE, Jonathan; SATHEESH, Sanjeev; MA, Sean; HUANG, Zhiheng; KARPATY, Andrej; KHOSLA, Aditya; BERNSTEIN, Michael et al.: “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.
- [SAE18] SAE: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Tech. rep. SAE International, 2018.
- [Sch10] SCHERER, Dominik; MÜLLER, Andreas and BEHNKE, Sven: “Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition”. In: *2010 International Conference on Artificial Neural Networks (ICANN)*. 2010, pp. 92–101.
- [Sch22b] SCHUTT, Peer; ROSU, Radu Alexandru and BEHNKE, Sven: “Abstract Flow for Temporal Semantic Segmentation on the Permutohedral Lattice”. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 5139–5145.
- [Shi15] SHI, Xingjian; CHEN, Zhourong; WANG, Hao; YEUNG, Dit-Yan; WONG, Wai-kin and WOO, Wang-chun: “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 802–810.
- [Shi16] SHI, Wenzhe; CABALLERO, Jose; HUSZÁR, Ferenc; TOTZ, Johannes; AITKEN, Andrew P.; BISHOP, Rob; RUECKERT, Daniel and WANG, Zehan: “Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network”. In: *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1874–1883.
- [Shi20] SHI, Hanyu; LIN, Guosheng; WANG, Hao; HUNG, Tzu-Yi and WANG, Zhenhua: “SpSequenceNet: Semantic Segmentation Network on 4D Point Clouds”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4573–4582.

- [Sia17] SIAM, Mennatullah; VALIPOUR, Sepehr; JAGERSAND, Martin and RAY, Nilanjan: “Convolutional Gated Recurrent Networks for Video Segmentation”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 3090–3094.
- [Sil12] SILBERMAN, Nathan; HOIEM, Derek; KOHLI, Pushmeet and FERGUS, Rob: “Indoor Segmentation and Support Inference from RGBD Images”. In: *2012 European Conference on Computer Vision (ECCV)*. 2012, pp. 746–760.
- [Sim14] SIMONYAN, Karen and ZISSERMAN, Andrew: “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *2014 International Conference on Learning Representations (ICLR)*. 2014.
- [Sim19] SIMON, Martin; MILZ, Stefan; AMENDE, Karl and GROSS, Horst-Michael: “Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds”. In: *2019 European Conference on Computer Vision Workshops (ECCVW)*. 2019, pp. 197–209.
- [Sin15] SINGH, Santokh: Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey. Tech. rep. 2015.
- [Sin19] SINDAGI, Vishwanath A.; ZHOU, Yin and TUZEL, Oncel: “MVX-Net: Multimodal VoxelNet for 3D Object Detection”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7276–7282.
- [Sir22] SIROHI, Kshitij; MOHAN, Rohit; BÜSCHER, Daniel; BURGARD, Wolfram and VALADA, Abhinav: “EfficientLPS: Efficient LiDAR Panoptic Segmentation”. In: *IEEE Transactions on Robotics* 38.3 (2022), pp. 1894–1914.
- [Smu07] SMUCKER, Mark D; ALLAN, James and CARTERETTE, Ben: “A Comparison of Statistical Significance Tests for Information Retrieval Evaluation”. In: *2007 ACM International Conference on Information and Knowledge Management (CIKM)*. 2007, pp. 623–632.

- [Son15] SONG, Shuran; LICHTENBERG, Samuel P. and XIAO, Jianxiong: “SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite”. In: *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 567–576.
- [Spr15] SPRINGENBERG, Jost Tobias; DOSOVITSKIY, Alexey; BROX, Thomas and RIEDMILLER, Martin: “Striving for Simplicity: The All Convolutional Net”. In: *2015 International Conference on Learning Representations Workshops (ICLRW)* (2015).
- [Sri15] SRIVASTAVA, Rupesh Kumar; GREFF, Klaus and SCHMIDHUBER, Jürgen: “Highway Networks”. In: *2015 International Conference on Learning Representations (ICLR)*. 2015.
- [Sun22] SUN, Jiadai; DAI, Yuchao; ZHANG, Xianjing; XU, Jintao; AI, Rui; GU, Weihao and CHEN, Xieyuanli: “Efficient Spatial-Temporal Information Fusion for LiDAR-Based 3D Moving Object Segmentation”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2022), pp. 11456–11463.
- [Sze15] SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent and RABINOVICH, Andrew: “Going Deeper with Convolutions”. In: *2015 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.
- [Tan19] TAN, Mingxing and LE, Quoc: “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *2019 International Conference on Machine Learning (ICML)*. 2019, pp. 6105–6114.
- [Tan20] TANG, Haotian; LIU, Zhijian; ZHAO, Shengyu; LIN, Yujun; LIN, Ji; WANG, Hanrui and HAN, Song: “Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 685–702.
- [Tat18] TATARCHENKO, Maxim; PARK, Jaesik; KOLTUN, Vladlen and ZHOU, Qian-Yi: “Tangent Convolutions for Dense Prediction in 3D”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3887–3896.

- [Tch17] TCHAPMI, Lyne; CHOY, Christopher; ARMENI, Iro; GWAK, JunY-oung and SAVARESE, Silvio: “SEGCloud: Semantic Segmentation of 3D Point Clouds”. In: *2017 International Conference on 3D Vision (3DV)*. 2017, pp. 537–547.
- [Tho19] THOMAS, Hugues; QI, Charles R.; DESCHAUD, Jean-Emmanuel; MARCOTEGUI, Beatriz; GOULETTE, François and GUIBAS, Leonidas: “KPCConv: Flexible and Deformable Convolution for Point Clouds”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6410–6419.
- [Una21] UNAL, Ozan; VAN GOOL, Luc and DAI, Dengxin: “Improving Point Cloud Semantic Segmentation by Learning 3D Object Detection”. In: *2021 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 2949–2958.
- [Vor20] VORA, Sourabh; LANG, Alex H.; HELOU, Bassam and BEIJBOM, Oscar: “PointPainting: Sequential Fusion for 3D Object Detection”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4603–4611.
- [Vu22] VU, Thang; KIM, Kookhoi; LUU, Tung M; NGUYEN, Thanh and YOO, Chang D: “SoftGroup for 3D Instance Segmentation on Point Clouds”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 2708–2717.
- [Wan15] WANG, Dominic Zeng and POSNER, Ingmar: “Voting for Voting in Online Point Cloud Object Detection”. In: *Robotics: Science and Systems 1.3* (2015), pp. 10–15.
- [Wan18a] WANG, Shenlong; SUO, Simon; MA, Wei-Chiu; POKROVSKY, Andrei and URTASUN, Raquel: “Deep Parametric Continuous Convolutional Neural Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2589–2597.
- [Wan18b] WANG, Weiyue; YU, Ronald; HUANG, Qiangui and NEUMANN, Ulrich: “SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2569–2578.

- [Wan19a] WANG, Xinlong; LIU, Shu; SHEN, Xiaoyong; SHEN, Chunhua and JIA, Jiaya: “Associatively Segmenting Instances and Semantics in Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4091–4100.
- [Wan19b] WANG, Zhixin and JIA, Kui: “Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1742–1749.
- [Wan21a] WANG, Chunwei; MA, Chao; ZHU, Ming and YANG, Xiaokang: “PointAugmenting: Cross-Modal Augmentation for 3D Object Detection”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11789–11798.
- [Wan21b] WANG, Zejie; ZHAO, Zhen; JIN, Zhao; CHE, Zhengping; TANG, Jian; SHEN, Chaomin and PENG, Yaxin: “Multi-Stage Fusion for Multi-Class 3D Lidar Detection”. In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2021, pp. 3113–3121.
- [Wan22a] WANG, Song; ZHU, Jianke and ZHANG, Ruixiang: “Meta-RangeSeg: LiDAR Sequence Semantic Segmentation Using Multiple Feature Aggregation”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9739–9746.
- [Wan22b] WANG, Weiqi; YOU, Xiong; YANG, Jian; SU, Mingzhan; ZHANG, Lantian; YANG, Zhenkai and KUANG, Yingcai: “LiDAR-Based Real-Time Panoptic Segmentation via Spatiotemporal Sequential Data Fusion”. In: *Remote Sensing* 14.8 (2022), p. 1775.
- [Web21] WEBER, Mark et al.: “STEP: Segmenting and Tracking Every Pixel”. In: *Advances in Neural Information Processing Systems Workshops (NeurIPSW)*. 2021.
- [Wil90] WILLIAMS, Ronald J. and PENG, Jing: “An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories”. In: *Neural Computation* 2.4 (1990), pp. 490–501.

- [Wu18] WU, Bichen; WAN, Alvin; YUE, Xiangyu and KEUTZER, Kurt: “SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1887–1893.
- [Wu19a] WU, Bichen; ZHOU, Xuanyu; ZHAO, Sicheng; YUE, Xiangyu and KEUTZER, Kurt: “SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud”. In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 4376–4382.
- [Wu19b] WU, Wenxuan; QI, Zhongang and FUXIN, Li: “PointConv: Deep Convolutional Networks on 3D Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9613–9622.
- [Xia21] XIAO, Pengchuan et al.: “PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3095–3101.
- [Xio19] XIONG, Yuwen; LIAO, Renjie; ZHAO, Hengshuang; HU, Rui; BAI, Min; YUMER, Ersin and URTASUN, Raquel: “UPSNet: A Unified Panoptic Segmentation Network”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 8810–8818.
- [Xu20] XU, Chenfeng; WU, Bichen; WANG, Zining; ZHAN, Wei; VAJDA, Peter; KEUTZER, Kurt and TOMIZUKA, Masayoshi: “Squeezesegv3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 1–19.
- [Xu21a] XU, Jianyun; ZHANG, Ruixiang; DOU, Jian; ZHU, Yushi; SUN, Jie and PU, Shiliang: “RPVNet: A Deep and Efficient Range-Point-Voxel Fusion Network for LiDAR Point Cloud Segmentation”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16004–16013.

- [Xu21b] XU, Shaoqing; ZHOU, Dingfu; FANG, Jin; YIN, Junbo; BIN, Zhou and ZHANG, Liangjun: “FusionPainting: Multimodal Fusion with Adaptive Attention for 3D Object Detection”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3047–3054.
- [Xu22] XU, Shuangjie; WAN, Rui; YE, Maosheng; ZOU, Xiaoyi and CAO, Tongyi: “Sparse Cross-scale Attention Network for Efficient LiDAR Panoptic Segmentation”. In: *2022 Conference on Artificial Intelligence (AAAI)*. 2022, pp. 2920–2928.
- [Yan18] YANG, Bin; LUO, Wenjie and URTASUN, Raquel: “PIXOR: Real-time 3D Object Detection from Point Clouds”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7652–7660.
- [Yan19a] YANG, Bo; WANG, Jianan; CLARK, Ronald; HU, Qingyong; WANG, Sen; MARKHAM, Andrew and TRIGONI, Niki: “Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019, pp. 6737–6746.
- [Yan19b] YANG, Jiancheng; ZHANG, Qiang; NI, Bingbing; LI, Linguo; LIU, Jinxian; ZHOU, Mengdie and TIAN, Qi: “Modeling Point Clouds With Self-Attention and Gumbel Subset Sampling”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3318–3327.
- [Yan19c] YANG, Tien-Ju; COLLINS, Maxwell D; ZHU, Yukun; HWANG, Jyh-Jing; LIU, Ting; ZHANG, Xiao; SZE, Vivienne; PAPANDREOU, George and CHEN, Liang-Chieh: “DeeperLab: Single-Shot Image Parser”. In: *arXiv preprint arXiv:1902.05093* (2019).
- [Yan20] YAN, Xu; ZHENG, Chaoda; LI, Zhen; WANG, Sheng and CUI, Shuguang: “PointASNL: Robust Point Clouds Processing Using Nonlocal Neural Networks With Adaptive Sampling”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5588–5597.

- [Yan21] YAN, Xu; GAO, Jiantao; LI, Jie; ZHANG, Ruimao; LI, Zhen; HUANG, Rui and CUI, Shuguang: “Sparse Single Sweep LiDAR Point Cloud Segmentation via Learning Contextual Shape Priors from Scene Completion”. In: *2021 Conference on Artificial Intelligence (AAAI)*. 2021, pp. 3101–3109.
- [Ye18] YE, Xiaoqing; LI, Jiamao; HUANG, Hexiao; DU, Liang and ZHANG, Xiaolin: “3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation”. In: *2018 European Conference on Computer Vision (ECCV)*. 2018, pp. 415–430.
- [Ye21a] YE, Maosheng; WAN, Rui; XU, Shuangjie; CAO, Tongyi and CHEN, Qifeng: “DRINet++: Efficient Voxel-as-point Point Cloud Segmentation”. In: *arXiv preprint arXiv:2111.08318* (2021).
- [Ye21b] YE, Maosheng; XU, Shuangjie; CAO, Tongyi and CHEN, Qifeng: “DRINet: A Dual-Representation Iterative Learning Network for Point Cloud Segmentation”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 7427–7436.
- [Yi19] YI, Li; ZHAO, Wang; WANG, He; SUNG, Minhyuk and GUIBAS, Leonidas J.: “GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3942–3951.
- [Yin20] YIN, Junbo; SHEN, Jianbing; GUAN, Chenye; ZHOU, Dingfu and YANG, Ruigang: “LiDAR-Based Online 3D Video Object Detection With Graph-Based Message Passing and Spatiotemporal Transformer Attention”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11492–11501.
- [Yin21] YIN, Tianwei; ZHOU, Xingyi and KRÄHENBÜHL, Philipp: “Center-based 3D Object Detection and Tracking”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 11779–11788.

- [Yoo20] Yoo, Jin Hyeok; KIM, Yecheol; KIM, Jisong and CHOI, Jun Won: “3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 720–736.
- [Yu18] YU, Fisher; WANG, Dequan; SHELHAMER, Evan and DARRELL, Trevor: “Deep Layer Aggregation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2403–2412.
- [Yu20] YU, Fisher; CHEN, Haofeng; WANG, Xin; XIAN, Wenqi; CHEN, Yingying; LIU, Fangchen; MADHAVAN, Vashisht and DARRELL, Trevor: “BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2633–2642.
- [Zen22] ZENG, Yihan; ZHANG, Da; WANG, Chunwei; MIAO, Zhenwei; LIU, Ting; ZHAN, Xin; HAO, Dayang and MA, Chao: “LIFT: Learning 4D LiDAR Image Fusion Transformer for 3D Object Detection”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 17151–17160.
- [Zha17] ZHAO, Hengshuang; SHI, Jianping; QI, Xiaojuan; WANG, Xiaogang and JIA, Jiaya: “Pyramid Scene Parsing Network”. In: *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6230–6239.
- [Zha18] ZHANG, Chris; LUO, Wenjie and URTASUN, Raquel: “Efficient Convolutions for Real-Time Semantic Segmentation of 3D Point Clouds”. In: *2018 International Conference on 3D Vision (3DV)*. 2018, pp. 399–408.
- [Zha19a] ZHANG, Zhiyuan; HUA, Binh-Son and YEUNG, Sai-Kit: “ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1607–1616.

- [Zha19b] ZHAO, Hengshuang; JIANG, Li; FU, Chi-Wing and JIA, Jiaya: “PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5560–5568.
- [Zha20a] ZHANG, Feihu; FANG, Jin; WAH, Benjamin and TORR, Philip: “Deep FusionNet for Point Cloud Semantic Segmentation”. In: *2020 European Conference on Computer Vision (ECCV)*. 2020, pp. 644–663.
- [Zha20b] ZHANG, Feihu; GUAN, Chenye; FANG, Jin; BAI, Song; YANG, Ruigang; TORR, Philip H.S. and PRISACARIU, Victor: “Instance Segmentation of LiDAR Point Clouds”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9448–9455.
- [Zha20c] ZHANG, Yang; ZHOU, Zixiang; DAVID, Philip; YUE, Xiangyu; XI, Zerong; GONG, Boqing and FOROOSH, Hassan: “PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9598–9607.
- [Zha20d] ZHAO, Lin and TAO, Wenbing: “JSNet: Joint Instance and Semantic Segmentation of 3D Point Clouds”. In: *2020 Conference on Artificial Intelligence (AAAI)*. 2020, pp. 12951–12958.
- [Zha21a] ZHANG, Biao and WONKA, Peter: “Point Cloud Instance Segmentation using Probabilistic Embeddings”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 8879–8888.
- [Zha21b] ZHAO, Lin; ZHOU, Hui; ZHU, Xinge; SONG, Xiao; LI, Hongsheng and TAO, Wenbing: “LIF-Seg: Lidar and Camera Image Fusion for 3D Lidar Semantic Segmentation”. In: *arXiv preprint arXiv:2108.07511* (2021).
- [Zha21c] ZHAO, Yiming; BAI, Lin and HUANG, Xinming: “FIDNet: LiDAR Point Cloud Semantic Segmentation with Fully Interpolation Decoding”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 4453–4458.

-
- [Zho17] ZHOU, Bolei; ZHAO, Hang; PUIG, Xavier; FIDLER, Sanja; BARRIUSO, Adela and TORRALBA, Antonio: “Scene Parsing through ADE20K Dataset”. In: *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5122–5130.
- [Zho18] ZHOU, Yin and TUZEL, Oncel: “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4490–4499.
- [Zho21] ZHOU, Zixiang; ZHANG, Yang and FOROOSH, Hassan: “Panoptic-PolarNet: Proposal-free LiDAR Point Cloud Panoptic Segmentation”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 13189–13198.
- [Zhu21a] ZHU, Ming; MA, Chao; JI, Pan and YANG, Xiaokang: “Cross-Modality 3D Object Detection”. In: *2021 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 3771–3780.
- [Zhu21b] ZHU, Xinge; ZHOU, Hui; WANG, Tai; HONG, Fangzhou; LI, Wei; MA, Yuexin; LI, Hongsheng; YANG, Ruigang and LIN, Dahua: “Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR-based Perception”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.10 (2021), pp. 6807–6822.
- [Zhu21c] ZHUANG, Zhuangwei; LI, Rong; JIA, Kui; WANG, Qicheng; LI, Yuanqing and TAN, Mingkui: “Perception-Aware Multi-Sensor Fusion for 3D LiDAR Semantic Segmentation”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16260–16270.

Publications

- [Due20a] DUERR, Fabian; PFALLER, Mario; WEIGEL, Hendrik and BEYERER, Jürgen: “LiDAR-based Recurrent 3D Semantic Segmentation with Temporal Memory Alignment”. In: *2020 International Conference on 3D Vision (3DV)*. 2020, pp. 781–790.
- [Due20b] DUERR, Fabian; WEIGEL, Hendrik; MAEHLISCH, Mirko and BEYERER, Jürgen: “Iterative Deep Fusion for 3D Semantic Segmentation”. In: *2020 IEEE International Conference on Robotic Computing (IRC)*. 2020, pp. 391–397.
- [Due21] DUERR, Fabian; WEIGEL, Hendrik and BEYERER, Jürgen: “Decoupled Iterative Deep Sensor Fusion for 3D Semantic Segmentation”. In: *International Journal of Semantic Computing (IJSC)* 15.03 (2021), pp. 293–312.
- [Due22] DUERR, Fabian; WEIGEL, Hendrik and BEYERER, Jürgen: “Range-Bird: Multi View Panoptic Segmentation of 3D Point Clouds with Neighborhood Attention”. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 11131–11137.
- [Sch22] SCHIEBER, Hannah; DUERR, Fabian; SCHOEN, Torsten and BEYERER, Jürgen: “Deep Sensor Fusion with Pyramid Fusion Networks for 3D Semantic Segmentation”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 375–381.

Acronyms

ADAS	Advanced Driver Assistance Systems
ASPP	Atrous Spatial Pyramid Pooling
BB	Basic Block
BEV	bird's eye view
BN	Batch Normalization
BoB	Bottleneck Block
CE	Cross-Entropy
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DLA	Deep Layer Aggregation
FA	feature aggregator
FCN	Fully Convolutional Network
FE	feature extractor
FoV	field of view

FPN	Feature Pyramid Network
FPS	Farthest Point Sampling
GRU	Gated Recurrent Unit
kNN	k -nearest neighbor
LReLU	Leaky Rectified Linear Unit
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MSA	mean-absolute error
MSE	mean-squared error
MVA	Multi View Aggregation
NHTSA	National Highway Traffic Safety Administration
NMS	Non-Maximum Suppression
pMLP	pointwise Multi-Layer Perceptron
PV	point view
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RV	range view
TBPTT	Truncated Backpropagation Through Time

VX voxel view

