IMPROVING 3D PERCEPTION FROM UNLABELED DATA

Dissertation Thesis

University: Czech Technical University in Prague
Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Artificial Intelligence and Biocybernetics
Supervisor: Prof. Ing. Tomáš Svoboda, Ph.D.
Supervisor-Specialist: Doc. Ing. Karel Zimmermann, Ph.D.

Ing. Patrik Vacek

April 2024

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Prof. Ing. Tomáš Svoboda Ph.D.) Principal Adviser

Abstract

The integration of 3D perception technologies, notably LiDAR (Light Detection and Ranging) point clouds, has revolutionized various domains such as autonomous driving and robotics. Leveraging deep learning architectures, these technologies enable machines to perceive and navigate complex environments accurately and efficiently. However, supervising the deep architectures, is impeded by the laborious and costly manual annotations required and therefore not scalable.

This thesis addresses these challenges by focusing on self-supervised and semi-supervised learning paradigms. The proposed approaches capitalize on cheap unlabeled data or data synthesis, to achieve reasonable performance without extensive annotations. The research emphasizes tasks crucial for autonomous driving, including semantic segmentation, object detection, and scene flow estimation.

To enhance scalability, a self-supervised data-driven method for simulating LiDAR sensors in game simulators for sim2real transfer is proposed, enabling the utilization of inexpensive synthetic data during model training. Additionally, a novel data augmentation framework utilizing pre-existing annotated data is introduced, significantly enhancing model performance, particularly for rare classes. Further, the temporal information inherent in LiDAR data sequences is exploited through a spatial-temporal aggregation module, enhancing semi-supervised learning. Together with multiple ensemble teachers, the new aggregation module provides high-quality pseudo-labels for student training, outperforming fully supervised methods with only small subset of manual labels. Furthermore, a self-supervised 3D scene flow framework is developed, incorporating novel consistency losses to improve flow estimation between sequential point clouds. This approach demonstrates superior performance and generalization across diverse driving datasets. Lastly, a joint optimization of flow with instance clustering is proposed, achieving state-of-the-art results, especially in dynamic scenes with multiple independently moving objects.

Collectively, these contributions advance the state-of-the-art in 3D perception tasks for autonomous driving, mitigating annotation costs, enhancing scalability, and improving generalization capabilities, thereby paving the way for more efficient and adaptable real-world applications.

Abstrakt

Integrace technologií 3D počítačového vidění, zejména bodových mraků LiDARu (Light Detection and Ranging), zásadním způsobem revolucionizovala oblasti jako je autonomní řízení a robotika. Využitím hlubokých architektur učení tyto technologie umožňují strojům vnímat a navigovat složitými prostředími relativně přesně a efektivně. Nicméně, supervize hlubokých architektur je omezena náročným a nákladným ručním anotováním, které není škálovatelné.

Tato disertační práce se zaměřuje na samo-supervizované a polo-supervizované učící paradigma. Navržené přístupy využívají levných neanotovaných či syntetizovaných dat, aby jimi naučené algoritmy dosáhly rozumného výkonu bez drahých a neškálovatelných anotací. Výzkum klade důraz na úlohy kritické pro autonomní řízení, jmenovitě sémantické segmentace, detekce objektů a odhadu toku scény.

Pro zvýšení škálovatelnosti je navržena samo-supervizovaná, daty řízená metoda pro simulaci senzorů LiDAR v herním simulátoru pro sim2real přenos, což umožňuje využití levných syntetických dat během trénování modelu. Kromě toho je představen nový postup pro augmentaci dat využívající předchozí anotovaná data, což významně zvyšuje výkon modelu, zejména pro zřídka objevující se třídy. Dále je využita temporální složka obsažená v sekvencích dat LiDARu pomocí prostorově-temporálního agregačního modulu, který zlepšuje polo-supervizované učení. Spolu se skupinou učitelských modelů poskytuje nový agregační modul kvalitnější pseudo-značky jako signál pro trénink studenta. Postup překonává některé supervizované metody s minimálním množstvím ručních anotací. Teze dále navrhuje samo-supervizovanou metodu pro odhad 3D toku scény, která zahrnuje nové ztrátové funkce využívající jak prostorovou, tak časovou konzistenci. Tento přístup vykazuje lepší výkon a generalizaci přes různé sady dat než dosavadní metody. Nakonec je navržena společná optimalizace toku se shlukováním instancí, která dosahuje lepších výsledků než stav poznání, zejména v dynamických scénách s více nezávisle se pohybujícími objekty.

Tyto příspěvky dohromady posouvají současný stav poznání v oblasti 3D vnímání pro autonomní řízení, snižují náklady na anotace a zlepšují schopnosti generalizace, čímž otevírají cestu k efektivnějším a adaptabilnějším aplikacím 3D vnímání.

Declaration

I hereby declare I have written this doctoral thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, April 2024 .....

Ing. Patrik Vacek

Acknowledgments

I thank my supervisor-specialist, Doc. Karel Zimmermann, for his advice, feedback, and friendship, which helped me finish my studies and enjoy them along the way. I am thankful for my supervisor, Prof. Tomáš Svoboda, who helped me to keep on track and who established a nice working environment. I also thank David Hurych from Valeo.ai for the collaboration and positive energy during the whole study. I am also thankful to my Alma Mater, the Czech Technical University, and especially the Department of Cybernetics, led by Prof. Tomas Svoboda during my PhD, for providing me with the facilities that allowed me to pursue my doctoral studies. I am also grateful to the Czech Science Foundation (GACR), the Research Center for Informatics (RCI), the Funding Agency of Czech Technical University in Prague, and Valeo for sponsoring my research. Mostly I am thankful to my family for financial support, guidance, invaluable lifelong relationship and all they do for me.

Contents

1 Thesis Overview and Contributions **1**

1.1 Motivation and Problem Statement 1

1.2 Aims and Contributions 2

1.3 Summary of Academic Publications 4

2 Supervision from Simulated Data. **6**

2.1 Introduction 6

2.2 Related Work 8

2.3 Method 10

2.3.1 Geometrical simulation 10

2.3.2 Data-driven intensity simulation 11

2.3.3 Learning of the intensity-predicting network 15

2.4 Experiments 17

2.4.1 Intensity prediction accuracy 17

2.4.2 Segmentation accuracy improvement 20

2.5 Discussion 22

3 Data Augmentation from Existing Labeled Data **24**

3.1 Introduction 24

3.2 Related Work 26

3.3 Method 27

3.3.1 **Road Estimation** 28

3.3.2 **Creating of Bounding Boxes** 28

3.3.3 **Placing of Objects** 30

3.3.4 **Occlusion Handling** 31

3.4 Experiments 32

3.4.1 **Datasets and Perception Tasks** 32

3.4.2 **3D Perception Models** 33

3.4.3 **Augmentations** . . . 33
3.4.4 **Evaluation of object detection and semantic segmentation** . . . 34
3.4.5 **Ablation Study of Object Detection** . . . 35
3.5 Discussion . . . 35

4 Supervision by Spatial-Temporal Aggregation with unlabeled Data **37**
4.1 Introduction . . . 37
4.2 Related Work . . . 39
4.3 Method . . . 40
4.3.1 **Point-cloud notations** . . . 40
4.3.2 **Time-aware feature extraction** . . . 40
4.3.3 **Task-dependent modules** . . . 41
4.3.4 **Training data** . . . 42
4.3.5 **Concordance and selection of pseudo-labels** . . . 42
4.4 Experiments . . . 44
4.4.1 Datasets . . . 44
4.4.2 3D Multi-Class Semantic Segmentation . . . 45
4.4.3 3D Object Detection . . . 46
4.4.4 Implementation Details . . . 47
4.5 Ablation Studies . . . 47
4.6 Discussion . . . 51

5 Motion Features From Scene Flow **53**
5.1 Introduction . . . 53
5.2 Related Work . . . 56
5.3 Methodology of Surface Regularity and Cyclic Smoothness . . . 58
5.3.1 Baseline losses of self-supervised scene flow . . . 59
5.3.2 Surface smoothness . . . 60
5.3.3 Cyclic smoothness . . . 60
5.3.4 Training objective and Architecture . . . 61
5.4 Experiments . . . 61
5.4.1 Evaluation on StereoKITTI . . . 65
5.4.2 Generalization over various LiDAR datasets . . . 65
5.4.3 Ablations studies . . . 66
5.5 Discussion to Cyclic and Surface Smoothness . . . 67
5.6 Joint Flow and Object Instance Optimization . . . 70
5.6.1 Method Overview . . . 72
5.6.2 Distance loss . . . 73

5.6.3 Hard rigidity loss . . . 73
5.6.4 Soft rigidity loss . . . 74
5.7 Experiments . . . 75
5.7.1 Comparison with State-of-the-Art Methods . . . 75
5.7.2 DBSCAN Clustering in Scene Flow . . . 76
5.7.3 Neural Prior with Hard and Soft Clusters. . . . 76
5.7.4 Results on StereoKITTI benchmark. . . . 78
5.7.5 Ablation Study . . . 79
5.8 Discussion . . . 82

6 Conclusion **84**
6.1 Conclusion . . . 84
6.2 Future Work . . . 85

A Additional Contributions **86**
A.1 Successfully supervised student theses: . . . 86
A.2 Lab tutor of bachelor courses: . . . 86
A.3 Collaboration with Industry . . . 86

B Author Publication Response **87**

Bibliography **91**

List of Tables

2.1 Distribution of hybrid loss classification bins 18
2.2 MSE error on intensity prediction 18
2.3 Evaluation of different modeled intensities on semantic segmentation performance on our SemanticKitti split 21

3.1 Semantic segmentation on SemanticKITTI 34
3.2 Object detection results with PV-RCNN 35
3.3 Real3D-Aug Object detection results with PointPillar architecture based on number of inserted classes 35
3.4 Real3D-Aug object detection results with PV-RCNN based on the number of inserted classes. 36

4.1 LiDAR semantic segmentation performance on SemanticKITTI 44
4.2 LiDAR semantic segmentation performance on nuScenes 45
4.3 LiDAR semantic segmentation performance on SemanticKITTI 45
4.4 Semi-supervised learning on SemanticKITTI 46
4.5 Results of 3D object detection 47
4.6 Effect of temporal diversity of teachers 47
4.7 Comparision to other teacher-student methods 50

5.1 Comparative results of scene flow estimation methods on stereo-based dataset 62
5.2 Performance on LiDAR datasets 64
5.3 Influence of proposed losses 65
5.4 Result of cycle losses on nuScenes 66
5.5 Performance on Argoverse2 dataset in foreground aware metrics 75
5.6 Performance on Argoverse2 dataset in object class aware EPE 75
5.7 Performance on LiDAR datasets 78
5.8 Comparison with State-of-the-Art on KITTI dataset 80
5.9 Ablation study of method components 82

5.10 Rigidity regularization ablation 82

List of Figures

2.1 Examples of simulated data 7
2.2 Extracting of LiDAR point clouds 12
2.3 Example of point cloud extraction 13
2.4 Modeling intensity from modalities 14
2.5 Label diversity in real and synthetic domain 16
2.6 Pipeline overview 17
2.7 Comparison of LiDAR intensities on the cars in the SemanticKitti dataset 19
2.8 Example of GTA scene with simulated LiDAR intensity 19
2.9 Comparison of real scene intensity 20
2.10 Example of segmentation of distant and occluded car 21
2.11 Segmentation of occluded van 22

3.1 Examples of augmentation method 25
3.2 Pipeline overview 27
3.3 Map generation 29
3.4 Creation of bounding box 30

4.1 Concordance of Teachers 38
4.2 Proposed architecture for aggregating sequence of point clouds 40
4.3 Time-aware neighborhood of a point 41
4.4 Impact of knowledge distillation 42
4.5 A scene from SemanticKITTI 48
4.6 A scene from Argoverse showing the robustness of our student model 48
4.7 Ablation on PL's selection strategy in Argoverse object detection 49
4.8 Impact of labeling proporiton 50
4.9 Distance-based improvement in class IoUs 51

5.1 Proposed self-supervised scene flow framework 54
5.2 Regularization losses overview 55

5.3 Cyclic Smoothness loss 59
5.4 Example of improvements brought by proposed framework on real LiDAR data . . . 63
5.5 Ablation of normal estimation 67
5.6 Qualitative example on StereoKITTI 68
5.7 Comparison of the proposed method with self-supervised competitors on Argoverse2 Dataset 70
5.8 Outline of the proposed losses 71
5.9 Qualitative example on Waymo Dataset 77
5.10 Flow estimation error vs. run time 79
5.11 StereoKITTI benchmark comparison 80
5.12 Visual comparison of scene flow with the state-of-the-art 81

6.1 Possible future direction of leveraging motion for 3D perception tasks 85

Chapter 1

Thesis Overview and Contributions

1.1 Motivation and Problem Statement

The integration of 3D perception technologies, particularly LiDAR (Light Detection and Ranging) point clouds, has catalyzed groundbreaking advancements across various domains such as autonomous driving and robotics [7, 11, 106, 36, 86, 105, 66]. These technologies leverage the rich spatial information provided by 3D sensors to enable machines to perceive and navigate complex environments with unprecedented accuracy and efficiency [66, 86, 49, 47, 4]. The deep learning architectures have proven instrumental in extracting meaningful features from voluminous point cloud data, enabling tasks such as object detection, segmentation, and scene understanding in large scale [4, 86, 63].

However, while supervised learning methods, particularly those based on deep learning, have demonstrated exceptional performance in leveraging this data, they are hindered by the substantial cost and labor associated with manual annotations [4, 42, 116, 86]. For instance, annotating a single point cloud for semantic segmentation in large-scale autonomous driving scenarios can be prohibitively expensive, making supervised learning approaches impractical for real-world applications. Furthermore, supervised models often struggle with generalization when faced with scenarios beyond their training data, highlighting the need for more scalable and adaptable solutions [63, 62, 116]. As the complexity of robotic tasks continues to evolve, there remains a pressing need to overcome the limitations imposed by the reliance on supervised learning paradigms.

To address these challenges, researchers are increasingly turning to alternative learning paradigms such as self-supervised [114, 115, 105] and semi-supervised learning [35, 132, 19]. These approaches leverage the abundance of unlabeled data, which is readily available from emerging 3D sensors such as RGBD cameras and LiDAR systems. By harnessing incomplete learning signals from unlabeled data, these methods offer a promising avenue for achieving reasonable performance without the need for costly annotations.

In summary, the problems of real-world 3D LiDAR perception are three-fold:

- **Scalability:** Our research aims to develop unsupervised point cloud perception algorithms that can scale effectively to large and diverse datasets without the need for manual annotation.
- **Generalization:** By leveraging unsupervised learning techniques, we seek to improve the generalization capabilities of point cloud perception models, enabling robust performance across different environments and sensor configurations.
- **Annotation Cost:** The proposed unsupervised approaches have the potential to significantly reduce the annotation cost associated with training data preparation, thereby lowering the barrier to entry for deploying point cloud perception systems in real-world applications.

1.2 Aims and Contributions

The main goal of the thesis is to progress the state-of-the-art in data-driven 3D perception tasks for autonomous driving by minimizing the aforementioned problems. Although there are many components and approaches to the goal, we focused on self-supervised and semi-supervised approaches to utilize easily obtainable unlabeled data or existing pre-annotated datasets. The thesis focuses on tasks of semantic segmentation [116, 149], object detection [35, 149] and scene flow estimation [114, 115] as it is a crucial part of autonomous driving perception.

Physical LiDAR Data Simulation. In order to increase scalability and decrease the annotation cost of the labeled data, we have proposed a data-driven method for simulating LiDAR sensors inside the game simulator for sim2real transfer, which is publicly available. The current simulators lacked the ability to synthesize LiDAR intensity feature [139, 97, 24], which is the important part of deep learning model input [139, 130, 7]. We have contributed by data-driven intensity simulation for the synthetic data in order to use the cheap simulation data during the training of models with intensity modality. We also experimentally show that enhancing the training set by such simulated data improves the segmentation accuracy on the real dataset with limited access to real data. The result was published in peer-reviewed journal [116]:

- Patrik Vacek, Otakar Jašek, Karel Zimmermann, and Tomáš Svoboda. Learning to predict lidar intensities. *IEEE Transactions on Intelligent Transportation Systems*, 23(4):3556–3564, 2022.

Data Augmentation from Existing Labels. Despite the benefits of synthetic data, we have encountered limits of the simulation as it poses a large domain gap [116]. Therefore, we proposed a data augmentation method that takes advantage of real, already annotated data with the aim of rendering augmentations as realistic as possible. The current methods usually randomly place annotated instances or synthetic CAD models [30, 135]. We have contributed by proposing an

augmentation framework that reuses real data, automatically finds suitable placements in the scene to be augmented, and handles occlusions explicitly[1]. Due to the usage of real data, the scan points of newly inserted objects in augmentation sustain the physical characteristics of the LiDAR, such as intensity and raydrop.

The pipeline proves competitive in training top-performing models for 3D object detection and semantic segmentation. The new augmentation provides a significant performance gain in rare and essential classes over the state of the art. The result was published on workshop [149]:

- Petr Šebek, Šimon Pokorný, Patrik Vacek, and Tomáš Svoboda. Real3d-Aug: Point Cloud Augmentation by Placing Real Objects with Occlusion Handling for 3D Detection and Segmentation. In *Computer Vision Winter Workshop (CVWW)*, 2023.

Semi-Supervised Temporality. Since the unlabeled data naturally comes in sequences, we decided to exploit the temporal nature of LiDAR measurements. Compared to the standard methods that pass multiple temporal frames on the input of architecture without specific temporal mechanism [58, 86], we contributed by proposing novel spatial-temporal aggregation module to group features from multiple measurements[2].

We leveraged sequences of point clouds to boost the semi-supervised pseudo-labeling technique in a teacher-student setup via training multiple ensembles of teachers, each focused on a different view of temporal information. This set of teachers, dubbed *Concordance*, provides higher quality pseudo-labels for student training than standard methods. The output of multiple teachers was combined via a novel pseudo-label confidence-guided criterion. Our approach, which uses only 20% manual labels, outperforms some fully supervised methods. A notable performance boost was achieved for classes rarely appearing in training data. The result was published in peer-reviewed journal [35]:

- Awet Haileslassie Gebrehiwot, Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Teachers in Concordance for Pseudo-Labeling of 3D Sequential Data. *IEEE Robotics and Automation Letters (R-AL)*, 8(2):536–543, 2023.

Unsupervised 3D Motion Flow Features. After examining the benefits of learning from the temporal domain, we decided to learn motion features *explicitly* in the form of a self-supervised 3D Scene flow task that research community used for boosting the performance of other tasks such as semantic segmentation [2], instance segmentation [105], scene reconstruction [63] and object detection [26]. The flow estimation between the sequential point clouds is usually optimized or regressed with some form of spatial regularization such as local smoothness or object rigidity [59, 37, 62, 119]. Relying on the assumption that scene elements are mostly rigid, current smoothness

[1] I am an author of the idea of using the real object instances with occlusion handling, which was implemented by supervised students as part of master thesis. I have also implemented the semantic segmentation experiments.

[2] I am an author of the idea of using the bigger span of temporal horizons, multiple teachers and the spatial-temporal aggregation module. I have also prepared a revised version and presented the paper at the ICRA 2023 conference.

losses are built on the definition of "rigid clusters" in the input point clouds. The definition of these clusters is challenging and has a significant impact on the quality of predicted flows [115, 114].

We proposed a novel learning framework for this task, which improves the necessary regularization. We introduced two new consistency losses that enlarge clusters while preventing them from spreading over distinct objects. In particular, we enforce *temporal* consistency with a forward-backward cyclic loss and *spatial* consistency by considering surface orientation similarity in addition to spatial proximity. The proposed losses are model-independent and can thus be used in a plug-and-play fashion to significantly improve the performance of existing models, as demonstrated on the two most widely used architectures. We also showcase the effectiveness and generalization capability of our framework on four standard sensor-unique driving datasets. The paper with the contribution was **submitted** to peer-reviewed journal [114]:

- Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Regularizing Self-supervised 3D Scene Flows with Surface Awareness and Cyclic Consistency. *Submitted to peer-reviewed Journal*, 2024.

We made another contribution by jointly optimizing the flow with instance clustering. Usually, the rigid objects are estimated by a variety of 3D spatial clustering methods. While state-of-the-art methods successfully capture overall scene motion using the Neural Prior structure, they encounter challenges in discerning multi-object motions [63, 20, 118, 119]. We identified the structural constraints and the use of large and strict rigid clusters as the main pitfall of the current approaches and we proposed a novel clustering approach that allows for combination of overlapping soft clusters with outlier rejection as well as non-overlapping rigid clusters representation. We evaluated our method on multiple datasets with LiDAR point clouds, demonstrating superior performance over the self-supervised baselines and reaching new state-of-the-art results. Our method especially excels in resolving flow in complicated dynamic scenes with multiple independently moving objects close to each other, which includes pedestrians, cyclists, and other vulnerable road users. The novel method for self-supervised flow estimation was **submitted** to peer-reviewed journal [115]:

- Patrik Vacek, David Hurych, Karel Zimmermann, and Tomas Svoboda. Let It Flow: Simultaneous Optimization of 3D Flow and Object Clustering. *Submitted to peer-reviewed Journal*, 2024.

1.3 Summary of Academic Publications

Peer-reviewed Journal publications:

- Patrik Vacek, Otakar Jašek, Karel Zimmermann, and Tomáš Svoboda. Learning to predict lidar intensities. *IEEE Transactions on Intelligent Transportation Systems*, 23(4):3556–3564, 2022

- Awet Haileslassie Gebrehiwot, Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Teachers in Concordance for Pseudo-Labeling of 3D Sequential Data. *IEEE Robotics and Automation Letters (R-AL)*, 8(2):536–543, 2023

Currently Submitted to peer-reviewed Journal:

- Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Regularizing Self-supervised 3D Scene Flows with Surface Awareness and Cyclic Consistency. *Submitted to peer-reviewed Journal*, 2024

- Patrik Vacek, David Hurych, Karel Zimmermann, and Tomas Svoboda. Let It Flow: Simultaneous Optimization of 3D Flow and Object Clustering. *Submitted to peer-reviewed Journal*, 2024

Peer-reviewed Workshop proceedings:

- Petr Šebek, Šimon Pokorný, Patrik Vacek, and Tomáš Svoboda. Real3d-Aug: Point Cloud Augmentation by Placing Real Objects with Occlusion Handling for 3D Detection and Segmentation. In *Computer Vision Winter Workshop (CVWW)*, 2023

Chapter 2

Supervision from Simulated Data.

2.1 Introduction

There have been over 1.2 billion vehicles in use over the world in 2015[1]. When a novel autonomous functionality, such as autonomous emergency braking, is to be put into operation, its reliability has to be thoroughly tested, because the impact on the accident rate is enormous. For example, if the new functionality exhibit 1 failure out of 1 million testing frames (9 hours of operating time of 30Hz sensor), the expected number of failure cases over the world per single day is over 150 million[2]. Consequently, testing on billions of frames in advance of real deployment is highly desired. It is hardly feasible to create testing set with billions of annotated frames which would cover all possible cases. In addition to that, many tasks comprise online control, which cannot be tested offline. A trustworthy simulation is the only technically tractable option.

There are several open-source simulators such as CARLA [24] or AirSim from Microsoft [97], which offer viable autonomous driving simulation with a realistic RGB camera model in a small synthetic world with a limited variety of textures and structures. In contrast to these open-source simulators, research community also reverse-engineered GTA V game engine. The mentioned game has been recently shown [52] to have a world model realistic enough for generating annotated training RGB images that improve performance on well known semantic segmentation challenges KITTI [36] or VOC [28]. Nevertheless, the simulation of other sensors, which are also essential for autonomous-driving such as LiDARs, is either missing (GTA V) or it is strictly geometry-based (CARLA).

Unfortunately, LiDAR point clouds consisting of geometry only lack information about the power of a receiving signal (*LiDAR intensity*) and therefore are not fully descriptive for modeling and evaluation of LiDAR sensor with full properties. Importance of including this LiDAR intensity as

[1] `https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide`

[2] This number is estimated as follows: Expected number of failures of a single car during one day is $\mathbb{E}\left[\mathrm{Bin}(24, 1/9)\right] = 2.5$. 95 % of vehicles are parked while the remaining 60 million cars are in motion [91], therefore the expected number of failures is $2.5 \times 60 \cdot 10^6$.

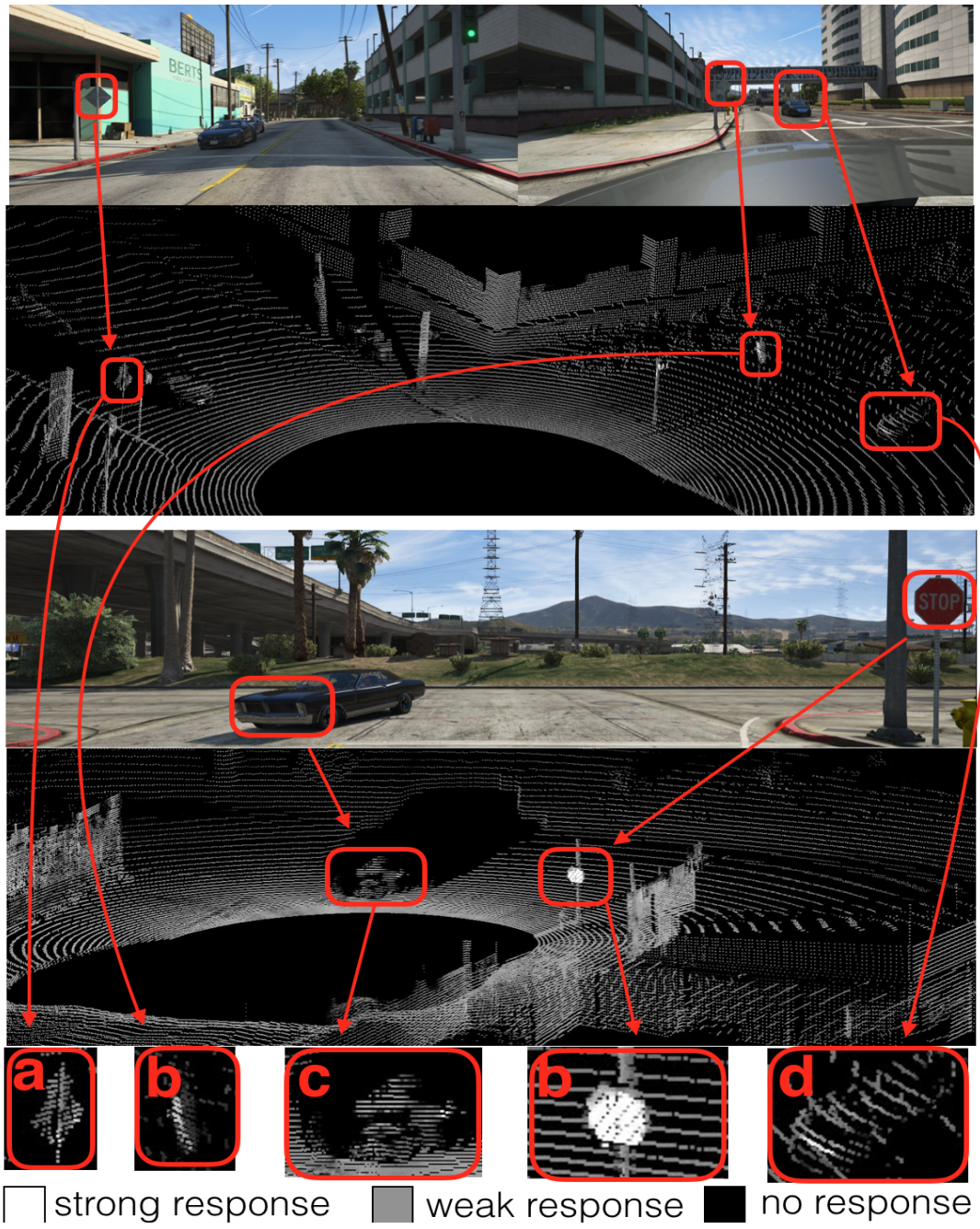


Figure 2.1: **Examples of simulated data:** Simulated RGB image and close-up of corresponding LiDAR scan with intensity encoded in grayscale. Strong responses appear consistently on reflective surfaces such as traffic signs facing towards the LiDAR (b) and license plates (d) despite the shadows in RGB images. Notice also correctly simulated systematic failures: (i) no or weak responses on the hood (c+d), (ii) weak response on the frontal mask of the bottom car which does not have a license plate (c), (iii) weak response on the traffic signs in the top image, which is facing from the LiDAR (a).

a feature has been demonstrated by [139] as it increases performance in semantic segmentation. The naive approach to model intensity feature is to map it as a monotonically decreasing function of depth. However, depth-based intensity undesirably underestimates behavior in the corner cases with unusual dispersion of the active signal, such as polished hoods, windows, and shallow puddles, registration plates, or traffic signs, see Figure 2.1 for a few examples. The material behavior is described by another contributing factor of the received signal, the reflectivity of the scanned objects [33]. However, the procedure of acquiring realistic material responses to the LiDAR beam in the simulation world would require large-scale physical specifications of generated objects. We propose to leverage other information about the object, such as its color and label description and study benefits of these modalities in prediction of LiDAR response learned from driving scenarios of the real world.

To close the gap between the real and synthetic data, we introduce and publicly release a GTA V LiDAR simulator. The simulator is trained on the real data to estimate realistic responses on unusual surfaces. The proposed method builds on top of the geometrical model, which re-projects the existing world into the LiDAR sensor. We enhance the geometrical model by modeling the strength of the LiDAR response. Modeling intensities allows injecting systematic failures and measurement noise into the geometrically simulated measurements. We experimented with two deep learning architectures [130] and [94] to learn the intensity estimation in a data-driven way. The intensity model is further used for enhancing synthetic data. We show that such data, when combined with the real training set, improves the segmentation accuracy on real testing data.

To sum up the contribution, we proposed a way of modeling intensity from the LiDAR geometry, RGB images and class label and showed that the data-driven simulation of LiDAR measurements, when combined with real training dataset, improves the segmentation accuracy on the real data. We also provided a publicly available LiDAR interface for the GTA V game, which allows for the automatic generation of synthetic annotated training and evaluation datasets and released a large public GTA V dataset for object detection and semantic segmentation from RGB+LiDAR data, which consists of approximately 40 000 frames [116]. Both source codes and dataset are available for download at `https://github.com/vras-group/lidar-intensity`.

2.2 Related Work

Large-scale LiDAR datasets. Recent advancements in the field of autonomous driving were influenced by large-scale datasets and benchmarks. This phenomenon is even more significant with the thriving success of deep learning. Kitti benchmark has set standards among public automotive datasets [36]. Besides regular RGB images of driving scenes, it also includes calibrated LiDAR readings. However, the annotation is done only in the RGB camera and therefore limited to the frontal view only. This limitation has been eliminated by the very recent SemanticKitti dataset [4],

where all points in LiDAR point clouds were annotated, excluding a few anomalies. nuScenes [7] is a recently published dataset that contains thousand driving scenarios. It is composed of 360 thousand LiDAR readings, which also include full annotations.

However, in order to build a fully autonomous vehicle, datasets of much larger magnitudes and different scenarios are necessary. Manual annotation is costly and consumes a large amount of man-hours [31], which makes it intractable for such a large scale. On top of that, datasets alone do not provide options for *validation* of autonomous driving capabilities with respect to the interpreted scene. These constraints point to the necessity of realistic and automatically annotated simulators.

Simulators with LiDAR point cloud properties. It was shown by numerous papers, that many state of the art detectors use intensity channel as a useful feature in learning segmentation from LiDAR [126, 128, 136] measurements. Intensity can provide a decisive distinction between two objects of a similar geometry by providing peak values on specific object parts similar to attention models [124], [98], [125] and therefore constitutes a valuable feature for classification tasks. Unfortunately, most of the current LiDAR simulators capable of creating a variable driving scene do not compute intensity values and offer geometry only [24, 29, 139]. Carla simulator [24] contains information about surface material, however, as far as we can tell, it cannot be leveraged to acquire LiDAR intensity. The simulator Blensor [38] offers information about material reflectivity. However, it is not possible to simulate different weather conditions. The Blensor also suffers from the fact that its base Blender was not developed for large scenes, but rather for smaller objects, and therefore, it is difficult to model a large world at the needed scale. The Virtual KITTI dataset [34] provides synthetically generated sequential images with depth information and pixel-wise annotation. The depth information can also be used to generate point clouds. However, the point clouds do not show the same characteristics as a real rotating lidar, including reflections. Also, the gap between real and synthetic data remains a great challenge [116]. One of the approaches to deal with the difference and portability to the real world is [95, 96], which can produce more realistic LiDAR data from simulation by learning GAN models.

Another option is to use computer games with state-of-the-art graphics, such as GTA-V. Driving in the Matrix [52] and Playing for Data [93] unlock the possibility of using a game engine for data gathering. However, Driving in the Matrix lacks finer annotation as it only extracts the stencil layer from the game, which does not differentiate between many object classes, and Playing for Data still requires a semi-manual labeling procedure. Both of these works also lack the ability of the LiDAR sensor, however, it can be circumvented by placing virtual cameras at the desired locations, as will be shown in this work. GTA-V engine was also exploited by [139], where geometrical LiDAR has been simulated in the frontal camera view. It comes, however, without intensity properties, and class labels of objects' 3D shape are approximated by bounding boxes only. GTA-V world has no concept of reflectivity of the material, and therefore, the returned LiDAR reflections are missing.

Simulation of intensity. LiDAR intensity is derived from three main components: geometric, physical, and environmental model of LiDAR [33]. The Geometric part is usually solved by basic computer vision algorithms such as ray-casting and projection of the points [147]. Physical and environmental models consist of various sensors and surrounding constants and target properties, which is usually not available in simulation [40]. These modalities are mainly reflectivity of material and beam divergence of a laser.

Work of SqueezeSegv2 [130] tried to model intensity using data; however, it resorted to using geometric information only. Another way of closing the domain differences is to substitute intensity between real and synthetic data by modeling echo pulse width (EPW) of the laser via [94] [25]. However, despite the fact that EPW is part of the LiDAR resulting intensity, the work [44] shows its lack of representativeness as a sole intensity indicator. Two objects can share the same EPW but have different reflectivity, so they do not cause the same resulting intensity. This work also shows that the implementation of EPW did not improve the model performance, as it is not descriptive enough feature for classification algorithms.

To the best of our knowledge, there is no other previous work trying to model a LiDAR intensity data-driven way. Also, none considered modeling intensity from RGB information or any other modalities besides geometric.

2.3 Method

The proposed pipeline is summarized in Figure 2.6. The LiDAR simulation employs four virtual roof-mounted cameras, which provides four temporally synchronized streams of RGBD images at a user-defined framerate. Four-tuples of depth images are converted into 360° point clouds respecting the geometry of the simulated LiDAR. This part is briefly summarized in Section 2.3.1. The resulting point clouds are deprived by random drop noise to rays following the same procedure from [130]. Finally, the LiDAR intensity is predicted by a single deep convolutional network. The intensity predicting network, as well as the learning procedure, are detailed in Section 2.3.2.

2.3.1 Geometrical simulation

The geometrical simulation of the LiDAR consists of two consecutive steps, which are briefly illustrated in Figure 2.2. First, a dense point cloud is generated from four temporally synchronized RGBD images and the known camera calibration matrices. For each pixel of the virtual RGBD

camera, we generate a corresponding 3D point $\mathbf{x}_{ego}$ in the camera coordinate frame as follows

$$\bar{\mathbf{x}}_{ego} = \begin{bmatrix} x_{ego} \\ y_{ego} \\ z_{ego} \\ 1 \end{bmatrix} = \mathbf{P}^{-1} \begin{bmatrix} x_{cam} \\ y_{cam} \\ D \\ 1 \end{bmatrix}. \tag{2.1}$$

$\bar{\mathbf{x}}_{ego}$ are homogeneous coordinates of the 3D point in a car coordinate system, $\mathbf{P} \in \mathbb{R}^{4\times4}$ is a camera projection matrix, $\{x, y\}_{cam}$ are coordinates of each pixel in an image (normalized to the range [-1, 1]) and D is the depth of each pixel. Resulting dense point cloud of $1920 \times 1200 = 2304000$ 3D points is transformed into world coordinate system using

$$\bar{\mathbf{x}}_{world} = \mathbf{W}^{-1}\bar{\mathbf{x}}_{ego}, \tag{2.2}$$

where $\bar{\mathbf{x}}_{world}$ are homogenous coordinates of the 3D point in the world coordinate system and a world matrix $\mathbf{W} \in \mathbb{R}^{4\times4}$ is a transformation matrix. Matrices $\mathbf{W}$ and $\mathbf{P}$ are obtained from the RAGE engine of GTA V.

3D points from all four cameras in the world coordinates are then concatenated into one dense point cloud, and points which are further than 130 m (operating range of commercial LiDARs) from the cameras' centers are then discarded. This results in a dense 3D point cloud with approximately 7×10^6 points. More technical details on extracting these dense point clouds can be found in [89].

Second, rays corresponding to the real LiDAR geometry (i.e., angular resolution and vertical field of view) are cast on the dense point-cloud, and the closest corresponding 3D points are extracted. Since horizontal FOV of the RGBD cameras is 91° and image width is 1920 pixels, the angular resolution of the dense point cloud is approximately 0.047°, which is approximately 3.65× finer horizontal resolution than that of the commercial LiDAR (Velodyne HDL-64E has a horizontal angular resolution of 0.1728°). The output of this procedure is a geometrically consistent point cloud.

We found that even though it is much more computationally demanding to generate this geometrically precise LiDAR representation *outside* the RAGE engine, it is also much more precise since ray-casting implemented within the RAGE engine approximates the 3D shape of the object by a bounding box as in [139].

2.3.2 Data-driven intensity simulation

Since we do not know the exact parameters of the LiDAR sensor and the reflectivity of surfaces in the simulated world, we cannot calculate intensity values directly during the simulation process. We overcome this drawback by learning to predict intensity levels from the real measurements in a data-driven way. The physical properties of "beamed" laser and received signal energy can be

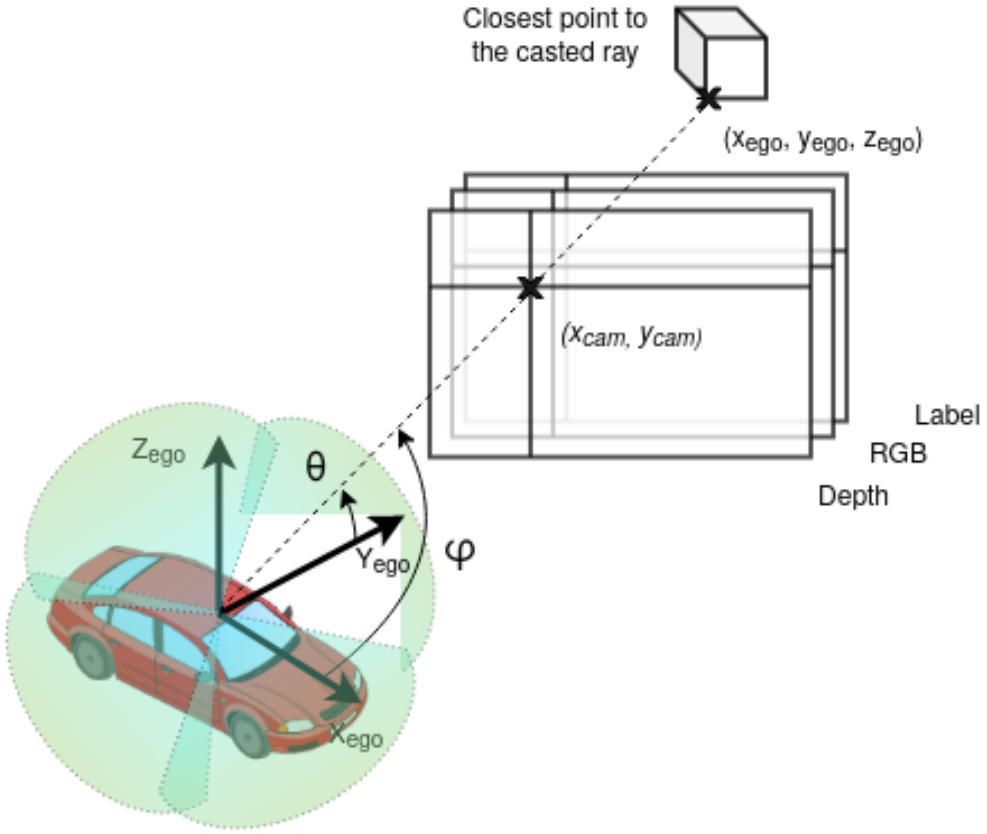


Figure 2.2: **Extracting of LiDAR point clouds** - By placing four virtual cameras on top of the car, we acquire images of a surrounding scene with depth, RGB, and label information. From these depth images, we construct dense point clouds in car-ego coordinates using a camera projection matrix (2.1). Then ray-casting procedure chooses the closest point in dense point cloud corresponding to LiDAR's angular resolution ϕ, θ and maximum range. As a result, newly created LiDAR point cloud of specific sensor parameters is obtained with all game source information (e.g. coordinates x_{ego}, y_{ego}, z_{ego}, RGB, Label) for every scan point.

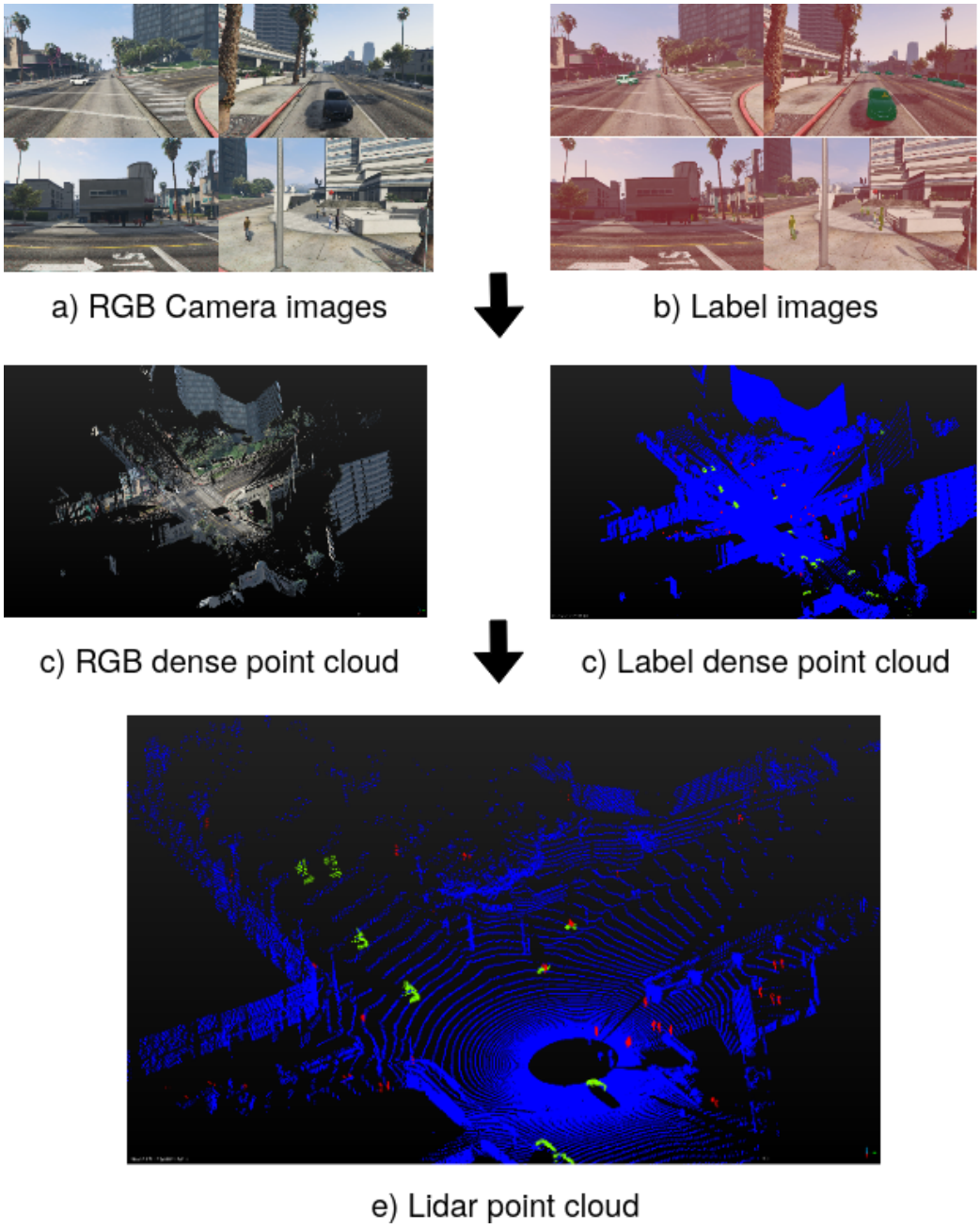


Figure 2.3: **Example of point cloud extraction** - we model Velodyne HDL LiDAR with 64 layers in 360° FOV. The procedure consists of the following steps: (a) Place four virtual cameras, which cover 360° FOV, to the position of the LiDAR. (b) Extract corresponding labels from the stencil buffer. (c+d) Reconstruct dense point clouds from all four cameras. (e) Estimate final point cloud by ray casting and dumping points exceeding maximal the range of the sensor.

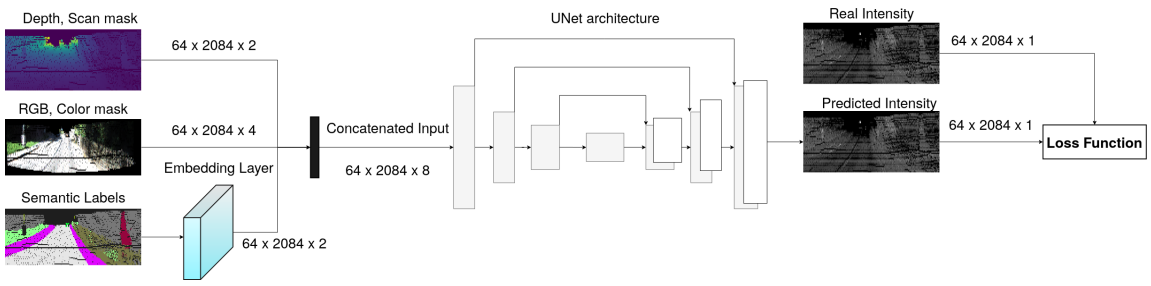


Figure 2.4: **Modeling intensity from modalities** - We use depth LiDAR measurements, calibrated images from the camera with LiDAR reading and existing labels from the SemanticKitti dataset as a source of input channels to the neural network. Inputs are sent into the neural network in form of an image-like grid with channels corresponding to the modalities, where label modality is embedded through the embedding layer resulting in a two-dimensional channel grid. We then compare intensity from real data to prediction in L2 loss function and train the model with backpropagation.

described as fixed sensor configuration and inconsistent environmental parameters using the LiDAR equation [33]. This LiDAR equation models the power of received LiDAR signal P_r, which is directly correlated to resulting intensity value I via normalization and calibration of the specific sensor. Since we model intensity using real data, this conversion will be included when modeling the same sensor. Lidar equation [33] models the power of the received signal as follows:

$$P_r = \frac{P_t D_r^2}{4\pi r^4 \beta_t^2} \eta_{sys} \eta_{atm} \sigma \tag{2.3}$$

The received signal intensity P_r can be calculated from transmitted signal power P_t, receiver aperture diameter D_r, traveled distance of laser to the target r, laser beam width β_t, sensor-specific parameter η_{sys}, atmospheric transmission factor η_{atm}, and back-scattering cross-section σ, which depends entirely on the target characteristics. Except for σ, all other parameters are defined or directly measured in constant LiDAR configuration. Signal power (P_t), laser beam width (β_t), sensor parameter η_{sys} and aperture diameter (D_r) are constants for specific LiDAR. Environment factor (η_{atm}) does not diverse along measuring sequence in the same weather conditions and range from the target is known from our geometric simulation. Then we need to consider target contribution to the intensity, which is modeled by previously mentioned cross-section σ, denoted as follows:

$$\sigma = \frac{4\pi}{\Omega} \rho_s A_s \tag{2.4}$$

where Ω is the scattering solid angle (divergence) of a laser beam, A_s is the target area, and ρ_s is the target's material spectral reflectance. The parameters depend on the geometry and reflectivity of the scanned object, i.e., the property of its material. We can leverage geometry information in our simulator, but it does not offer any information about reflectivity. We assume that material can

be estimated based on its color and possibly by information about the type of the object consisting of that material, i. e. class label. Lidar does not contain any information about RGB color. To compensate for the lack of RGB, we use a multi-sensor dataset [4], which has camera images calibrated with respect to the LiDAR. From these camera images, we project RGB channel to LiDAR scan points. This dataset also comes already annotated with class labels.

In contrast to others, we suggest exploiting all modalities available during the simulation – RGB colors, depth, *and* semantic labels. We train a deep convolutional neural network to predict the intensity from the multi-modal data.

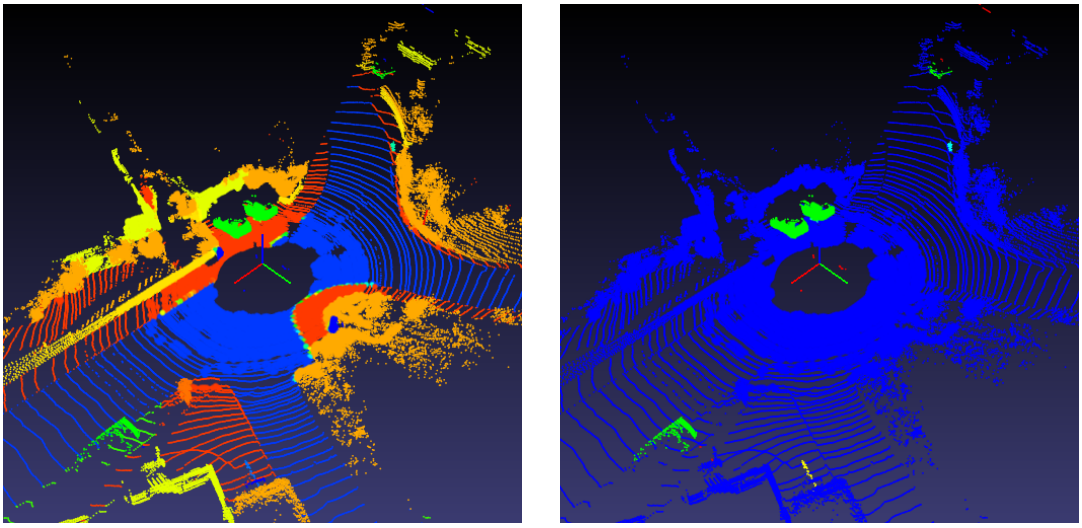
2.3.3 Learning of the intensity-predicting network

The intensity-predicting network is trained on the real data obtained from the SemanticKitti dataset [4]. This dataset contains 360° LiDAR scans, pixel-level labels, and RGB images, which are however available for the forward view only. We argue, that the training on the forward view generalizes well on other views because the testing accuracy in other views is comparable.

The learning process is outlined in Figure 2.4. To simplify the learning process, all measured modalities are projected to the cylindrical projection with a center placed at the position of the LiDAR and mapped as channels in a grid consisting of single LiDAR beams. Since there is a natural dropout in rays during LiDAR sweeps, we add a binary logic mask of successfully returned rays. Similar binary mask is also added for RGB color, which is assigned to LiDAR rays that correspond to RGB in camera projection. Consequently, the SemanticKitti dataset is converted to the set of multi-channel 2D images containing depth (D), red (R), blue (B), green (G), label (L), intensity (I), ray mask (M) and color mask (CM) values.

We work with the four following input combinations of the intensity-predicting network: D, D+L, D+RGB, D+RGB+L, which are all trained to predict the intensity channel (I) from the aforementioned inputs. The proposed network extends the existing architecture of Unet [94]. We also experiment with SqueezesegV2 architecture for comparison with [130]. Contrary to [130], we omitted XYZ channels, because these do not generalize when trained only on the forward view. In particular, D network is identical to the SqueezeSegV2 without XYZ channels, and the other networks extend the dimensionality of the input layer accordingly while keeping the other layers the same. Especially 4-class label modality (L), where every class value is transformed through embedding layer into a two-dimensional vector (i.e., it adds two additional input channels).

SemanticKitti dataset contains 19-class label descriptions. However, our GTA simulator is able to produce 4-class unique labels, see the comparison in Fig 2.5. With a more diverse object categories, label modality increase precision of intensity prediction, as can be seen in Table 2.2. That implies the potential usefulness of label feature in intensity prediction, however due to lack of categories in the current GTA simulator, we stick to the 4-class label (car, pedestrian, bicycle, background - all others).



(a) Kitti frame with all 19 labels (b) Kitti with GTA labels only

Figure 2.5: **Label diversity in real and synthetic domain** - Example of annotations in both worlds. The Real dataset has a much richer description of present objects, but thanks to the nature of the simulation world, it is feasible to introduce new categories in the engine in the future.

Predicting the intensity can be seen as a regression problem. Work [130] proposes the hybrid loss, which classifies intensity values to bins and also regresses the deviation from the classified bin, as it, according to [130], should yield better results of prediction compared to L2 loss. We experiment with both types of losses. As done in [130], our classification in hybrid loss is split into 10 bins distributed over the density of intensity value, and deviation from classified bin was predicted as another output channel from the model. Therefore in the case of hybrid loss, our prediction model has ten outputs channels for bin classification and one for regression of the deviation. The channels are then summed to the resulting intensity value, which is compared to real intensity value by a mean squared error (MSE). We trained and validated the model of intensity on the SemanticKitti dataset, compared them with training using masked L2 loss (2.5). Mask in L2 loss corresponds to the (M) input channel. Opposed to [130], masked L2 loss showed to be superior in our case, as can be seen in Table 2.2.

$$\mathcal{L} = \frac{1}{n} \sum_{i,j} (I_{i,j} - \hat{I}_{i,j})^2 \cdot m_{i,j}, \tag{2.5}$$

where i, j denotes pixel coordinates in grid-like image, I real intensity value, $\hat{I}$ predicted intensity value, m binary mask of returned scan points and n number of successfully returned rays in grid frame (i.e. the sum of m) to get mean value of loss function.

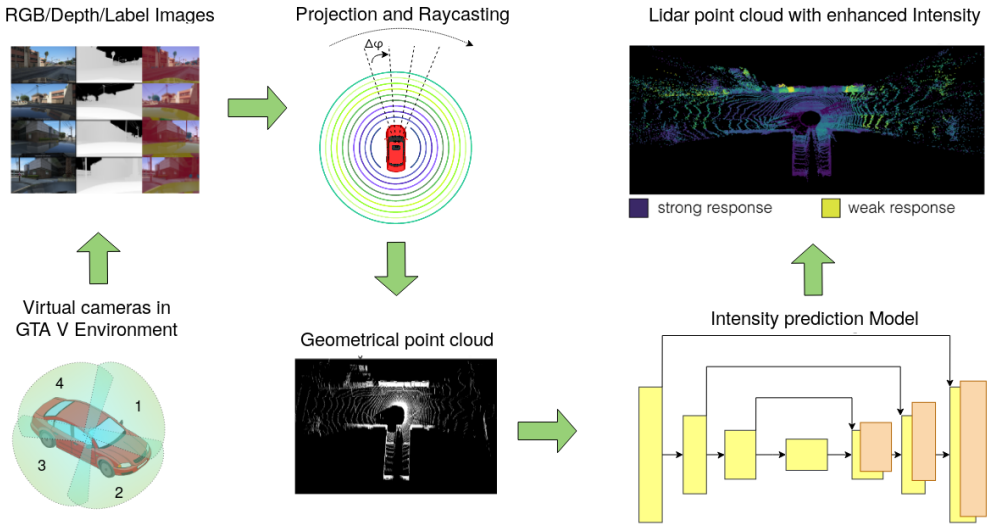


Figure 2.6: **Pipeline overview:** At first, we collect depth, RGB, and label images in 360° view by placing four cameras to the desired sensor position and from depth information create dense point cloud with RGB and label channels. With ray-casting, we choose points corresponding to LiDAR parameters and estimate intensity by the deep convolutional network from depth, RGB and label input grids. For intensity prediction we used Unet architecture[94].

2.4 Experiments

We evaluate the proposed intensity predictor in two ways: i) intensity prediction accuracy, see Section 2.4.1 which shows how close are the predicted intensities to the real ones, and ii) improving segmentation accuracy when using intensity prediction see Section 2.4.2 which demonstrates that extending the real training set by simulated point clouds improves the segmentation accuracy. The intensity prediction model was trained on 10000 LiDAR frames and tested on 2792 LiDAR frames recorded in the spatially distinct environments from the SemanticKitti dataset. In the segmentation experiment, we used a smaller portion of real dataset to study the impact of highly scalable simulated data.

2.4.1 Intensity prediction accuracy

We evaluate intensity prediction accuracy on every pixel from the LiDAR grid in terms of the mean squared error (MSE). As a prediction model, we use neural networks with encoder-decoder structure that contains skip connections in order to preserve high as well as low-level features. As long as the model is expressive enough and has a specific number of inputs (D, DL, D+RGB, D+RGB+L), it is possible to adopt different model architectures such as [125], [124] and fine-tune them for high performance.

We compare four different input combinations D, D+L, D+RGB, D+RGB+L and two different loss functions for SqueezeSegV2 - Hybrid loss from [130] and L2 loss. See Table 2.1 for details on classification values distribution and architectures. We also experiment with the aformentioned architectures of Unet neural network [94] with L2 loss and omit the hybrid loss, since we achieved consistently better performance with L2 loss in all tested modalities with SqueezeSegV2.

Table 2.1: **Distribution of hybrid loss classification bins**

Bin	1	2	3	4	5
Min value	0.0	0.079	0.163	0.225	0.264
Bin	6	7	8	9	10
Min value	0.289	0.310	0.334	0.368	0.546

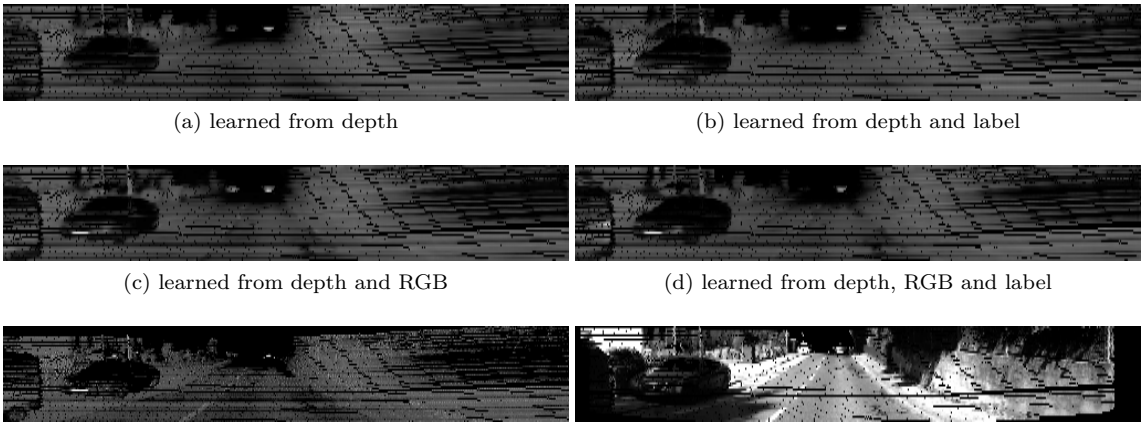
For the optimization task, we experimented with different setups and finally used Adam algorithm [55] with a learning rate 0.003 and weight decay 0.001 with both models. Training set is divided to 7500 training and 2500 validation frames.

Table 2.2: **MSE error on intensity prediction** - Comparison of different variations of modalities and loss functions for intensity prediction, all numbers in percentage.

Architecture and Loss function	D	D+L	D+RGB	D+RGB+L
SqueezeSegV2 + Hybrid loss	1.11 [130]	1.11	1.02	1.023
SqueezeSegV2 + L2 loss	0.745	0.744	0.692	0.693
Unet + L2 loss	0.644	0.671	0.623	**0.621**

Experiments reveal L2 - loss on the D+RGB+L input combination achieves the lowest MSE error on the testing data, see Table 2.2 for details. Examples of predicted intensities are provided in Figure 2.7. The intensities are projected into the camera frame for better readability. The images demonstrate that using the RGB information allows predicting stronger responses on license plates and traffic signs while using only the depth modality yields limited results. Both proposed architectures significantly outperform a simple method, such as the intensity estimated as grayscale values (see Figure 2.7(f)).

We are particularly interested in objects with high reflectivity, namely car license plates and traffic signs as they consistently show high LiDAR intensity and are valuable for scene interpretation and car detection. Intensity prediction on synthetic data can be seen in Figures 2.8, where adding RGB modality to the learning and inference showed to be superior in distinguishing these objects. With color information in model prediction, it is also possible to differentiate lane markings on the street. This can be especially valuable in segmenting other instances, that can be used for navigation in the scene. Predicting from D+RGB looks also qualitatively more realistic and closer to real LiDAR intensity, see Figure 2.7.

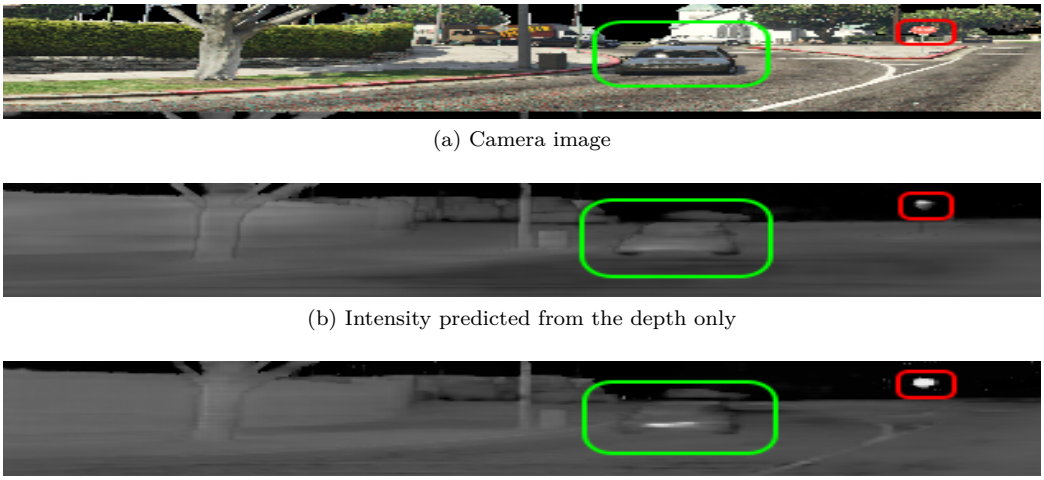


(a) learned from depth (b) learned from depth and label

(c) learned from depth and RGB (d) learned from depth, RGB and label

(e) Ground truth intensity (f) Grayscale as intensity channel

Figure 2.7: **Comparison of LiDAR intensities on the cars in the SemanticKitti dataset** - These figures represent our intensity prediction according to used modalities in the same setting. Without RGB modality (a), (b), there is no high response from the license plate, see the real intensity in (e) for comparison. Whereas (c) and (d) successfully predict the high intensity of the received beam. Substitution grayscale value for intensity failed entirely due to light conditions, as can be seen in (f).



(a) Camera image

(b) Intensity predicted from the depth only

(c) Intensity predicted from depth and RGB

Figure 2.8: **Example of GTA scene with simulated LiDAR intensity** - on the camera RGB image (a) is a car (green mark) and a traffic sign (red) mark. Predicted intensity from depth (b) and depth + RGB (c) showed different values on objects of interest, car's license plate, and traffic sign, where we expect greater values of intensity. Adding RGB modality helps to recognize licence plate (c) and more realistic values on the sign. RGB also differentiates lane marking.

(a) SemanticKitti scene - camera image (b) SemanticKitti scene - real intensity

Figure 2.9: **Comparison of real scene intensity** - Generating intensity across different domains keep systematic failures consistent, see Figure 2.8 for comparision. Traffic signs and license plates generate high signal feedback, while rest of the scene remains uniform. Therefore we can assume preservation of intensity characteristics.

2.4.2 Segmentation accuracy improvement

This experiment demonstrates that extending the costly real training data by easier accessible simulated point clouds improves the segmentation accuracy. Input to the segmentation network is a 2D image-like grid with channels containing depth, LiDAR intensity, and pixel mask, which serves as an indicator of a valid return of the ray. Some rays do not return in real LiDAR, and some exceed the maximum distance of sensor measurement in the GTA simulation.

The architecture of the segmentation network is SqueezeSeg with the CRF module. As a loss function, we used Focal loss [67] which happened to bring better results in training as opposed to the standard Cross entropy loss function. Focal loss is described in Equation (2.6). The value of parameter γ is set to 2.

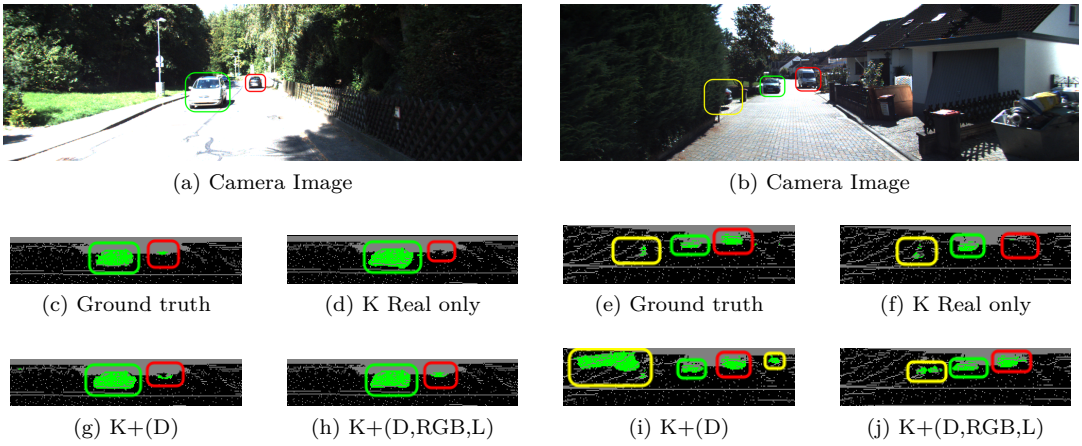
$$FL(p_t) = -(1 - p_t)^{\gamma} \log p_t \tag{2.6}$$

The output contains pixel-level semantic labels. We compare several segmentation networks trained with different combinations of training datasets. First, we evaluate networks trained on synthetic data: GTA without intensity, GTA(D) and GTA(D+RGB+L). Second, we train the prediction model on 1k real frames from SemanticKitti (K). Last, we add 40k synthetic frames to the real ones, K + GTA(D), K + GTA(D+RGB+L). We stick to the standard evaluation metric used in autonomous driving research [143] – Intersection-over-Union (2.7) – and evaluate segmentation performance on the car category.

$$IoU \quad = \quad \frac{TP}{TP + FP + FN}\,, \tag{2.7}$$

where TP denotes true positive points of a certain class, FP denotes false positives points and FN false negatives points of the class. The results are shown in Table 2.3.

Adding artificial GTA data with generated LiDAR intensity improved the performance of the segmentor, especially for vehicles in a greater distance. Adding RGB and Label modality to intensity prediction proved to be superior to the baseline – using only depth for the intensity prediction. It



(a) Camera Image (b) Camera Image

(c) Ground truth (d) K Real only (e) Ground truth (f) K Real only

(g) K+(D) (h) K+(D,RGB,L) (i) K+(D) (j) K+(D,RGB,L)

Figure 2.10: **Example of segmentation of distant and occluded car** - Adding GTA data increase segmentation performance mainly on cars in greater ranges, see the red markings on (a) and (b). There is a cropped detail on ground truth in (c). As you can see on (d), training only on the part of this dataset is not sufficient for greater range detection, there are probably not many distant cars in the dataset. However, with more training examples gathered from synthetic data, which may include similar scenes, we are able to segment it in (h),(i),(j).

yields better performance with both learning from synthetic data only and adding synthetic together with real data.

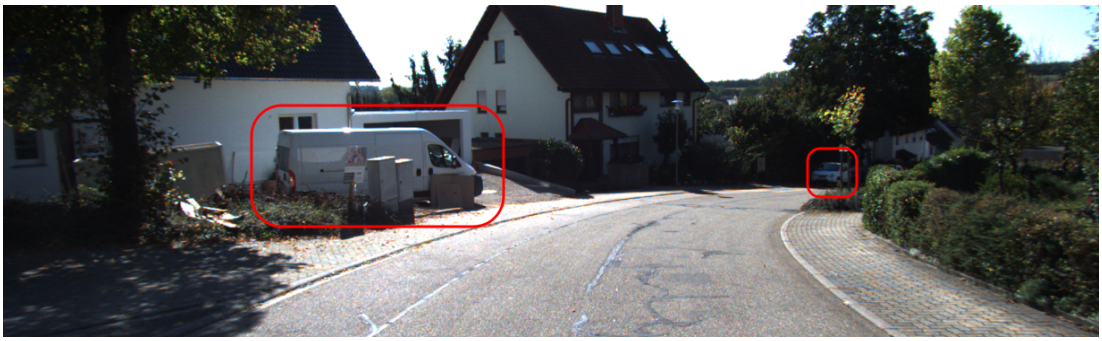
Table 2.3: **Evaluation of different modeled intensities on semantic segmentation performance on our SemanticKitti split**. All numbers in percentage.

Training set	Real data	Synthetic data	IoU
GTA w/o intensity	0	40k	23.83
GTA(D)	0	40k	31.38
GTA(D+RGB+L)	0	40k	**33.04**
K	1k	0	75.16
K + GTA(D)	1k	40k	78.79
K + GTA(D+RGB+L)	1k	40k	**79.76**

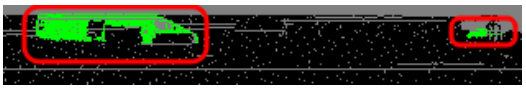
An example of a boost in segmentation can be seen in Figure 2.10, where the RGB modality improves the distinction between a vehicle that is covered by an object and the object itself. The van in Figure 2.11 is not segmented if using depth only. Red marking means improved detection with enhanced data, yellow means false detection and green shows positive detection.

There is a large disproportion in IoU performance between using GTA data only compared to using real date which implies a significant domain gap between the two worlds. Our intensity predictor improves the results, however the domain gap between simulated and real LiDAR scans

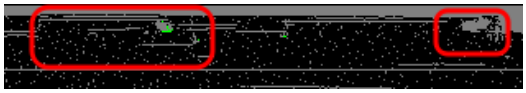
still dominates.



(a) Camera Image



(b) Ground truth



(c) Kitti Real only

(d) Kitti + GTA(D)

(e) Kitti + GTA(D+RGB+L)

Figure 2.11: **Segmentation of a occluded van** - In this situation, we see a van, which is parked behind electric panel and its wheels are covered by bush and carton (a). Detail of ground truth is shown in (b). Segmentation trained solely on real data failed to detect van in this setup (c), together with GTA intensity, learned from the depth only (d). On the other hand, the addition of RGB modality benefits in segmentation of van and also correctly detect distant car.

2.5 Discussion

Conclusion. We proposed a new way of modeling lidar intensity from scene geometry, RGB images and generated labels. It has been shown that adding proposed synthetic lidar point clouds with enhanced intensity to learning improves segmentation results on the real lidar dataset. Predicted intensity based on RGB and label had an increase in segmentation performance over depth-based intensity. We have also shown that new modalities and masked L2-loss increase the accuracy of intensity prediction. All results were evaluated on the real data only. Simulation interface and the synthetic training set consisting of panoramic RGB images and lidar point clouds have been made publicly available.

Limitations. There is still an insufficient domain adaptation between real-world and GTA simulation, mainly in geometrical properties and ray dropout. There is also limited amount of object classes in GTA simulator, limiting its effectiveness on open-set objects.

Future Work. Future work can consist of showing the results in challenging weather and visibility conditions such as fog, rain, and shallow puddles. One can also address problems of color and geometry domain shift as it should improve our intensity prediction model.

Chapter 3

Data Augmentation from Existing Labeled Data

3.1 Introduction

Data augmentation is a way to effectively decrease the need for more annotated data by enriching the training set with computed variations of the data. This type of augmentation is usually achieved with geometrical transformations, such as translation, rotation, and rescale applied to the already labeled samples [41, 134, 137, 12].

In general, 3D point cloud augmentations [41, 30] have been much less researched than image augmentation techniques [134, 12, 10, 73]. For example, the aforementioned 3D point cloud augmentations only enrich the geometrical features of the training samples but do not create new scenarios with the previously unseen layout of objects. The lack of modeling a realistic class population of the scenes is still a bottleneck of augmentation techniques. This problem can be addressed by augmentation that uses simulated virtual data and scene configurations. However, the effect of such data on training is low due to nonrealistic physical and visual features compared to real data.

We focus on improving the learning of 3D perception networks by enhancing LiDAR data in autonomous driving scenarios with data augmentation. Depth information allows for per-object manipulation when augmenting the point clouds [30]. We take advantage of the spatial position of annotated objects and place them in different scenes while handling occlusions and class-specific inhabitancy, see Figure 3.1.

Our proposed method (dubbed *Real3D-Aug* [149]) augments the road and sidewalks for class-specific insertion and exploits the bounding boxes of objects to avoid collisions. Compared to state-of-the-art LiDAR-Aug [30], which is suitable only for object detection, our bounding box generation allows augmenting the semantic segmentation datasets and simulates realistic occlusions throughout

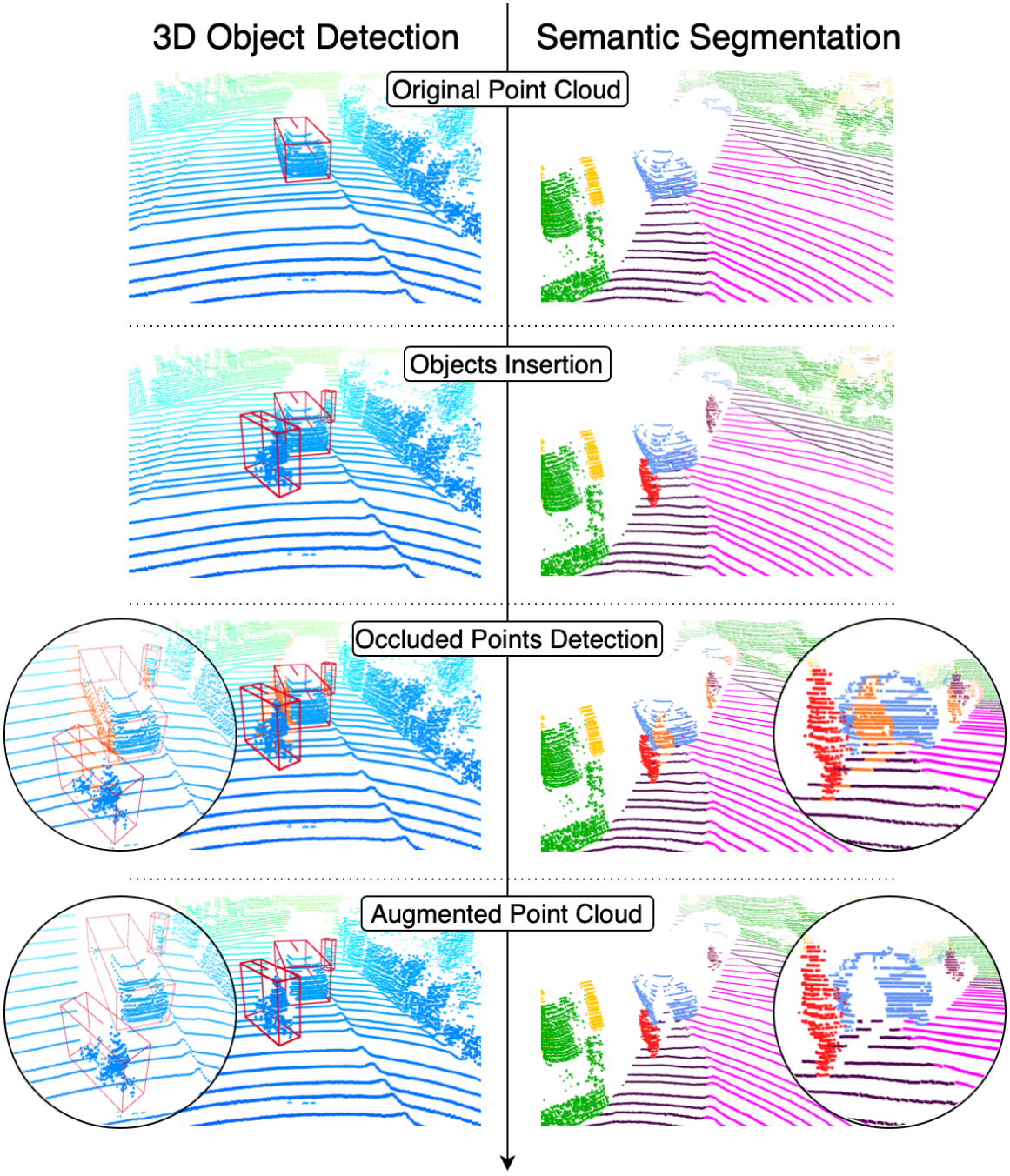


Figure 3.1: **We show examples of our augmentation method in 3D object detection and semantic segmentation**. First, we insert objects one by one and then simulate their visibility to model realistic occlusions. Note the details of the scene (circled) and the detection of occluded orange points. After removal, we see the final augmented version of the point cloud in the last row

the spherical projection. The inserted augmentations come from the same dataset and are placed at the same distance, ensuring natural reflection values and point distribution, including ray dropouts. We evaluate the proposed method on tasks of 3D object detection and semantic segmentation.

To sum up the contribution, we presented a new augmentation framework suitable for *both* 3D object detection and semantic segmentation and we proposed a novel way to *model occlusions* and physically consistent insertion of objects for augmentation. We demonstrated the usefulness of our method on autonomous driving benchmarks and show improvement, especially in rarely represented classes.

3.2 Related Work

Data Augmentation. One of the first approaches to augmenting LiDAR data was GT-Aug, which was published within the 3D detection model SECOND [135]. GT-Aug adds samples from the ground-truth database, which is precomputed before the training phase. The samples are randomly selected and inserted into the scene as is. If a collision occurs, the added object is simply removed. The visibility and occlusion handling of added scan points or the inserting strategy is not taken into account. Global data augmentations (Gl-Aug) [41] such as rotation, flip, and scale are commonly used in 3D point-cloud neural networks. These augmentations provide a different geometrical perspective, which supports the neural network with more diversity of training samples. An attempt to automate the augmentation strategy was proposed in [17], which narrows the search space based on previous training iterations. The state-of-the-art LiDAR-Aug [30] enriches the training data to improve the performance of the 3D detectors. Additional objects are rendered on the basis of CAD models. Simulations of intensity and raydrops are not discussed in the article. LiDAR-Aug [30] also simulates occlusion between additional objects and the rest of the scene, unlike GT-Aug [135]. Recent method [92], similar to our one, also focuses on inserting objects into point clouds. The main difference between the methods is in the real visibility simulation. Approach [92] upsamples the number of points in the sample, which are then projected into a range image, where visible points are selected and then sparsed. From our point of view, this approach does not consider possible raydrop on objects located between the ego and the inserted sample. It can cause parts of the inserted sample to be falsely visible because some LiDAR beams could drop out from the obstacle and create holes in the range image.

3D Perception Tasks. Learning in the LiDAR point cloud domain poses challenges, such as low point density in regions at the far end of the FOV, the unordered structure of the data, and sparsity due to the sensor resolution. Three common approaches to aggregation and learning the LiDAR features are voxel-based models [144, 135], re-projection of data into 2D structure [133, 79], and point cloud-based models [146, 107]. To show the ability to generalize, we evaluate our proposed

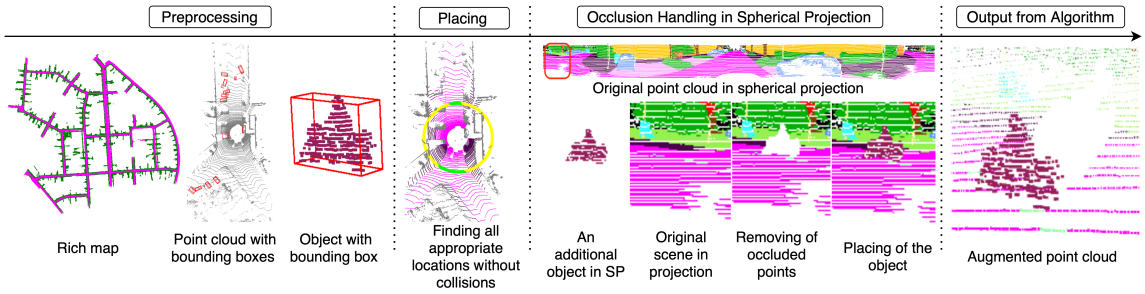


Figure 3.2: Overview of the proposed Real3DAug. We process the data in order to estimate all possible placements, all bounding boxes in the scene, and augmenting objects from different frames. The possible placement of augmenting objects is a conjunction of the same depth as the cut-out object (yellow circle) and a suitable area from the map of possible insertions (green). Occlusion handling is performed in spherical projection. The result is re-projected to the scene to the 3D augmented point cloud.

method based on different model feature extractors and on two tasks of 3D object detection and semantic segmentation.

One of the key aspects of our approach is placing the object in a realistic position by estimating the road for vehicle and cyclist insertions and the sidewalk for pedestrian insertion. Recent research has shown, that a fast, fully convolutional neural network can predict the road from the bird's eye view projection of the scene [8]. However, this method does not handle occlusions, i.e. it does not predict the road behind obstacles, e.g. vehicles. Non-learnable methods proposed in [22, 6] can separate ground from non-ground points, which can be further improved by utilizing the Jump-Convolution-Process [100]. All these methods (and other established types like RANSAC, PCA, and height thresholding) filter out all ground points regardless of class road or sidewalk. In our setup, we need to distinguish them, so we rely on the segmentation network learned from the dataset.

3.3 Method

Our augmentation method places additional objects into an already captured point cloud. The objects must be placed in adequate locations; therefore, the road and pedestrian area must be estimated (in Subsection 3.3.1). The method avoids collisions between additional objects and objects that are in the original point cloud. We analyze overlapping bounding boxes. Therefore, we need to create bounding boxes for semantic datasets that come without object boxes (in Subsection 3.3.2). More details on placing additional objects are given in Subsection 3.3.3. Lastly, the method handles realistic occlusions between objects (in Subsection 3.3.4). The overview of the proposed method is visualized in Figure 3.2.

3.3.1 Road Estimation

To place the new objects, we need to know where they realistically appear in the scene. This information may be given by prior high definition scene description (called HD map [7, 11]) if included in datasets; however, KITTI dataset [36] does not provide them. We estimate valid roads and sidewalk areas for both tasks according to the pipeline described in Figure 3.3. First, we pseudo-label 3D points by Cylinder3D [146], a state-of-the-art semantic segmentation neural network, which was pre-trained on the SemanticKITTI dataset [4]. The resulting predictions are then projected onto the 2D LiDAR (x, y) ground plane, discretized with a cell size resolution of 1×1 meter. Then we divide the space in the scene for the road (cyclist placement) and the sidewalk (pedestrian placement) as follows:

Road: To obtain a continuous *road area*, a morphological closing is used on the projection. We use a disk seed with a dimension of three.

Pedestrian area: The estimate is based on the assumption that pedestrians are supposed to walk along the road border. Cells closer than two pixels from the border of the road estimate are processed and subsequently dilated. We use a disk seed with a dimension of two.

SemanticKITTI contains poses of each point cloud in sequence. Therefore, road and sidewalk labels can be transformed into a global coordinate system and accumulated in space. The accumulated sequence of road and sidewalk labels leads to a more accurate estimation of the placement areas in the 2D LiDAR (x, y) ground plane projection. Accumulating multiple scans in one frame densifies the LiDAR point cloud and naturally reduces the need for morphological operations.

3.3.2 Creating of Bounding Boxes

For a collision-free placement of objects, the bounding boxes are required. The bounding box is parameterized by the center coordinates (x, y, z), size dimensions (l, w, h), and heading angle (yaw). For object detection in the KITTI dataset, the bounding boxes are already provided as ground-truth labels. However, the SemanticKITTI dataset contains only the semantic label of the class together with the instance of the object (each object in one frame has a different instance). We mitigate the absence of the bounding boxes by separating individual objects from the scene based on an instance and estimate bounding boxes, see Figure 3.4. In case of the absence of instance labels, we would cluster the semantic segmentation points to get the instances via density-based clustering. In the case of close-by segmentation, more than one instance can be inserted without damaging the consistency of our approach.

Modeling the bounding boxes is divided into three steps:

Wrapping: Object-labeled 3D Lidar points are projected to the ground plane. The 2D projected points are wrapped in a convex hull.

Smallest area: Assume the convex hull consists of n points. We construct $n - 1$ rectangles so that two neighboring points on the convex hull compose one side of the rectangle. The remaining

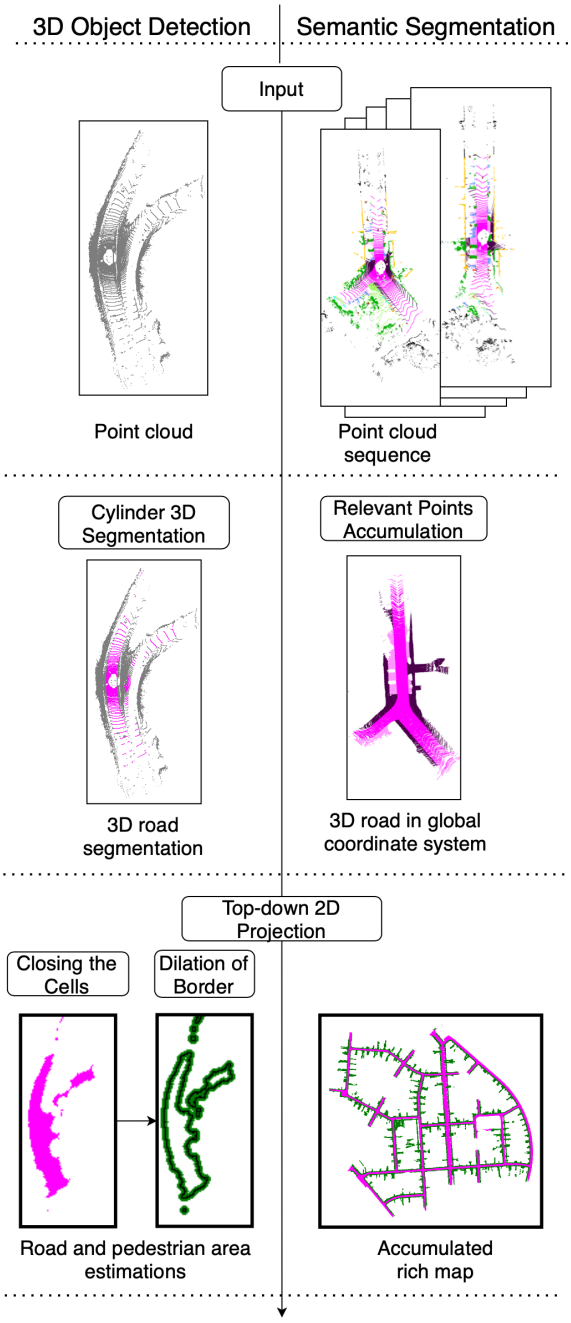


Figure 3.3: Rich map generating. Road maps are created from points' positions and labels. Semantic datasets already contain labels for each road point, in the case of the detection dataset, labels are pseudo-labeled by neural network [146]. We then project segmented points into a 2D bird's eye view and acquire road and sidewalk maps by morphological operations on the 2D projection, namely closing for the road and dilation of road boundary for the sidewalk–pedestrian area.

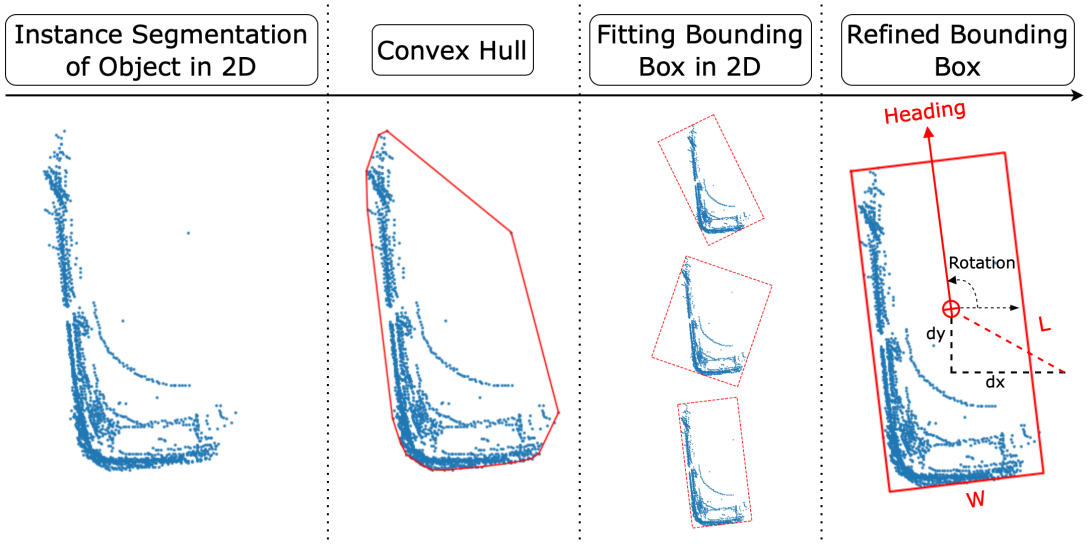


Figure 3.4: Creation of the bounding box in Bird's Eye View around the car. First, a convex hull is constructed around points; then we fit a bounding box to estimate position x, y, dimensions *length*, *width*, *height*, and orientation *yaw*. The z is estimated as if the object touches the road without intersecting it.

sides of the rectangle are added to achieve the smallest area.

Refinement: Too few points may represent some objects. They are scanned at a great distance or are significantly occluded by closer objects. Bounding boxes may also be distorted by occlusions. We analyze the heights, widths, and lengths of the bounding boxes in the KITTI dataset for classes "Car", "Pedestrian", and "Cyclists", which we use in Semantic KITTI. We obtain the distributions for each class and parameter. For each random variable, we calculate the lowest decile. The lowest decile values are the minimum threshold values of the bounding box. The maximal values of bounding boxes are set as the maximal values for the corresponding dimension that occurred in the KITTI data set.

For bicycle, motorcycle, motorcyclist, and truck objects in the SemanticKITTI dataset, we do not have corresponding statistics for bounding box dimensions since they are not present in KITTI. Therefore, the limits were hand-crafted from the first 100 generated samples from SemanticKitti. We also used the first decile, but with a 10% margin of safety.

3.3.3 Placing of Objects

Placing one or multiple objects requires knowing the bounding box dimensions and yaw angles. Only points within the bounding boxes are used to augment different frames of the dataset. For the semantic segmentation datasets (task), these points are further filtered to have an appropriate label. In the case of the object detection datasets, points that are pseudo-labeled as the road or sidewalk

classes are removed to ensure that the cutout point cloud contains only the object points.

To maintain the most realistic augmentation, our method places the object at the same distance with the same observation angle. It can be achieved by rotating its point cloud by the vertical z-axis of the frame origin. This way, realistic object point density and LiDAR intensity are maintained due to the preserved range between the sensor and the object. It also keeps the same observation angle. Then, we consider the collision-free location of the insertion:

Location: Objects must be fully inserted on the appropriate surface. We place vehicles and cyclists on the streets and pedestrians on sidewalks. Thought pedestrians can move on the streets as well, we do not observe this occurance in the evaluation datasets and therefore do not consider it during insertion. For each appropriate position, the z coordinate of the object is adjusted to ensure that the object touches the surface according to the road prediction level.

Collision avoidance: At first, the sole bounding box belonging to the object is cut from the scene and placed in the augmented frame on the road level. For the insertion of vehicles and cyclists, the bounding box must not contain any point other than road; same for pedestrians and the pedestrian area. Then, we check whether the inserted bounding box overlaps with each of the original boxes from the augmented scene and skip insertion when it does.

3.3.4 Occlusion Handling

By inserting objects into the scene, we model consistent occlusions in the point cloud from newly added points. We consider the occlusion of a newly inserted object by original points closer to the LiDAR sensor, as well as the occlusions caused by the inserted object itself.

Data projection: The occlusion handling uses a spherical projection, similarly to [133], to solve realistic visibility after the additional object is placed. The spherical projection stores the minimal distance between the sensor and the points projected to the corresponding pixel. To correct the holes in the object, the projection is morphologically closed by a rectangular seed of dimension 5×3 (5 rows and three columns). The pixels closed by the seed are assigned the depth computed from the neighboring pixels as an average of the depths in that seed area. Morphological closing is computed separately for the scene and object.

Algorithm 1 Occlusion handling

Input: Scene point-cloud $\mathcal{P}$, Scene projection, Object point-cloud, Object projection
Output: success, Scene point-cloud

```
point_counter ← 0
success ← False
for each pixel in object's spherical projection do
    if distance of object is smaller then in scene then
        Remove scene points in pixel (they are occluded)
        Add points projected to object s. p. pixel to scene
        point_counter ← point_counter + nbr of added points
if point_counter > minimal point for class then
    success ← True
return success, Scene
```

Removing occluded points: The algorithm goes through every pixel in the spherical projection. Every pixel contains information about the distance of the point. All scene points more distant than the inserted point are removed since they would be naturally occluded by the added object. as they are occluded by the placed object. Consequently, all object points, which were projected in the same pixel, are added to the scene point cloud. The algorithm also returns boolean values, which represent if the number of added sample points exceeds the threshold for a given class. We used this to prevent super hard cases, with only, e.g., three visible points from the object. A pseudocode of the algorithm is shown in Algorithm 1.

3.4 Experiments

In this section, we show the experimental evaluation of our method on KITTI and SemanticKITTI datasets with comparison to other types of data augmentation such as Global Augmentation [41], Ground Truth insertion[135] and LiDAR-Aug [30]. We experiment with two neural networks for each task.

3.4.1 Datasets and Perception Tasks

3D object detection. For Evaluation of object detection, we use the KITTI benchmark. The data set consists of 7,481 training scenes and 7,518 testing scenes with three object classes: “car”, “pedestrian”, and “cyclist”.

The test labels are not accessible, and access to the test server is limited. Therefore, we followed the methodology proposed by [30] and divided the training data set into training and validation parts, where the training set contains 3,712 and the validation 3,769 LiDAR samples [14]. To be consistent with other methods [30, 36], the evaluation was carried out on a validation set for all annotated object classes.

A metric is the standard average precision (AP) of 11 uniformly sampled recall values. We set a prediction to be true positive, when the bounding box overlaps with ground-truth in at least

50% for pedestrians and cyclists classes, and 70% for car class. We denote AP for "Pedestrian" as $\mathbf{AP}_{\text{Ped}}\mathbf{50}(\%)$, $\mathbf{AP}_{\text{Cyc}}\mathbf{50}(\%)$ for "Cyclist" and $\mathbf{AP}_{\text{Car}}\mathbf{70}(\%)$ for "Cars". The difficulties of the predictions are divided based on the sizes of the bounding box, occlusion, and truncation into "Easy", "Moderate", and "Hard", as required by the [36] benchmark.

Semantic segmentation. We use the SemanticKITTI [4] benchmark. The dataset is an extension of the original KITTI [36] benchmark with dense point-wise annotations provided for each 360° field-of-view frame. The dataset generally offers 23,201 3D scans for training and 20,351 for testing. The training data set was divided into training and validation parts with 19 annotated classes.

As for metric, we utilized standard intersection over union, IoU = TP/(TP + FP + FN). Performance is evaluated for each class, as well as the average (mIoU) for all classes.

3.4.2 3D Perception Models

We tested the augmented data on two *3D object detection* models, each based on a different type of feature extractor backbone. *PV-RCNN* [103] is a 3D object detection model that combines a 3D voxel convolutional neural network with a pointnet-based set abstraction approach [87]. The second is PointPillar [112], which encodes the point cloud in vertical pillars. The pillars are later transformed into 3D pseudo-image features.

For *segmentation task*, we use Cylinder3D [146] and SPVNAS [107] multiclass detector. Cylinder3D [146] is the top-performing architecture on the Semantic KITTI dataset with public codes. SPVNAS [107] achieves significant computation reduction due to the sparse Point-Voxel convolution and holds the fourth place on the competitive SemanticKITTI leaderboard right behind Cylinder3D [146].

Each neural network was set to the default parameters proposed by the authors of the architectures, with its performance reported on KITTI 3D benchmark and SemanticKITTI. We trained each neural network three times for object detection and five times for semantic segmentation. Average performance was considered as the final score of the method.

3.4.3 Augmentations

All augmentations were trained with the same hyperparameters to ensure a fair comparison between methods. The approach of GT-Aug was performed with information of the precomputed planes, which is an approximation of the ground from the KITTI dataset. This step should ensure that the inserted objects lie on the ground. For our proposed augmentation method, we add objects with a zero-occlusion KITTI label only (Easy). Some cases are naturally transformed into other difficulties (Moderate and Hard) by newly created occlusions.

For global augmentation of the scenes, we used uniformly distributed scaling of the scene in the range $[0.95, 1.05]$, rotation around the z-axis (vertical axis) in the range $[-45°, 45°]$ and random flipping over the x-axis from the point cloud as in [41, 30].

Table 3.1: **Semantic segmentation on SemanticKITTI.** Comparison of our method with the global augmentation baseline. Both methods are evaluated using SPVNAS [107] and Cylinder3D [146] architectures. The reported results are averaged over five runs for SPVNAS, and only one run was performed for Cylinder3D due to the large training time. The augmented categories are denoted by *. We observe a performance gain in each of them except for one: trucks. Improvement is especially notable in the motorcyclist class, which contains only a few training examples in the dataset with only global augmentations.

	mIoU	car *	bicycle *	motorcycle *	truck *	other-vehicle	person *	bicyclist *	motorcyclist *	road	parking	sidewalk	building	fence	vegetation	trunk	terrain	pole	traffic-sign
SPVNAS	60.6	95.4	29.6	58.1	**64.2**	47.6	66.2	79.1	0.0	**93.0**	**48.5**	**80.2**	**89.7**	**58.6**	87.8	67.0	73.4	63.5	47.3
SPVNAS(**ours**)	**62.7**	**95.9**	**44.1**	**73.4**	49.2	**48.4**	**70.3**	**85.4**	**12.0**	92.8	45.6	79.6	89.3	56.9	**89.1**	**67.6**	**76.7**	**63.7**	**48.8**
Cylinder3D	58.8	95.6	42.6	59.3	33.2	41.0	67.1	78.8	0.0	92.4	**42.2**	78.4	89.8	57.3	**87.4**	**67.2**	73.70	**65.0**	45.9
Cylinder3D(**ours**)	**63.0**	**96.2**	**50.4**	**71.2**	**64.2**	**50.2**	**69.7**	**88.8**	**12.6**	**93.3**	35.4	**79.8**	**90.6**	**59.8**	87.4	59.0	**73.7**	64.8	**49.2**

The maximum number of added objects in semantic segmentation was set to 10 per scene, and the object class is selected randomly (uniform distribution) each time of the insertion.

3.4.4 Evaluation of object detection and semantic segmentation

We compare our method (Real3D-Aug) with copy-and-paste augmentation (GT-Aug) [135] and with state-of-the-art LiDAR-Aug augmentation [30]. In the Real3D-Aug multiclass (mc), we added 4.7 pedestrians and 6.7 cyclists on average per scene. All methods were trained with global augmentations [41] if not stated otherwise.

In Table 3.2 we show the results of LiDAR-Aug with PV-RCNN. The numbers are taken from the original paper due to the unpublished codes and the lack of technical details about their CAD model and ray-drop characteristic. In the original article, LiDAR-Aug was trained under unknown hyperparameters and was not applied to the cyclist category. Real3D-Aug surpasses the LiDAR-Aug in the pedestrian class by a large margin despite all the difficulties. Both GT-Aug and Real3D-Aug achieve significant performance improvement. Real3D-Aug achieves a significant improvement with PV-RCNN in the pedestrian class, where we achieve 15.4%, 10.96%, and 7.87% improvement in Easy, Moderate, and Hard difficulty, and GT-Aug achieves 7.52%, 3.74%, and 0.48% improvement compared to the model without (w/o) any object augmentation. Real3D-Aug also slightly improves the performance on the car compared to no object augmentations, but Lidar-Aug and GT-Aug surpass our method on car category.

In Table 3.1 we show the results for SPVNAS [107] and Cylinder3D [146] architecture. In the semantic segmentation task, we increased the mean IoU for both networks.

We are not comparing with GT-Aug [135] and LiDAR-Aug [30] in the semantic segmentation task. The methods above were not designed for segmentation, whereas our method allows for augmenting

Table 3.2: **Object detection results with PV-RCNN.** Our method achieves the best results in the categories "pedestrian" and "easy cyclists". (mc) abbreviates multiclass

	$\mathbf{AP}_{\text{Car}}$ **70**(%)			$\mathbf{AP}_{\text{Ped}}$ **50**(%)			$\mathbf{AP}_{\text{Cyc}}$ **50**(%)		
Method	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
w/o Object-Aug	87.77	78.12	76.88	65.92	59.14	54.51	76.80	59.36	56.61
GT-Aug [135]	89.17	81.92	78.78	65.69	59.33	54.78	88.30	**72.55**	**67.79**
LiDAR-Aug [30]	**90.18**	**84.23**	**78.95**	65.05	58.90	55.52	N/A	N/A	N/A
Real3D-Aug (mc)	88.70	78.63	78.09	**73.57**	**66.55**	**62.17**	**92.69**	65.06	63.43

both tasks.

In the semantic segmentation task for SPVNAS, we achieve an increase of 2.14 in mean IoU compared to the common augmentation technique [41], see Table 3.1. We observe an increased IoU of all classes added, except for the truck category. With the Cylinder3D network, the increment can be seen in the IoU of all added classes. Our method also increases the performance on not augmented classes since we add more negative examples to other similar classes.

3.4.5 Ablation Study of Object Detection

In Tables 3.3 and 3.4 we show the influence of adding a single object to the scene in comparison to GT-Aug. Each configuration is named after the added class, and the lower index indicates the average number of objects added per scene. We can see that, in the case of PointPillar, adding only one class decreases performance in the other classes. We suspect that it is caused by similarities between classes. For example, pedestrians and bicycles are simultaneously present in the class "cyclist". Therefore, it is beneficial to add both classes simultaneously. In the case of PV-RCNN, the addition of one class improves the performance of both.

Table 3.3: **Real3D-Aug Object detection results with PointPillar architecture based on number of inserted classes.**

Augmentation	$\mathbf{AP}_{\text{Ped}}$ **50**(%)			$\mathbf{AP}_{\text{Cyc}}$ **50**(%)		
	Easy	Mod	Hard	Easy	Mod	Hard
GT-Aug	54.52	49.04	45.49	**77.64**	**61.30**	**58.15**
Real3D-Aug (Ped_1)	**55.72**	51.30	47.47	46.33	33.84	32.47
Real3D-Aug (Cyc_1)	46.87	44.17	41.77	72.65	52.71	49.04
Real3D-Aug (mc)	55.50	**52.00**	**49.03**	76.82	52.74	50.18

3.5 Discussion

Conclussion. We propose an object-centered point cloud augmentation technique for 3D detection and semantic segmentation tasks. Our method improves performance on important and rarely

Table 3.4: **Real3D-Aug object detection results with PV-RCNN based on the number of inserted classes.**

Augmentation	$\mathbf{AP}_{\text{Ped}}$ **50**(%)			$\mathbf{AP}_{\text{Cyc}}$ **50**(%)		
	Easy	Mod	Hard	Easy	Mod	Hard
GT-Aug	65.69	59.33	54.78	88.30	**72.55**	**67.79**
Real3D-Aug (Ped_1)	70.96	66.63	61.14	78.97	63.47	57.31
Real3D-Aug (Cyc_1)	65.63	59.14	57.47	82.79	63.69	62.39
Real3D-Aug (mc)	**73.57**	**66.55**	**62.17**	**92.69**	65.06	63.43

occurring classes, e.g. pedestrian, cyclist, motorcyclist, and others. Our method is self-contained and requires only 3D data. All augmentations can be preprocessed, so it does not increase the training time.

Limitations. We still rely on road segmentation model to place the augmentations. When there is a mistake in road estimation, the object can be placed into unrealistic location resulting in out-of-distribution training sample bias.

Future work. One way to further improve the method is to incorporate a more informative selection of placements based on the uncertainty of the detection model. Potential improvement can also lies in plugging the proposed framework to active learning data engine to leave it up to the detection model, where to place the augmentations.

Chapter 4

Supervision by Spatial-Temporal Aggregation with unlabeled Data

4.1 Introduction

Pseudo-labeling [60] has emerged as a versatile and powerful tool for reducing the annotations needs. A teacher model trained on a small amount of labeled data is used to annotate lots of unlabeled data automatically. The student model trains on the combination of a small labeled set and a large pseudo-labeled set. This work introduces ways to boost pseudo-labeling when dealing with temporally ordered data streams. The proposed framework is instantiated and evaluated in the context of 3D point-cloud analysis for driving applications. In contrast to unordered data, the temporal ordering of the data grants the student and teacher access to a meaningful temporal context. In particular, the teacher can also access future data in the form of privileged information [117] that is available at the time of pseudo-labeling but not at the students' inference time. Consequently, the teacher can benefit from past and future frames, thus making the most of the temporal consistency over an extended time window. We noticed that the range of this temporal window has a crucial influence on teachers' performance, as it captures different temporal contexts. The complexity of spatio-temporal events in 3D driving scenes would require the teacher to operate simultaneously at different temporal ranges. Learning a large enough teacher capable of modeling the aforementioned complexity would require many labels, contradicting the motivation of easing the annotation. We take a more practical approach where multiple complementary teachers are trained, each operating in its own temporal range with the past and future frames unannotated. These teachers use the *Concordance* to assess the confidence of the extracted pseudo-labels (PLs) and to select the most confident ones for student training eventually.

We experimentally demonstrate that Concordance-based pseudo-labeling (i) achieves competitive

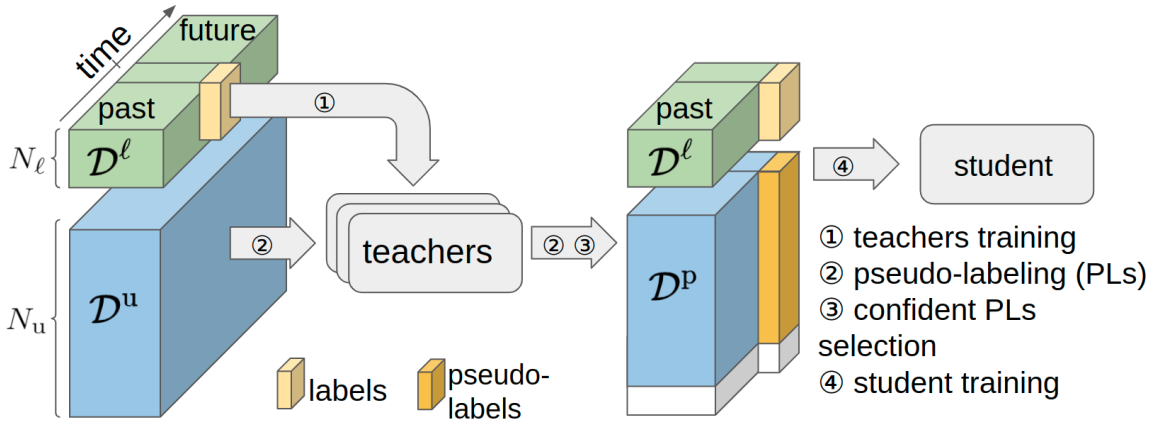


Figure 4.1: **Proposed "Concordance of teachers" for pseudo-labeling of sequences**. A set $\mathcal{D}^\ell$ of sequences with a central frame labeled and a larger set $\mathcal{D}^u$ of unannotated ones are available for training; ① Multiple offline teachers are trained with full supervision on $\mathcal{D}^\ell$, each with a different temporal range towards future and past frames; ② The teachers are run on $\mathcal{D}^u$ to produce pseudo-labels (PLs) for central frames; ③ Sequences with the most confident PLs according to Concordance of teachers are selected, forming the pseudo-labeled set $\mathcal{D}^p$. The white box depicts the discarded PLs; ④ The student is trained on $\mathcal{D}^\ell \cup \mathcal{D}^p$, to work online with past and current frames only.

performance with state-of-the-art fully supervised methods [133, 79, 140, 146] with only a fraction of labeled data, (ii) outperforms pseudo-labeling methods that do not leverage multiple teachers [50]. Our approach (Fig. 4.1) only assumes that two sets of sequences are available: The first with the central frames annotated and the second larger and devoid of annotation. Several teachers are trained on the first set to predict the label of the middle frame of an input sequence. Once trained, all the offline models run on the second dataset to pseudo-label the central frames. The most promising automatically annotated samples are weighted and added to the first set based on time-aware Concordance sample selection. The resulting large labeled set, with the future frames not available at the input, is used to train the final online model.

We put this framework to work for different spatial-temporal perception tasks on sequences of outdoor point clouds. We take advantage of the temporal ordering to provide more accurate pseudo-labels than an ordinary PL method would deliver. We demonstrate its superiority on the tasks of 3D detection and 3D semantic segmentation in driving scenes on two architectures [102, 146] and three datasets (see Fig. 4.2). Our contributions [35] to pseudo-labeling of temporal data are: 1) An effective way to aggregate time-ordered unannotated/annotated 3D scans; Leveraging such privileged information improves teacher's performance for 3D semantic segmentation and object detection tasks. 2) A novel confidence-guided criterion for better pseudo-labels selection and loss function guidance. 3) A novel weighting of pseudo-labels via the Concordance of teachers trained on different temporal ranges.

4.2 Related Work

Spatial and temporal consistency in point clouds. Unlike 2D image which stores the pixels in the grid of regular size, LiDAR point clouds are unstructured data, where we can't apply 2D convolutions directly. PointNet architecture [88] directly consumes raw point clouds to extract features with permutation invariance and global features for classification. PointNet++ [87] further improves the extraction of local features. The architecture of MeteorNet [58] works with multiple input point clouds to extract features from additionally available points with consistent temporal properties. Choy *et al.* [21] use sparse 4D CNN for spatiotemporal perception to improve robustness in detection tasks. Spatial synchronization to the reference scan and adding one additional channel of encoded time lead to further performance gain [46]. Qi *et al.* [86] use temporal information for point cloud densification based on tracking previously detected objects for automatic data annotation. Conversely to us, they use a top-performance detection model pre-trained in a fully supervised way. We focus on building a set of teacher models (Concordance) from a minimal amount of annotated data and apply distillation through pseudo-labeling.

Knowledge distillation. Training the student model is usually done by distilling knowledge in the feature or output space [43]. Liu *et al.* [69] distill an ensemble of teachers into a single student and extend the idea using different architectures suitable for different tasks as teachers. Cho and Hariharan [19] show that larger models are not necessarily better teachers, mainly due to parameter complexity mismatch. Mirzadeh *et al.* [80] propose a multistep knowledge distillation with an intermediate-sized network to bridge the gap between student and teacher complexity. The benefit of using the teacher network can also come from exploiting privileged information [13]. In our work, we adopt a teacher-student framework and distill future data in sequential frames when training the teacher models.

Semi-supervised learning. Semi-supervised learning approaches have been heavily researched for image recognition, less so for point clouds [50, 141]. Extending an image pseudo-labeling approach [60] to 3D perception [13] has recently shown that automatic labeling of LiDAR data could be leveraged to achieve considerable performance gain. An early approach [108] proposes to extract useful training examples from unlabeled data by exploiting the temporal information in LiDAR scans for classification. Enforcing consistency of model predictions across perturbed versions of unlabelled data [141] proves to be beneficial in 3D object detection. For the task of 3D semantic segmentation, learning with a point-guided contrastive loss [50] increases performance even with fewer ground-truth labels. The authors show that using pseudo-labels and confidence thresholding can help to improve feature learning. This method is compared to ours in Section 4.4. Our approach focuses on semi-supervised learning using pseudo-labeling and knowledge distillation. It is worth mentioning that a complementary line of work explores approaches such as self-supervised pre-training [132, 83] and domain adaption [48, 51] to avoid labeling too many samples.

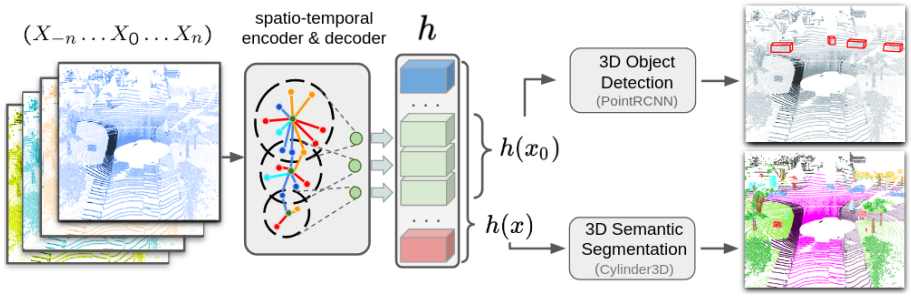


Figure 4.2: **Proposed architectures that aggregate sequence of 3D point clouds**. We build a modified version of Cylinder3D for semantic segmentation and PointRCNN for object detection that can aggregate multiple frames inside its spatio-temporal encoder. The output of the spatio-temporal encoder-decoder are extracted point features $\boldsymbol{h}(\mathbf{x})$. For object detection, Only features from the reference frame, i.e., $\boldsymbol{h}(\boldsymbol{x_0})$ for $\mathbf{x}_0 \in X_0$, are used to train PointRCNN for object detection. Here, we use the full set of features $\boldsymbol{h}(\mathbf{x})$ for semantic segmentation due to the setup of the state-of-the-art Cylinder3D architecture.

4.3 Method

4.3.1 Point-cloud notations

A point cloud $X = \{\mathbf{x}^k\}_{k=1}^K$ is a finite order-less collection of 3D points, where the number of points K is assumed constant over time to keep the notation simple. We consider symmetric time-ordered point cloud sequences of the form $X_{-n:n} = (X_{-n}, \dots, X_{-1}, X_0, X_1, \dots, X_n)$ composed of a *reference scan* X_0, preceded by n past scans and followed by n future ones. For the symmetric sequences, the reference scan (frame) is the same as the central one, see Fig. 4.1. All scans in the sequence are transformed into the coordinate system of the reference one.

4.3.2 Time-aware feature extraction

We build on a modified architectures of the Cylinder3D [146] for semantic segmentation and PointRCNN [102] for object detection. Both architectures consist of MLP modules responsible for attaching rich spatial features to individual scan points, followed by task-dependent modules. In contrast to single-frame perception, we must handle successive scans; therefore, we propose a time-aware extension of their original MLPs.

Given a sequence $X_{-n:n}$ of point clouds, the backbone architecture estimates a feature vector $\boldsymbol{h}(\mathbf{x}_0)$ for each point $\mathbf{x}_0$ in the reference scan X_0. This feature vector encompasses all contextual information from its spatio-temporal neighborhood $\mathcal{N}(\mathbf{x}_0)$ defined as an hourglass-like 3D shape centered in $\mathbf{x}_0$:

$$\mathcal{N}(\mathbf{x}_0) = \big\{\mathbf{x}_t \in X_t : \|\mathbf{x}_t - \mathbf{x}_0\| \leq r(|t|),\ t \in [\![-n, n]\!]\big\}, \tag{4.1}$$

where r is an increasing function as in [58] (Fig. 4.3). The feature of each $\mathbf{x}_0 \in X_0$ is finally defined

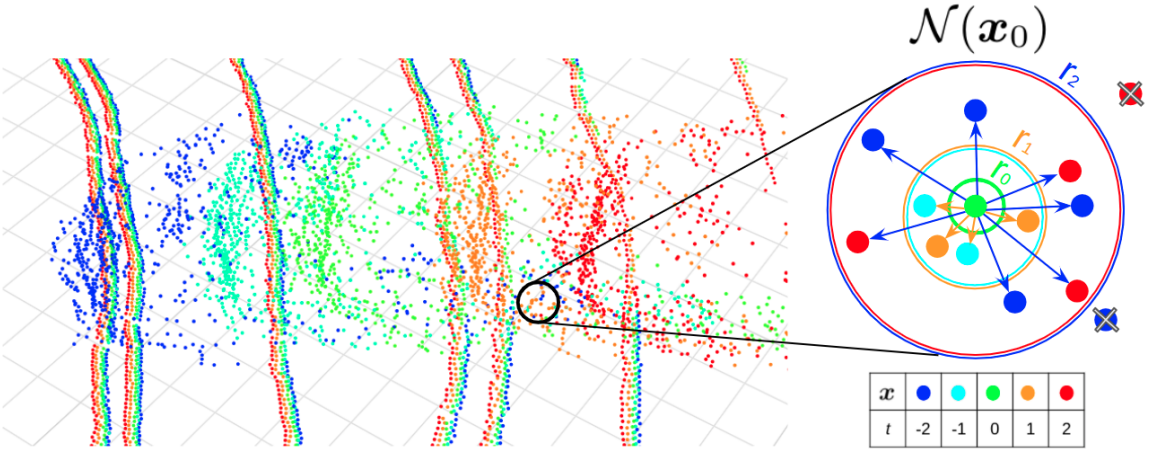


Figure 4.3: **Time-aware neighborhood of a point in the reference scan**. Points from different times are presented in different colors, where circled green point $\mathbf{x}_0$ is from the (green) reference frame X_0. Its spatio-temporal neighborhood $\mathcal{N}(\mathbf{x}_0)$ is composed of all points in scan X_t in a spatial radius $r(|t|)$, for $t = -n, \cdots, n$. Crossed points are excluded from this spatio-temporal neighborhood.

as:

$$\boldsymbol{h}(\mathbf{x}_0) = \max_{\mathbf{x}_t \in \mathcal{N}(\mathbf{x}_0)} \{\phi(\mathbf{x}_t - \mathbf{x}_0, t)\}, \tag{4.2}$$

that is, by max-pooling of time-aware pairwise features over the spatio-temporal neighborhood, where ϕ is an MLP with shared weights to be trained, and $t \in [\![-n, n]\!]$.

The construction of the spatio-temporal neighborhoods obeys the intuition that the maximum distance an object can travel between two scans increases with the object's speed and the temporal separation between the scans. Thus, the maximum spatial distance for grouping points should increase with their temporal distance.

4.3.3 Task-dependent modules

The feature extraction method provides point-wise feature vectors and their corresponding 3D positions for the points in the reference coordinate frame of X_0. The feature vectors are then fed into subsequent task-dependent modules.

3D Object Detection. We consider the 3D detection of vehicles with our multi-frame adaptation of PointRCNN [102]. Here, the bounding box labels are not associated with specific input points but with a specific 3D position. If we consider box labels from all frames, we would not know which points on input are associated with each one. Therefore, we use the bounding box labels only for the reference scan X_0 and mask the box labels from other frames. This differs from semantic segmentation, where each point has its own exclusive class probability in each frame.

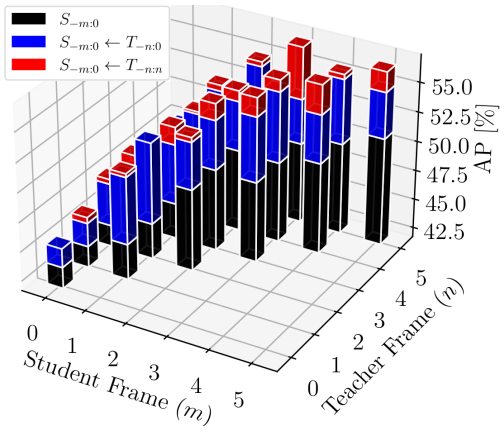


Figure 4.4: **Impact of knowledge distillation**. Performance of distilling privileged information from a single teacher into a student in the 3D object detection task.

3D Semantic Segmentation. We adopt Cylinder3D [146] and extend it into a semi-supervised approach. A trained Cylinder3D classifier semantically labels each point. Here, we use labels for all temporal input instants.

4.3.4 Training data

We consider two types of teacher models for training: (i) $T_{-n:n}$ with access to future frames, and (ii) $T_{-n:0}$, which has access only to past frames.

To understand the effect of distilling a teacher model with access to privileged information $T_{-n:n}$, we have performed an experiment where we train a student model by the pseudo-labels provided by one teacher with and without privileged information. As shown in Fig. 4.4 distilling a single teacher with privileged information in a student model $S_{-m:0} \leftarrow T_{-n:n}$ provides the best performance (red on top) over the baseline supervised student (black pillars) and over distilling a single teacher without privileged information into a student model $S_{-m:0} \leftarrow T_{-n:0}$ (blue pillars).

4.3.5 Concordance and selection of pseudo-labels

Inspired by our finding from Fig. 4.4, we want to fully exploit the information contained in the available scan sequences of length $(2n + 1)$. We propose Concordance of teachers, where we train a set of n teachers with a varying span of the temporal context. It means that teacher $T_{-1:1}$ is trained on the subsequences $X_{-1:1}$, teacher $T_{-2:2}$ on $X_{-2:2}$ and so on, for the sake of simplicity, we further drop the distinction between these sequences and assume that given a sequence $X \in \mathcal{D}^{\mathrm{u}}$ any network will crop the temporal range appropriately. Given the Concordance of teachers

$$\mathcal{T}^{\{1,\dots,n\}} = \{T_{-1,1}, T_{-2,2}, \dots, T_{-n,n}\}. \tag{4.3}$$

We independently process every unlabeled sequence $X \in \mathcal{D}^{\mathrm{u}}$ by all trained teachers. Since the nature of outputs is slightly different for the semantic segmentation (point-wise class probabilities) and for the object detection (3D bounding boxes with class probabilities), we split the Concordance description at this point to avoid any misinterpretation.

Concordance for semantic segmentation. We estimate the pseudo-label k^* and its confidence c for each output point. Given the point, each teacher $T \in \mathcal{T}^{\{1,\dots,n\}}$ provides a vector of class probabilities $\hat{\mathbf{y}}^T$. We then estimate the teacher-wise pseudo-labels $k^*(T)$

$$k^*(T) = \operatorname*{argmax}_{k \in K} \hat{\mathbf{y}}_k^T . \tag{4.4}$$

We pay special attention to the teacher with the strongest opinion, the one with the highest output value $\hat{\mathbf{y}}_k^T$. In particular, we denote this teacher as T^*, its pseudo-label k^* and the score of this pseudo-label as $y^* = \hat{\mathbf{y}}_{k^*}^{T^*}$.

We determine the pseudo-label of the given output point as k^*. The confidence of k^* is defined as the weighted sum of two criteria: (i) the score y^* of the pseudo-label and (ii) the number of other teachers T that predict the same class, *i.e.*, for which $k^* = k^*(T)$. The trade-off between these two criteria is determined by a non-negative weight λ as follows:

$$\hat{c} = y^* + \lambda \sum_{T \in \mathcal{T} \setminus T^*} \mathbb{1}[\![k^* = k^*(T)]\!] . \tag{4.5}$$

To preserve compatibility with training loss Eq. (4.6), this confidence is clipped: $c = \min(1, \hat{c})$.

Concordance for object detection. Each teacher provides a different set of 3D bounding boxes (bbs) and class probabilities. To calculate final pseudo-label confidence, we need to extract bbs that correspond to a single physical object. We greedily search for clusters of bbs with mutual intersection-over-union above a user-defined threshold. The algorithm starts by building the first cluster from the strongest bb. Once there are no more bbs with a sufficient IoU with the strongest bb, we stop building the cluster, suppress all associated bbs, and continue building a following cluster from the remaining bbs. The class probabilities $\mathbf{y}^T$ corresponding to every single cluster are then used directly to compute the pseudo-label and its confidence by the same procedure as for the semantic segmentation. Following the standard practice in pseudo-labeling (*e.g.*, [72] in object detection or [65] in semantic segmentation), the final selection of pseudo-labels is obtained by thresholding the confidence. Individual pseudo-labels with confidence below the chosen threshold are masked out of the loss function and treated as a *Don't Care* class. The final selection of pseudo-labels and associated data form the training set $\mathcal{D}^{\mathrm{p}}$.

Training the student. The student model only performs inference online and, therefore, is learned only with past input sequences. Following pseudo-labeling through the teacher-student framework [60, 104], we train the student model on both the human-labeled set and the pseudo-labeled one.

Table 4.1: **LiDAR semantic segmentation performance on SemanticKITTI *single-scan* test set.** Our method, $S_{0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$ ('Ours (20%)') utilizes only 20% of the labeled data, the remaining 80% of training data being automatically annotated, to achieve a performance comparable with state-of-the-art methods. 'Cylinder3D (20%)' denotes Cylinder3D [146] trained, like our method, with only 20% of labeled data; all other results are obtained from the literature, where full (100%) labeled data is used. Performance in IoU percentages, per class and averaged, the higher, the better. Green and red indicate fully-supervised methods that have performance below and above the performance of the proposed method, respectively. '*' means that techniques such as fine-tuning and test-time augmentation (TTA) with flip and rotation are applied.

	mIoU	car	bicycle	motorcycle	truck	o-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	o-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
RangeNet++ [79]	52.2	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
PolarNet [140]	54.3	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	90.0	61.3	84.0	65.5	67.8	51.8	57.5
SqueezeSegV3 [133]	55.9	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9
KPConv [109]	58.8	96.0	32.0	42.5	33.4	44.3	61.5	61.6	11.8	88.8	61.3	72.7	31.6	95.0	64.2	84.8	69.2	69.1	56.4	47.4
Cylinder3D [146]	61.8	96.1	54.2	47.6	38.6	45.0	65.1	63.5	13.6	91.2	62.2	75.2	18.7	89.6	61.6	85.4	69.7	69.3	62.6	64.7
(AF)2-S3Net [16]*	69.7	94.5	65.4	86.8	39.2	41.1	80.7	80.4	74.3	91.3	68.8	72.5	53.5	87.9	63.2	70.2	68.5	53.7	61.5	71.0
PVD [45]*	71.2	97.0	67.9	69.3	53.5	60.2	75.1	73.5	50.5	91.8	70.9	77.5	41.0	92.4	69.4	86.5	73.8	71.9	64.9	65.8
Cylinder3D (20%)	51.9	92.1	31.7	28.5	25.1	22.1	49.6	32.4	26.4	86.9	47.2	67.5	12.6	88.7	55.7	83.2	64.4	64.8	53.4	53.9
Ours (20%)	58.9	92.9	46.4	36.6	35.1	27.3	62.4	54.0	24.0	90.0	60.8	72.1	22.2	92.0	65.6	84.6	70.3	63.7	59.3	60.2

The loss function for training the student is summed as follows:

$$\mathcal{L} = \frac{1}{M} \sum_{(\mathbf{y},c)\in\mathcal{D}^{\mathrm{p}}\cup\mathcal{D}^{\ell}} c\, \mathcal{L}_{\mathrm{task}}(\hat{\mathbf{y}}, \mathbf{y}), \tag{4.6}$$

where $\mathbf{y}$ is one hot label encoding vector, $\hat{\mathbf{y}}$ are model predictions, and c is the corresponding confidence from our Concordance selection. Samples collected from the original human-labeled dataset have $c = 1$. The original loss function of the task-dependent module is denoted $\mathcal{L}_{\mathrm{task}}$. All data are sampled from the combined datasets $\mathcal{D}^{\mathrm{p}}$ and $\mathcal{D}^{\ell}$, and M is the number of samples in the union.

4.4 Experiments

4.4.1 Datasets

We evaluate our approach on the Argoverse dataset [11] for 3D *vehicle* object detection, and SemanticKITTI [4] and nuScenes [7] for 3D semantic segmentation. **Argoverse** provides a large collection of LiDAR sequences with 3D bounding-box labels, from which we utilize the first 10% of human-labeled sequences ($\mathcal{D}^{\ell}$) and 90% being gathered without annotation in $\mathcal{D}^{\mathrm{u}}$ for pseudo-labeling. **SemanticKITTI** provides a large-scale set of driving-scene sequences for 3D semantic segmentation. It consists of 22 sequences that split from 00 to 10 for training (08 reserved for validation) and 11

Table 4.2: **LiDAR semantic segmentation performance on nuScenes valid set**. 'Ours (20%)' utilizes only 20% of the GT annotation, the remaining 80% of training data being automatically annotated, to achieve a performance comparable with state-of-the-art methods. All other results are obtained from the literature, where full (100%) GT annotation is used.

	mIoU	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	trafficcone	trailer	truck	driveable	other	sidewalk	terrain	manmade	vegetation
(AF)2-S3Net [16]	62.2	60.3	12.6	82.3	80.0	20.1	62.0	59.0	49.0	42.2	67.4	94.2	68.0	64.1	68.6	82.9	82.4
RangeNet++ [79]	65.5	66.0	21.3	77.2	80.9	30.2	66.8	69.6	52.1	54.2	72.3	94.1	66.6	63.5	70.1	83.1	79.8
PolarNet [140]	71.0	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7
PVD [45]	76.0	76.2	40.0	90.2	94.0	50.9	77.4	78.8	64.7	62.0	84.1	96.6	71.4	76.4	76.3	90.3	86.9
CylAsy3D [145]	76.1	76.4	40.3	91.2	93.8	51.3	78.0	78.9	64.9	62.1	84.4	96.8	71.6	76.4	75.4	90.5	87.4
Cylinder3D (20%)	62.0	66.4	13.8	74.7	82.8	16.1	52.1	63.3	48.4	39.3	71.8	95.0	61.6	68.1	71.1	85.0	82.6
Ours (20%)	71.8	73.8	29.3	85.0	90.4	41.6	73.6	74.1	61.1	54.9	78.0	96.1	70.8	73.3	73.9	87.1	85.4

Table 4.3: **LiDAR semantic segmentation performance on SemanticKITTI *multi-scan* test set**. Moving object classes are prefixed with 'mv'; N.B., our model fails to segment 'moving-truck' and 'moving-other' objects as there are no examples of such categories in the 20% labeled split. This is the limitation of the data split.

	mIoU	car	bicycle	motorcycle	truck	o-vehicle	person	road	parking	sidewalk	o-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	mv-car	mv-truck	mv-other	mv-person	mv-biclist	mv-motor
DarkNet53 [4]	41.6	84.1	30.4	32.9	20.0	20.7	7.5	91.6	64.9	75.3	27.5	85.2	56.5	78.4	50.7	64.8	38.1	53.3	61.5	14.1	15.2	0.2	28.9	37.8
SqueezesSegv3 [101]	43.1	88.5	24.0	26.2	29.2	22.7	6.3	90.1	57.6	73.9	27.1	91.2	66.8	84.0	66.0	65.7	50.8	48.7	53.2	41.2	26.2	36.2	2.3	0.1
KPConv [109]	51.2	93.7	44.9	47.2	42.5	38.6	21.6	86.5	58.4	70.5	26.7	90.8	64.5	84.6	70.3	66.0	57.0	53.9	69.4	0.5	0.5	67.5	67.4	47.2
CylAsy3D [145]	51.5	93.8	67.6	63.3	41.2	37.6	12.9	90.4	66.3	74.9	32.1	92.4	65.8	85.4	72.8	68.1	62.6	61.3	68.1	0.0	0.1	63.1	60.0	0.4
Cylinder3D (20%)	42.1	89.4	35.2	22.9	16.3	15.9	11.6	88.1	53.9	69.2	12.6	88.6	56.8	83.2	65.7	61.3	53.2	59.2	65.8	0.0	0.0	43.3	47.9	12.8
Ours (20%)	47.2	93.0	45.3	35.7	27.4	19.4	14.4	90.5	61.4	75.0	15.6	91.3	62.1	83.3	69.3	64.0	59.7	63.6	77.4	0.0	0.0	64.0	57.5	9.5

to 21 for testing. The dataset has two challenges, i.e., *single-scan* with 19 class categories and *multi-scan* with 25 class categories, including 19 from single-scan and six moving-object categories. **nuScenes** contains 1000 scenes with a great diversity of urban traffic and weather conditions. It officially divides the data into 700/150/150 scenes for train/val/test. For our experiment, we cut each sequence into two parts, the first 20% for the human-labeled set $\mathcal{D}^{\ell}$ and the latter 80% for the unlabeled set $\mathcal{D}^{\mathrm{u}}$.

4.4.2 3D Multi-Class Semantic Segmentation

We train a student model on pseudo-labels generated by the concordance of teachers. Here, we utilize Cylinder3D and the output class probabilities for all individual scan points are treated with our confidence-guided criterion (Eqs. 4.4, 4.5). Training is done by optimizing the cross-entropy loss and the Lovasz-softmax loss [5] weighted by our confidence-guided criterion, as in Eq. 4.6. The standard mean Intersection over Union (mIoU) metric is used for evaluation.

Table 4.4: **Semi-supervised learning on SemanticKITTI validation set**. Performance in mIoU (%). 'Guided-Point-SSL' denotes [50] semi-supervised models; '$S_{0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$' denotes our approach with distillation from the concordance of teachers.

	Labeled data		
method	20%	30%	40%
Guided Point SSL [50]	58.8	59.4	59.9
$S_{0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$ (Ours)	**59.9**	**60.7**	**62.2**

Single-scan semantic segmentation. In this experiment, we compare the results of our method with fully-supervised state-of-the-art LiDAR segmentation on SemanticKITTI single-scan test set and nuScenes validation set. Further, we present some qualitative results in Fig. 4.5, which show that our model helps to improve the segmentation quality as compared to its supervised-only counterparts. As shown in Table 4.1, our method $S_{0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$ trained with only 20% of ground truth (GT) and 80% of pseudo-labeled outperforms all methods based on 3D-to-2D projection with fully-annotated training data [133, 79, 140]. Moreover, our method shows comparable results to voxel partition and 3D convolution-based methods, including fully-supervised Cylinder3D. We made a similar comparison with the fully-supervised state-of-the-art methods on nuScenes validation split. Our method $S_{0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$ trained with only 20% of GT and 80% of pseudo-labels outperforms some of the fully-supervised models, see Table 4.2.

Multi-scan semantic segmentation. Compared to the single-scan set-up, the multi-scan segmentation in SemanticKITTI has six more categories accounting for moving objects (*car*, *truck*, *other-vehicle*, *person*, *bicyclist* and *motorcyclist*). In this experiment, all methods utilize multiple input point clouds. In Table 4.3, we show that our method, with only 20% of human-labeled training data, outperforms methods that use full manual annotations, namely, DarkNet53 [4] and SqueezesSegv3 [101], and is on par with KPConv [109] and CylAsy3D [145]. Our method outperforms all others on the *moving-car* and *traffic-sign* categories.

Comparison to state-of-the-art semi-supervised method. To further assess the merit of our approach, we compare it to the most recent semi-supervised segmentation work, Guided-Point-SSL [50] on the SemanticKITTI validation set. As shown in Table 4.4, our method learned from Concordance of teachers, $S_{0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$, outperforms Guided-Point-SSL with 20%, 30% and 40% labeled data by 1.1, 1.3 and 2.3 mIoU points respectively.

4.4.3 3D Object Detection

The models are trained in the same way as described in PointRCNN [102], except for the confidence-guided criterion and the usage of multiple frames at the input. In Table 4.5, we compare the proposed method to the baseline $S_{\text{-}5:0}$ [102] and to the Mean-Teacher framework (MT) $S_{\text{-}5:0} \leftarrow T_{\text{-}5:0}$ [141].

Table 4.5: **Results of 3D object detection**. Detection performance (AP percentage) of proposed students trained with 10% human-labeled training data and of oracle model trained with full (100%) labeled data on Argoverse validation set. '*' indicates reimplementation into our multi-frame PointRCCN architecture.

$S_{\text{-}5:0}$ [102]*	$S_{\text{-}5:0} \leftarrow T_{\text{-}5:0}$ [141]*	$S_{\text{-}5:0} \leftarrow \mathcal{T}^{\{3,4,5\}}$ (Ours)	Oracle
51.1	56.9	58.3	62.6

Table 4.6: **Effect of temporal diversity of teachers.** The student trained using concordance of teachers from different temporal ranges outperforms the one trained from the same temporal range but with different initialization in both (a) 3D object detection (Argoverse validation set) and (b) semantic segmentation (SemanticKITTI validation set).

Object Detection	AP
$S_{\text{-}3:0} \leftarrow \mathcal{E}^{\{3,3\}}$	54.3
$S_{\text{-}3:0} \leftarrow \mathcal{T}^{\{2,3\}}$	**55.6**
$S_{\text{-}4:0} \leftarrow \mathcal{E}^{\{4,4\}}$	57.2
$S_{\text{-}4:0} \leftarrow \mathcal{T}^{\{3,4\}}$	**57.7**
$S_{\text{-}5:0} \leftarrow \mathcal{E}^{\{5,5\}}$	58.0
$S_{\text{-}5:0} \leftarrow \mathcal{T}^{\{4,5\}}$	**58.2**

Semantic Segmentation	mIoU
$S_{\text{-}1:0} \leftarrow \mathcal{E}^{\{1,1,1\}}$	59.5
$S_{\text{-}1:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$	**60.6**
$S_{\text{-}2:0} \leftarrow \mathcal{E}^{\{2,2,2\}}$	59.9
$S_{\text{-}2:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$	**60.6**
$S_{\text{-}3:0} \leftarrow \mathcal{E}^{\{3,3,3\}}$	60.0
$S_{\text{-}3:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$	**60.9**

To reach a fair comparison, we have re-implemented the MT [141] into our architecture and used the classification and regression branches of the original PointRCNN loss function instead of the MT [141] consistency loss. The proposed method $S_{\text{-}5:0} \leftarrow \mathcal{T}^{\{3,4,5\}}$ learned from the concordance of teachers achieves 58.3 AP when trained with 10% human-labeled training data, outperforming the baseline [102] by 7.2 AP and the MT [141] by 1.4 AP. Moreover, it closes 62.6% of the gap between the baseline [102] and the 'Oracle' (the model $S_{\text{-}5:0}$ trained with 100% GT labeled training data).

4.4.4 Implementation Details

We trained models with an ADAM optimizer with a learning rate of 0.001 for 40 epochs on semantic segmentation and 200 and 50 epochs of the RPN and RCNN branches of the object detection model, respectively, using 4 Nvidia A100 GPUs running for 3 days of training for each task. In the object detection task, we subsample each point cloud to 16,384 points from each frame as inputs to the model. We have used three set-abstraction layers ϕ with sizes 4096, 1024, and 128 for our multi-scale time-aware grouping to subsample points into groups. We have used $\lambda = 0.1$ in our experiments.

4.5 Ablation Studies

Ablation on temporal diversity of teachers. We show that the temporal diversity among

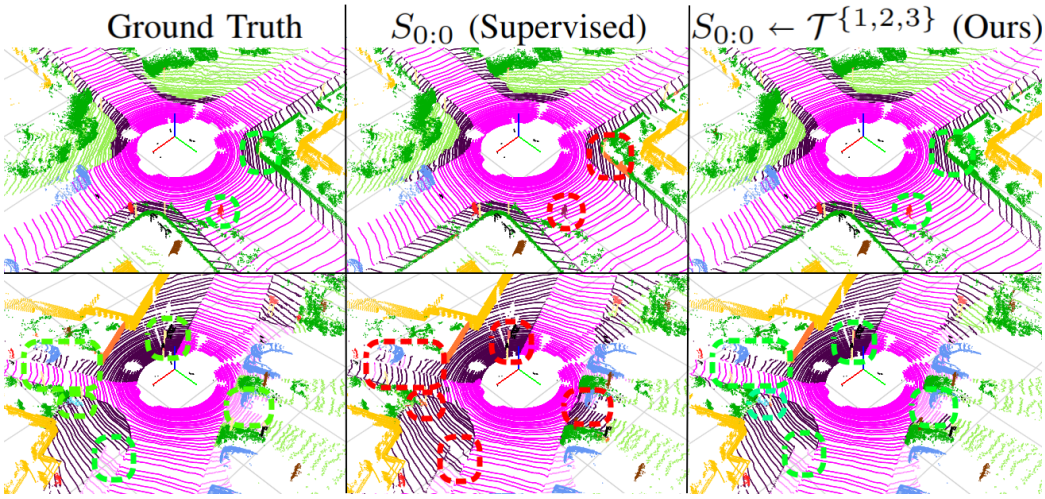


Figure 4.5: A scene from SemanticKITTI with its GT semantic segmentation (1st column) showing that our method $S_{\text{-}\,0:0} \leftarrow \mathcal{T}^{\{1,2,3\}}$ (3rd column) can segment person ■, side-walks ■, bicycle ■ and drivable areas ■ better than the supervised baseline based on Cylinder3D (2nd column). Correct and incorrect segmentations are indicated by green and red circles, respectively. Both methods utilize 20% of human-labeled data.

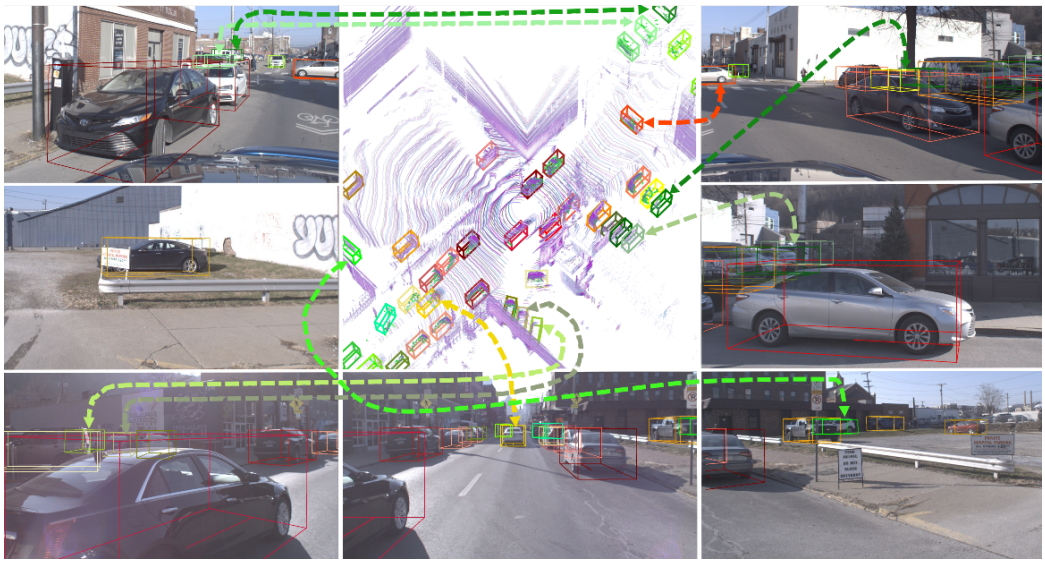


Figure 4.6: A scene from Argoverse showing that student $S_{\text{-}\,5:0} \leftarrow \mathcal{T}^{\{3,4,5\}}$ can be robust to severe occlusions. Some of the correctly detected boxes do not contain GT vehicle points since they correspond to vehicles occluded in the current frame and partially occluded in the past frames, as can be observed in the ring of camera images.

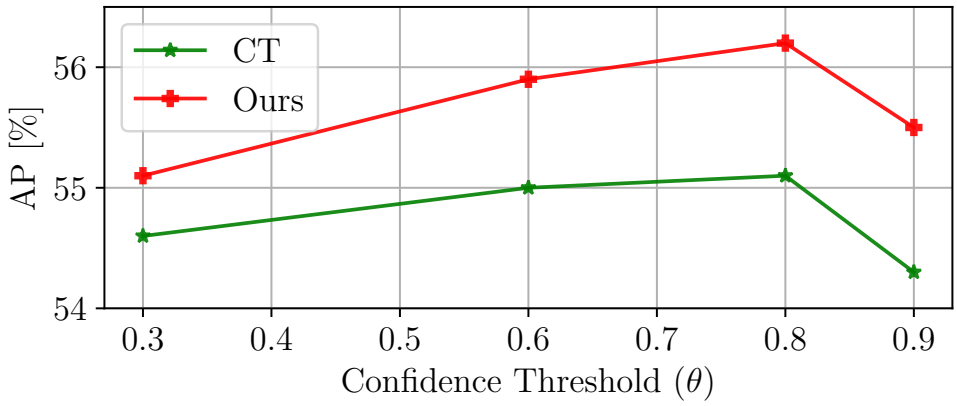


Figure 4.7: **Ablation on PL's selection strategy in Argoverse object detection**. Performance as a function of threshold θ. 'Ours' is the proposed confidence-guided criterion and 'CT' the standard confidence-based threshold [72, 142, 122].

teachers over-performs teachers with the same temporal range but with various training initializations. Following the Concordance notation, a set of teachers with the same temporal range is denoted $\mathcal{E}^{\{n,\dots,n\}} = \{T^1_{-n,n}, \dots, T^K_{-n,n}\}$, where each randomly initialized teacher $T^k_{-n,n}$ is operating in *the same* temporal range as others. As shown in Table 4.6, the student trained by teachers from different temporal ranges outperforms one trained by teachers on the same temporal range, in both 3D object detection and 3D semantic segmentation.

Selection of pseudo-labels. This ablation study demonstrates the benefits of the proposed confidence-guided criterion. The standard baselines here are tuning a single confidence-based threshold (CT) for all pseudo-labels [72, 142, 122]. Models with our proposed selection criterion outperform the CT across multiple confidence thresholds, see Fig. 4.7.

Effect of labeled and pseudo-labeled dataset ratio. We performed an ablation study to understand the effect of labeled vs. pseudo-labeled training data. We have used the same architecture setup, only varying the amount of labeled and pseudo-labeled data, $|\mathcal{D}^\ell| = 10$, 20, 30, 40, 60, and 100% of training data. As shown in Fig. 4.8, the gain increases significantly by ~ 10 mIoU when the model uses $|\mathcal{D}^\ell| = 20\%$ of training data compared to $|\mathcal{D}^\ell| = 10\%$. However, the relative gain decreases as the number of labeled data increases to 30 and 40%. This trend shows that the size ratio between $\mathcal{D}^\ell$ and $\mathcal{D}^{\text{p}}$ should be carefully set to achieve adequate performance with the smallest amount of labeled data possible. Moreover, the proposed method, when it uses $|\mathcal{D}^\ell| = 60\%$ of training data (and $|\mathcal{D}^{\text{p}}| = 40\%$), reaches the performance of the fully-supervised baseline model $S_{0:0}$ which is Cylinder3D [146] trained with 100% of labeled training data; it can be observed by comparing the top-right ending points on the blue and green lines.

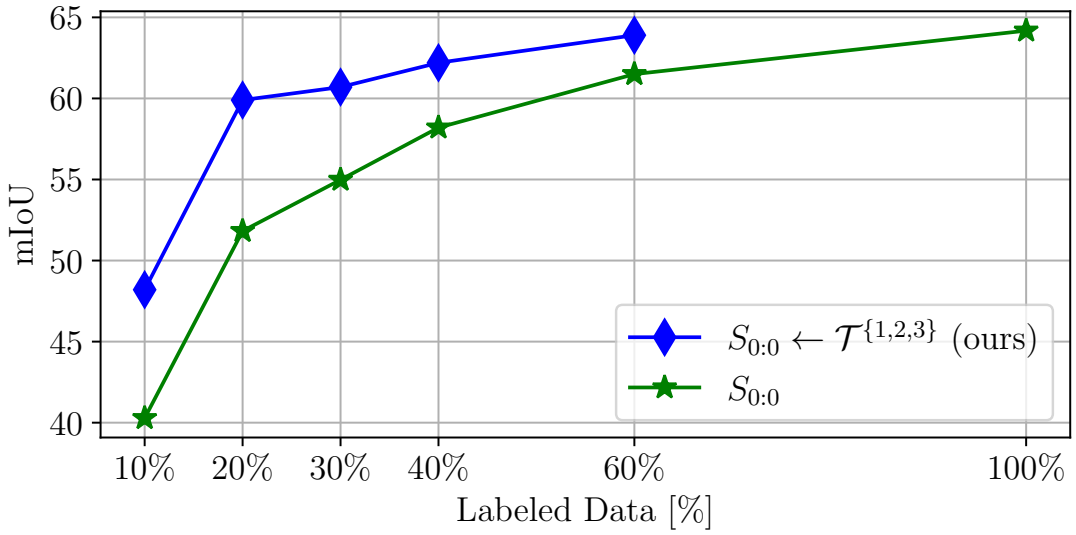


Figure 4.8: **Impact of labeling proporiton**. Semantic segmentation performance on SemanticKITTI validation set as a function of labeled data proportion size.

Table 4.7: **Comparision to other teacher-student methods**. All methods use $S_{0:0}$, are trained with 20% of labeled data and are evaluated on the SemanticKITTI validation set.

methods	mIoU [%]
Cylinder3D + KD [113]	54.8
Cylinder3D + EN [9]	56.0
Cylinder3D + Ours	59.9

Comparision to other teacher-student frameworks. We compare in Table 4.7 our method with other teacher-student approaches such as knowledge distillation (KD) [113], and Ensemble (EN) [9]. We report a comparison using the Cylinder3D $S_{0:0}$ model trained with the methods above using the hyperparameters from Section 4.4.4. All methods are trained with 20% labeled data. As shown in Table 4.7, the proposed method significantly outperforms KD [113] and EN [9] baseline teacher-student methods.

Distant and close objects. We investigate how multi-scan segmentation is affected by the distance of the points to the ego-vehicle and the number of labeled and pseudo-labeled points for each object class. We compare our method to the baseline [146] on the SemanticKITTI validation set to show the relative gain. Both models are trained with 20% labeled training data. As shown in Fig. 4.9, a notable performance boost is observed for rarely-appearing classes, especially within 30 meters distance from the ego-vehicle.

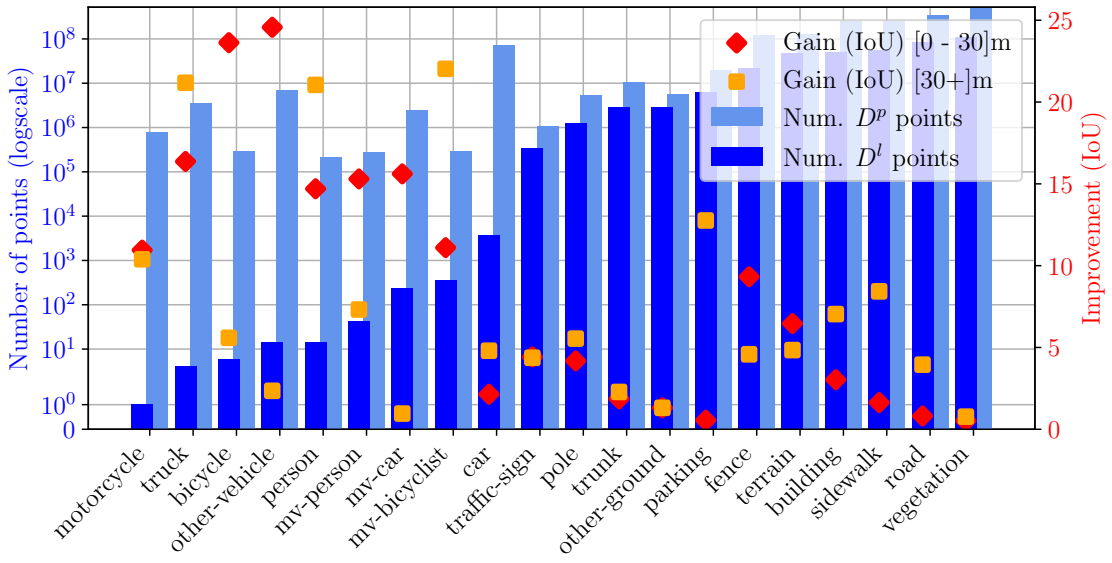


Figure 4.9: **Distance-based improvement in class IoUs on SemanticKITTI.** Classes are sorted according to the number of associated labeled training points in $\mathcal{D}^\ell$ and visualized together with pseudo-labeled points ($\mathcal{D}^\text{p}$). We show relative IoU gain within a 30-meter distance from the ego-vehicle ([0-30]m) and at a further distance ([30+]m). Moving objects are prefixed with 'mv'; N.B.: we have omitted classes that do not have points in the labeled split $\mathcal{D}^\ell$.

4.6 Discussion

Conclusion. We propose a novel pseudo-labeling framework that leverages spatio-temporal information from unlabeled sequences of point clouds. We demonstrate its merit in two 3D perception tasks on publicly available datasets. The reported performance gains stem from (i) A better selection of the final pseudo-labels via the concordance of multiple teachers operating at different temporal ranges; (ii) A novel pseudo-label confidence-guided criterion. Thanks to the privileged information available in the different temporal ranges, the Concordance of teachers delivers strong pseudo-labeled samples. Using manual labeling of only 20% of training data, our method achieved state-of-the-art performance in semi-supervised 3D semantic segmentation and competes even with methods that use the full set of labels on this task. By the nature of our pseudo-labeling framework, the proposed approach is complementary to other techniques that use sequential data, and can thus be combined with them to further boost the performance.

Limitations. The SemanticKITTI dataset has a huge data imbalance, and we perform the $\mathcal{D}^\ell$ and $\mathcal{D}^\text{u}$ split without any relevance to the number of points per object class. We observe cases where no points belong to a specific object category in the labeled set $\mathcal{D}^\ell$ resulting in teachers' inability to recognize a particular class, see 'mv-truck' and 'mv-other' in Table 4.3. One should ensure that all the object categories are present in the labeled set $\mathcal{D}^\ell$. Secondly, in object detection, since we estimate only objects in the reference frame, we sometimes observe false *false positives*, i.e., detections that

are correct but missing in the GT annotation due to no points in the reference frame. The model learns point features from different times and estimate vehicle position in the reference frame. We did not address this issue in evaluating the object detection task.

Future work. The proposed spatial-temporal aggregation effects only the input of the network and only indirectly effect performance. One potential follow-up work is to better initialize architecture with the help of Vision foundation models. We chose to follow-up by explicitly pairing the points over temporal horizon and enforce consistent predictions scene flow, which is described in the following sections.

Chapter 5

Motion Features From Scene Flow

5.1 Introduction

Scene flow is defined as a three-dimensional motion field of points in the physical world. As it is a key input for many fundamental robotic and computer vision tasks, such as ego-motion estimation [2, 110], instance segmentation [105, 47], scene reconstruction [63] and object detection [26], its accurate estimation is crucial. Learning-based approaches to scene flow estimation were first fully supervised as in the seminal work [70]. However, ground-truth scene flows in [70] and follow-up works [57, 18, 127, 85] are usually synthetic, as annotations of real-world scene flows are very scarce. This makes fully-supervised methods challenging to scale and adapt to real-world use cases. Consequently, the research community shifted its attention to self-supervised methods that learn to estimate the scene flow from un-annotated sequences of point clouds, which proved to be very efficient [59].

The first mechanism at work in self-supervised methods is point cloud alignment: for each pair of consecutive point clouds in the training set, point-to-point correspondences are established based on spatio-temporal or visual similarity, and the model tries to predict flows that approximate these correspondences at best. If no additional constraints are considered, the predicted flows can arbitrarily deform geometric structures in the scene, which often ends up in a degenerate solution due to the high number of incorrect correspondences; see the top-left image in Figure 5.2 for an illustration. To avoid this, additional regularization terms are often introduced to enforce desirable properties. The regularization term typically determines *rigid clusters* (subsets of points corresponding ideally to a single rigid body) and enforces their flow to be a rigid motion via so-called *smoothness* [59] or *rigid loss* [37]; see top-right image in Figure 5.2. However, state-of-the-art models [57, 59] rely on clusters that are both over-fragmented (several clusters on the same object) and inaccurate (clusters bleeding over object borders), which limits the benefits of the regularization: one still observes unrealistic flows that deform rigid bodies substantially.

In this topic, we proposed to revisit how self-supervised scene flow learning is regularized, thus

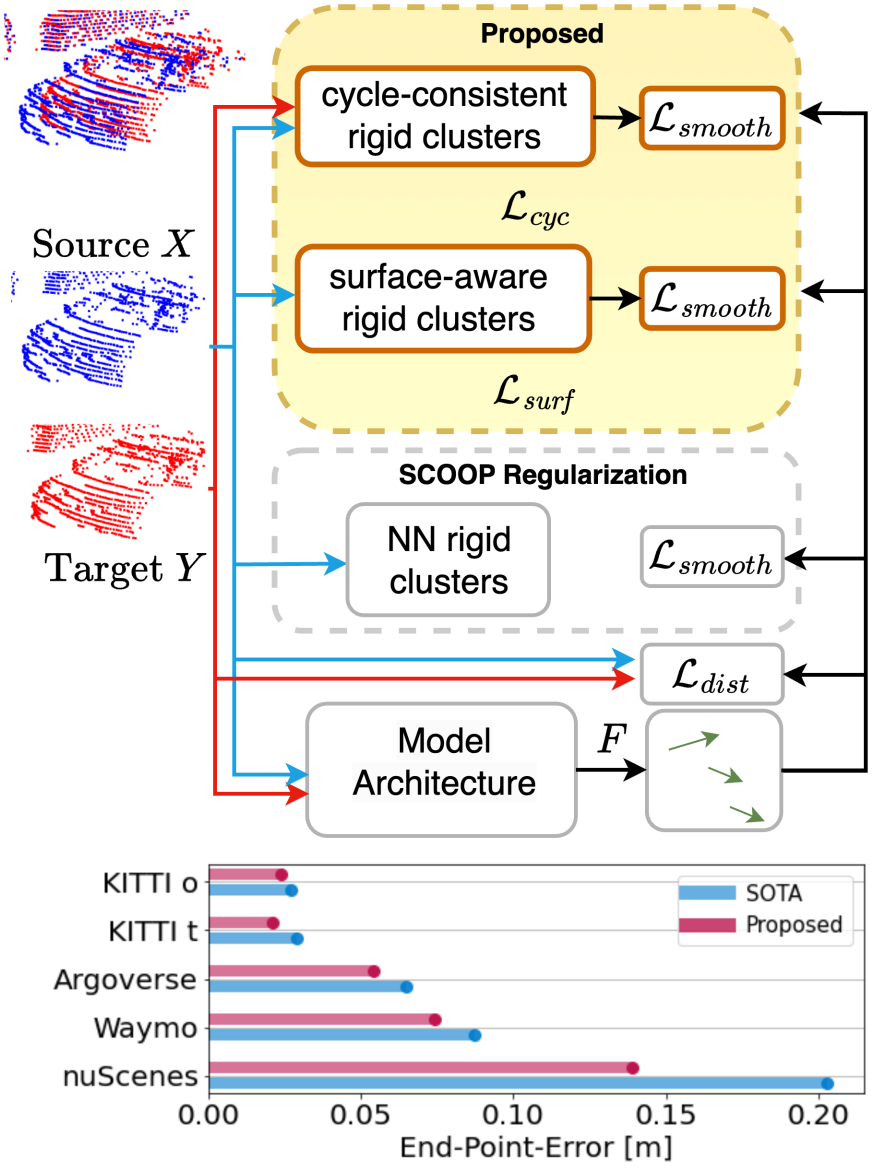


Figure 5.1: **Proposed self-supervised scene flow framework.** Self-supervised scene flow prediction is usually trained with losses that enforce the alignment of source and target point clouds and the smoothness of the flow ($\mathcal{L}_{dist}$ and $\mathcal{L}_{smooth}$ respectively). We improve the latter by introducing a surface-aware loss, $\mathcal{L}_{surf}$, and a cyclic temporal consistency one, $\mathcal{L}_{cyc}$. The proposed framework outperforms the state of the art on all tested datasets.

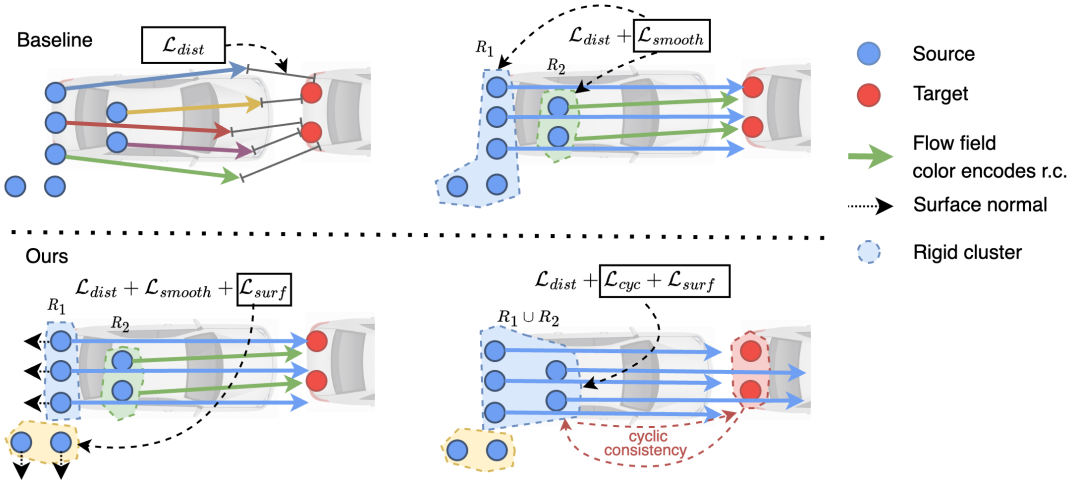


Figure 5.2: **Illustration of baseline (top) and proposed (bottom) losses to train self-supervised scene flows**. (*Top-left*) Classic approaches first enforce the alignment of the two point clouds, irrespective of their structure. This results in wrong correspondences and incorrect flows. (*Top-right*) To improve results, local smoothness enforces motion consistency within rigid clusters. Defined only on the proximity of points, such clusters can typically be too small (R_2) or connect unrelated rigid bodies (R_1), which limits the efficiency of the smoothness loss. (*Bottom-left*) By taking into account surface orientation similarity in addition to spatial proximity in the definition of clusters, we mitigate the latter issue. (*Bottom-right*) We also propose a new cyclic consistency loss that enforces two-way time consistency between the source and target point clouds, based on significantly larger and more accurate rigid clusters. In each figure, flow vectors are colored by rigid clusters ('r.c.'), e.g., there are all colored differently when using only $\mathcal{L}_{dist}$.

reaching novel state-of-the-art performance on several benchmarks. In more detail, we made several contributions [114, 115]:

1. We proposed a novel way to form rigid clusters by explicitly considering the cyclic smoothness [114] of the flow between source and target scans. This delivers significantly larger clusters into the smoothness loss (Figure 5.2, bottom-right).
2. We are first to improve the search for point correspondences in the alignment loss with the assumption of local surface regularity [114] (Figure 5.2, bottom-left).
3. We proposed a novel clustering method [115], where the rigid object segmentation is initialized from spatial-temporal grouping and is jointly optimized with the flow.
4. We compared quantitatively our framework on four publicly-available sensor-unique datasets [11, 7, 78, 106], with several baselines, and we reach new state-of-the-art results in all setups.
5. We achieved significant improvements on long tail classes [115], such as independently moving pedestrians and cyclists, where current solutions fail due to structural prior regularization.

5.2 Related Work

Supervised scene flow regression During the nascent phases of learning-based 3D scene flow estimation, methodologies leaned upon synthetic datasets [76, 39, 123] for initial training. However, owing to the sim-to-real distribution gap, these approaches yield sub-optimal performance when confronted with real-world point clouds [63]. FlowNet3D [70] adeptly incorporates principles from FlowNet [42] and underwent fully supervised training, employing L2 loss with ground-truth flow annotations. Meteor-Net [71] further leverages temporally ordered frame inputs to enhance the quality of flow inference. A continuous convolution, as introduced in [123], compensates for both ego-motion and object-motion. HPLFlowNet [39] turns points into a permutohedral lattice and applied bilateral convolution. FLOT [85] proposed a correspondence-based network, computing an optimal transport as an initial flow, followed by flow refinement through trained convolutions. BiPFN [18] bidirectionally propagates features from each point cloud, enriching the point feature representation. A hierarchical network that directly obtains key points flow through a lightweight Trans-flow layer employing local geometry priors was proposed in [23]. The recently unveiled IHNet architecture [127] proposes an iterative hierarchical network, steered by high-resolution inferred information and estimating flow in a coarse-to-fine manner. While all aforementioned methods rely on ground-truth flows, our emphasis is on scalable ground-truth-free flow regression.

Self-supervised scene flow Recent methodologies circumvented the necessity for ground-truth flow by adopting self-supervised learning. The pioneering effort to escape full supervision emerged in [81], employing a self-supervised nearest neighbor loss and ensuring cycle consistency between forward and reverse scene flow. PointPWC-Net [131] proposes a wholly self-supervised methodology, leveraging a combination of nearest-neighbors and Laplacian losses. Flowstep3D [57] estimates flow with local and global feature matching using correlation matrices and iterative warping. The method called SCOOP [59] utilizes a hybrid framework of correspondence-based feature matching, followed by a flow refinement layer using self-supervised losses. Li et al. [61] address the common challenges of neglecting surface normals and the potentiality of many-to-one correspondences in matching. They reframe the matching task as an optimal transport problem. RigidFlow [62] posits a strategy for generating pseudo scene flow within the realm of self-supervised learning. This approach hinges on piecewise rigid motion estimation applied to sets of pseudo-rigid regions, discerned through the supervoxels method [68]. Notably, the method emphasizes region-wise rigid alignments as opposed to point-wise alignments. In a parallel vein, Najibi et al. [82] harness self-learned flow in an automated pseudo-labeling pipeline. This method is directed towards training an open-set object detector and trajectory predictor. Speeding up the Neural Scene Flow stands as the focal aim in [64]. The authors pinpoint the Chamfer distance as a computational bottleneck and reintroduce the Distance Transform as a correspondence-free loss function. Remarkably, they attain real-time performance with precision on par with learning-based methods. The work of Agostinho et al. [1] introduces a differentiable layer for rotation estimation refinement from an initial guess provided by Kabsch [54] algorithm, which extends the optimization of the point cloud registration problem. Integration of this layer into conventional learning-based methodologies yields an enhancement in the overall quality of the estimated flow. In the pursuit of self-supervised scene flow estimation, Shen et al. [99] incorporates superpoint generation directly into the model. They implemented an iterative refinement process for both the flow and dynamically evolving superpoints. Our method belongs to the self-supervised family, as we do not require human labels. We also regularize flow by enforcing smoothness on the pre-computed smaller groups of points compared to rigid movement on under-segmented clusters as in [62].

Direct flow optimization. The Graph Prior approach, as introduced by [84], is rooted in pure optimization, enabling flow estimation without the need for training data. The Neural Prior[63] illustrates that flows can be optimized by incorporating a structure-based prior within the network architecture. The central objective in [64] revolves around expediting the Neural Scene Flow [64]. Li *et al.* identify the Chamfer distance as a computational bottleneck and use the Distance Transform as a surrogate correspondence-free loss function. The methodology outlined in Scene Flow via Distillation by [118] embodies a direct distillation framework. This innovative approach utilizes a label-free optimization method to generate pseudo-labels, which are subsequently employed to

supervise a feed-forward model to achieve better speed/performance ratio. The two newest optimization-based methods, the MBNSFP [119] and Chodosh [20] adopt Neural Prior architecture and enhanced it with spatial consistency regularization [119] and post-process cluster rigidity [20], respectively. We also use rigidity regularization, but we do not limit our method to fixed initial clusters as [119, 20]. Instead, we jointly optimize initial cluster and flow to progressively merge and enlarge clusters while enforcing rigidity on them. Another difference to aforementioned methods, is the absence of Neural Prior as structural regularization, which plays crucial role for populated dynamic scenes.

Registration and rigidity regularization Since the most standard self-supervised nearest-neighbor loss has multiple local minima, one must introduce regularization mechanisms to reach physically plausible flows, i.e., that follow well the motion of rigid structures and objects. Weak supervision in the form of ego-motion and foreground segmentation was shown to provide an object-level abstraction for the estimation of rigid flows [37]. Without the segmentation signals, the FlowStep3D [57] enforces the source point cloud to preserve its Laplacian when warped according to the predicted flow. The Laplacian was approximated by the point nearest neighborhood, forming the smoothness loss. Similarly, SLIM [2] incorporates the smoothness loss in the process of learning network weights. Meanwhile, SCOOP [59] integrates the smoothness loss within the optimization-based flow refinement layer. Implicit rigidity regularization was successfully imposed via strong model prior in [63, 3, 138]. In [15], a novel global similarity measure takes the form of a second-order spatial compatibility measure on consensus seeds, that serve as input to a weighted SVD algorithm, ultimately producing global rigid transformation.

We focus on meaningful regularization of self-supervised methods. To enhance the flow rigidity, we introduce novel losses that leverage the correspondence neighborhood in the target point cloud, matched to the source through the flow vectors, and introduce temporal cyclic smoothness. We also introduce surface-aware smoothness based on normals to separate close rigid clusters. Contrary to [15], we use spatial consistency mechanism for rigidity regularization, but for multiple objects.

5.3 Methodology of Surface Regularity and Cyclic Smoothness

Given two successive point clouds $X \in \mathbb{R}^3$ and $Y \in \mathbb{R}^3$ captured from the same dynamic scene at instants t and $t + \delta t$, the scene flow $F = \{\mathbf{f}_\mathbf{x}, \mathbf{x} \in X\}$ at time t is the set of the 3D displacements of points in *source* point cloud X over time interval δt, the 3D motion field in other words. The *target* point cloud Y is leveraged to predict this flow.

Learning how to predict scene flows from unlabeled training pairs of point clouds is classically done using an unsupervised loss that promotes point cloud alignment along with flow smoothness under the assumption that the scene is mainly composed of rigid objects or object parts. We propose to improve such self-supervised approaches with the introduction of two novel consistency losses, as

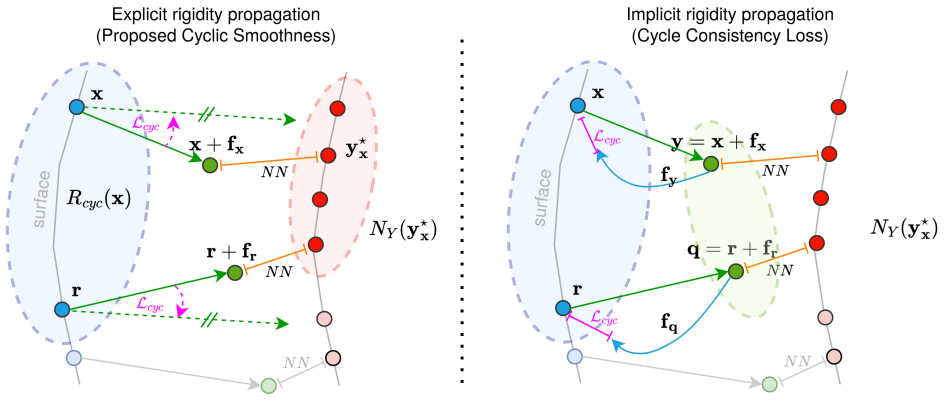


Figure 5.3: **Cyclic Smoothness loss.** The new loss $\mathcal{L}_{cyc}$ enforces the same flow (dashed green arrow) over the rigid cluster $R_{cyc}(\mathbf{x})$ (light blue set) defined as follows: Given the source point $\mathbf{x}$ and its best match $\mathbf{y}^\star_\mathbf{x}$ in target point cloud according to flow F, we construct its k-nearest neighborhood $N^k_Y(\mathbf{y}^\star_\mathbf{x})$ (light red set). Any source point $\mathbf{r}$ whose flow $\mathbf{r}+\mathbf{f_r}$ sends it there is included in the rigid cluster $R_{cyc}(\mathbf{x})$. While the proposed $\mathcal{L}_{cyc}$ (left) *explicitly* detects the rigid object as a compact cluster via normals similarity in the target point cloud and then propagates the knowledge directly to the source point cloud by enforcing rigid flow, the baseline Cycle Consistency [81] $\mathcal{L}_{cycle}$ (right) *implicitly* detects the rigid object by running the flow prediction in green point cloud $(\mathbf{x}+\mathbf{f_x})$ in the backward direction and then enforces the flow (blue arrows) to get back into its source.

illustrated in the bottom part of Figure 5.2.

5.3.1 Baseline losses of self-supervised scene flow

If no correspondence annotations are available, a typical self-supervised loss favors a flow that aligns as much as possible the source point cloud to the target one, e.g, using the nearest-neighbor distance [81]:

$$\mathcal{L}_{dist}(F) = \frac{1}{|X|}\sum_{\mathbf{x}\in X}\min_{\mathbf{y}\in Y}\|\mathbf{x}+\mathbf{f_x}-\mathbf{y}\|_2^2. \tag{5.1}$$

Since this loss enforces no spatial consistency whatsoever, each point is allowed to flow independently, and rigid objects can get substantially deformed or fragmented. To prevent this in scenes with rigid objects, one can define for each point $\mathbf{x}\in X$ a *rigid cluster* $R(\mathbf{x})\subset X$, i.e, a set of other points likely to lie on the same rigid fragment. A typical choice [57, 59] is to simply consider the set $N^k_X(\mathbf{x})$ of k-nearest neighbors of $\mathbf{x}$ in X. Once rigid clusters $R(\mathbf{x})$ are defined, the smoothness of the flow F is classically enforced through a robust L_1 loss:

$$\mathcal{L}_{smooth}(F,R) = \frac{1}{|X|}\sum_{\mathbf{x}\in X}\frac{1}{|R(\mathbf{x})|}\sum_{\mathbf{r}\in R(\mathbf{x})}\|\mathbf{f_x}-\mathbf{f_r}\|_1. \tag{5.2}$$

It is desirable to have rigid clusters as large as possible to maximize the benefit of the loss,

preventing rigid objects from getting deformed or fragmented. However, in the case of simply using k nearest neighbors, a too-large value of k incurs the risk of wrongly grouping multiple independent rigid parts into a common rigid cluster. Striking the right balance is challenging.

Towards this end, we propose to refine the definition of the clusters so as to respect the spatio-temporal consistency of objects. In particular, we extend the definition by explicitly reconstructing the surface of objects, and by enforcing cyclic consistency of rigid clusters between source and target point clouds.

5.3.2 Surface smoothness

We improve first the definition of rigid clusters by taking into account the orientation of the object's surface at each point besides its mere position. This is significant information in scenes with a prevalence of nearly planar surfaces; it will help separate two close rigid objects that have different motions. Denoting $\mathbf{n_x}$ the normal of the surface captured by X at position $\mathbf{x}$, we define a novel surface-aware descriptor $\phi_\mathbf{x} = (\mathbf{x}, \mathbf{n_x}) \in \mathbb{R}^6$ at this location. It allows us to compute a surface-aware cluster

$$R_{surf}(\mathbf{x}) = N^k_\Phi(\phi_\mathbf{x}), \tag{5.3}$$

where $\Phi = \{\phi_\mathbf{x}, \mathbf{x} \in X\}$. In practice, we find k_n nearest neighbors of the point $\mathbf{x}$, followed by obtaining principal vectors of covariance matrices of each of the points in the neighborhood. The main principal vector normalized to one corresponds to the normal vectors $\mathbf{n_x}$, which is used to construct surface-aware descriptor ϕ. Grouping similar surfaces like this improves local rigidity. Then, we follow the method described in [111] for sign disambiguation. The implementation is available in the open-source PyTorch3D library [90]. The new surface-aware smoothness loss then reads:

$$\mathcal{L}_{surf}(F) = \mathcal{L}_{smooth}(F, R_{surf}). \tag{5.4}$$

By using this loss instead of the regular one, flow consistency is enforced among pairs of scene elements that are close and similarly oriented.

5.3.3 Cyclic smoothness

We now introduce a second smoothing loss that enforces cyclic (forward-backward) consistency between times t and $t + \delta t$. It aims at propagating the information about object rigidity from the target point cloud back to the source one via forward correspondences, see Figure 5.3.

Following matching loss (5.1), a given scene flow F effectively matches each source point $\mathbf{x}$ with a point $\mathbf{y}^\star \in Y$:

$$\mathbf{y}^\star = \arg\min_{\mathbf{y} \in Y} \|\mathbf{x} + \mathbf{f_x} - \mathbf{y}\|_2^2. \tag{5.5}$$

Based on this correspondence, we define the *cyclic rigid cluster* of point $\mathbf{x}$ as a set of points $\mathbf{r} \in X$

whose displaced position according to F falls within the neighborhood of $\mathbf{y}^\star_\mathbf{x}$. More formally:

$$R_{cyc}(\mathbf{x}) = \{\mathbf{r} \,|\, \mathbf{r} + \mathbf{f}_\mathbf{r} \in N_Y^k(\mathbf{y}^\star_\mathbf{x})\}. \tag{5.6}$$

The corresponding cycle-consistent loss reads:

$$\mathcal{L}_{cyc}(F) = \mathcal{L}_{smooth}(F, R_{cyc}). \tag{5.7}$$

Note that the definition of this new loss does not introduce any extra parameters.

5.3.4 Training objective and Architecture

Given an architecture that can be trained to predict a scene flow F from a pair (X, Y) of point clouds, we can make use of the novel proposed losses. In that case, the training objective minimizes the loss:

$$\mathcal{L} = \mathcal{L}_{dist} + \alpha_{surf}\mathcal{L}_{surf} + \alpha_{cyc}\mathcal{L}_{cyc}, \tag{5.8}$$

where α_{surf} and α_{cyc} are hyperparameters balancing the contribution of individual loss terms. This approach is architecture-independent and can thus be combined with existing models. In practice, we put it at work with two existing scene flow models, namely SCOOP [59] stereo-based datasets and Neural Prior [63] for real LiDARs.

5.4 Experiments

In this section, we demonstrate quantitatively and qualitatively the performance gains stemming from our proposed losses on two state-of-the-art architectures and four benchmarks. We also analyze these gains, as well as our design and parameter choices, in ablation studies.

Datasets and metrics. We performed experiments on four scene flow datasets. The first one is stereoKITTI [78], which contains real-world self-driving scenes. We used the commonly benchmarked subset released by [70] consisting of point clouds created from stereo images. The dataset, dubbed KITTI_o, was split by [81] into train and test parts denoted KITTI_v and KITTI_t respectively. Additionally, the authors introduced an unlabelled dataset, KITTI_r, consisting of actual LiDAR sensor data from the same driving sequences. Next, we used three large-scale LiDAR autonomous driving datasets, namely Argoverse [11] (and also Argoverse2 [129]), nuScenes [7], and Waymo [106], with challenging dynamic scenes captured by different LiDAR sensors. The LiDAR datasets were sampled and processed according to [63] for the fair comparison.

As there are no official scene flow annotations, we adopted the data processing approach from [53] to derive pseudo-ground-truth scene flow information based on object detection annotations, akin to

Table 5.1: **Comparative results of scene flow estimation methods on stereo-based datasets.** We evaluate scene flow based on standard metrics *EPE*, *AS*, *AR* and *Out*, for different settings of supervision, train data and test data. Combined with two architectures (SCOOP and Neural Prior), our approach beats all fully-supervised baselines trained on FT3D as well as all self-supervised methods. There are differences between the reported performance ('*') of SCOOP and the one recomputed from original codebase ('†'), which surpasses the former. To be fair in evaluation, we report both performances while optimizing the flow until convergence to achieve the models' upper-bound performance. Our proposed loss terms improve the performance even further. For our method, metrics are averaged over 6 runs.

Method	Supervision	Train data	Test data	*EPE* [m] ↓	*AS* [%] ↑	*AR* [%] ↑	*Out.* [%] ↓
FlowNet3D[42]	Full	FT3D	KITTI_o	0.173	27.6	60.9	64.9
FLOT[85]	Full	FT3D	KITTI_o	0.107	45.1	74.0	46.3
BiPFN[18]	Full	FT3D	KITTI_o	0.065	76.9	90.6	26.4
R3DSF[37]	Full	FT3D	KITTI_o	0.042	84.9	95.9	20.8
FlowStep3D[57]	Self	FT3D	KITTI_o	0.102	70.8	83.9	24.6
SCOOP*[59]	Self	FT3D	KITTI_o	0.047	91.3	95.0	18.6
SCOOP†[59]	Self	FT3D	KITTI_o	0.037	89.4	94.9	18.0
SCOOP + Ours	Self	FT3D	KITTI_o	**0.024** (-35%)	**97.1**	**98.6**	**13.9**
Graph Prior[84]	Self	N/A	KITTI_t	0.082	84.0	88.5	-
Neural Prior[63]	Self	N/A	KITTI_t	0.036	92.3	96.2	13.2
Neural Prior + Ours	Self	N/A	KITTI_t	**0.034** (-5%)	**92.5**	**97.5**	**12.9**
JGWTF[81]	Self	nuScenes + KITTI_v	KITTI_t	0.105	46.5	79.4	-
SPF[99]	Self	KITTI_r + KITTI_v	KITTI_t	0.089	41.7	75.0	-
SCOOP*[59]	Self	KITTI_v	KITTI_t	0.039	93.6	96.5	15.2
SCOOP†[59]	Self	KITTI_v	KITTI_t	0.029	95.9	98.0	12.2
SCOOP + Ours	Self	KITTI_v	KITTI_t	**0.021** (-28%)	**98.9**	**99.5**	**11.3**

the methodology employed in [63]. Finally, we removed ground points in the height of 0.3 meters or lower for Waymo dataset, following the procedure outlined in previous works [70, 63, 59, 64, 37, 119]. For Argoverse1 and Argoverse2, we used the accompanying ground maps to remove ground points and constrain the range of input points to 35 meters as done in [20, 37, 131, 59] if not stated otherwise.

Evaluation metrics. For proper evaluation of results we need to define *point error* e_i and *relative point error* e_i^r. We define these errors in meters as follows:

$$e_i = \|\mathbf{f}_i - \mathbf{f}_i^{\text{gt}}\|_2, \quad e_i^{\text{r}} = \frac{\|\mathbf{f}_i - \mathbf{f}_i^{\text{gt}}\|_2}{\|\mathbf{f}_i^{\text{gt}}\|_2},$$

where $\mathbf{f}_i$ and $\mathbf{f}_i^{\text{gt}}$ are the predicted and ground-truth flow for point $\mathbf{p}_i$, respectively. From e_i, e^{r} we calculate Average Point Error in meters (EPE), Strict Accuracy (AS), Relaxed Accuracy (AR), Angle Error (θ) and Outliers (*Out.*), which are standard metrics used in the literature, e.g., [63, 59]. AS is a percentage of points that reached errors $e_i < 0.05$ or $e_i^{\text{r}} < 5\%$. The AR is the percentage of points for which the error satisfies either $e_i < 0.1$ or $e_i^{\text{r}} < 10\%$. Metric *Out.* is the percentage of points with error either $e_i > 0.3$ or $e_i^{\text{r}} > 10\%$, and finally θ denotes the mean angle error between $\mathbf{f}_i$

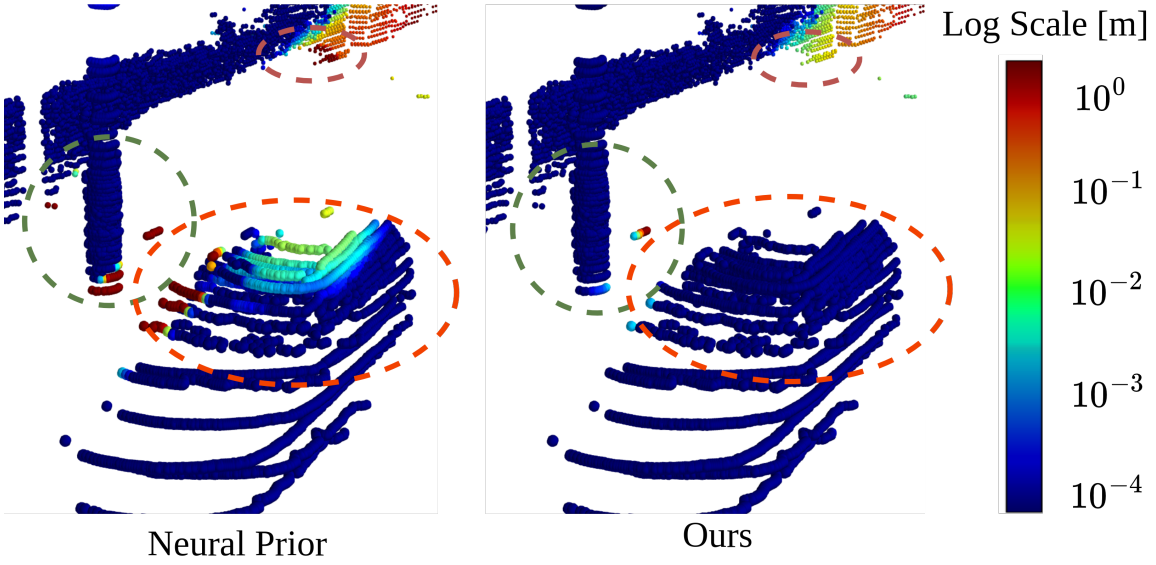


Figure 5.4: **Example of improvements brought by proposed framework on real LiDAR data**. In this scene from the Argoverse dataset, we show the per-point flow estimation error encoded by color on a logarithmic scale. While the Neural Prior [63] baseline on the left fails to produce consistent flows along the majority of the vehicle body (orange ellipse), the addition of our proposed losses corrects the full body rigidity. The same applies to the pole (green ellipse) and the wall in the back (red ellipse).

Table 5.2: **Performance on LiDAR datasets.** We show performance on standard LiDAR benchmarks with commonly reported metrics. Fully-supervised methods are trained on $FT3D_o$ dataset. When adding the proposed losses, we consistently improve the performance of the state-of-the-art Neural Prior method across all datasets. Our method results are averages over 6 runs.

Method	D	*EPE* [m]↓	*AS* [%]↑	*AR* [%]↑	θ [rad]↓
FlowNet3D[42]	Argoverse	0.455	1.34	6.12	0.736
JGWTF[81]		0.542	8.80	20.28	0.715
PointPWC-Net[131]		0.409	9.79	29.31	0.643
Neural Prior[63]		0.065	77.89	90.68	0.230
Neural Prior + Ours		**0.054** (-17%)	**81.11**	**92.51**	**0.223**
FlowNet3D[42]	Nuscenes	0.505	2.12	10.81	0.620
JGWTF[81]		0.625	6.09	13.90	0.432
PointPWC-Net[131]		0.431	6.87	22.42	0.406
Neural Prior[63]		0.203	49.64	76.03	0.244
Neural Prior + Ours		**0.139** (-32%)	**55.56**	**80.43**	**0.220**
R3DSF[37]	Waymo	0.414	35.47	44.96	0.527
Neural Prior[63]		0.087	78.96	89.96	0.300
Neural Prior + Ours		**0.074** (-15%)	**81.65**	**91.45**	**0.290**

and $\mathbf{f}_i^{\text{gt}}$.

In automotive scenes, the majority of the points come from a static background. When the metrics (*EPE*, *AS*, *AR*) are calculated and averaged for the full point cloud, the results show mostly performance on static points, lowering the importance of dynamic objects [20, 118]. If we dig even further, the dynamic points are heavily dominated by large vehicles and the performance on smaller, yet equally important classes, like Pedestrians and Cyclists are not observable from the dynamic *EPE* metrics. Therefore, we also report the metrics per-class on the main benchmarks like Argoverse2. We compare the metrics based on previous state-of-the-art for fair comparisons, i.e. on StereoKITTI, we do not use the threeway *EPE* but overall *EPE*.

Baselines, settings, implementation details. We benchmark our method with other 3D scene flow frameworks such as fully-supervised methods trained on synthetic FT3D dataset [77] with ground-truth flows [70, 57, 85], weakly-supervised models with access to the foreground segmentation labels [37], self-supervised models [59, 81, 131], and optimization-based approaches [63, 84].

Point clouds in both FT3D and stereoKITTI are obtained similarly by lifting stereo images to 3D through ground-truth 2D optical flow. Hence, their domain gap is relatively small, and they offer cross-dataset evaluation, enabling models to be trained on synthetic FT3D in a fully-supervised fashion. On the other hand, in LiDAR-based autonomous driving scenarios, point clouds are much sparser and provide different sampling patterns with no direct one-to-one correspondences, resulting in a much more challenging setting for scene flow estimation. To be consistent with the

Table 5.3: **Influence of proposed losses.** The performance reached with various loss combinations is evaluated on nuScenes dataset with Neural Prior architecture. We observe that, by gradually adding the proposed losses, we improve the performance in all metrics.

$\mathcal{L}_{smooth}$	$\mathcal{L}_{cyc}$	$\mathcal{L}_{surf}$	EPE [m] ↓	AS [%] ↑	AR [%] ↑
			0.203	49.70	76.03
✓			0.175	53.22	79.11
	✓		0.163	50.04	76.47
		✓	0.166	55.18	80.39
✓	✓		0.145	53.59	79.27
	✓	✓	**0.139**	**55.56**	**80.43**

evaluation of top-performing scene flow models, we test our method on all points present in the scene and not partial chunks and optimize until the convergence of the loss or until reaching the maximum predefined number of iterations. We incorporate in our framework the two top-performing architectures SCOOP [59] and Neural Prior [63], which are the current state-of-the-art respectively on stereoKITTI and on LiDAR scene flow datasets.

Our method is implemented in PyTorch and trained on one NVIDIA Tesla A100 GPU. We use the default parameters for both architectures, as reported in the original papers. We use $\alpha_{surf} = 10$ as SCOOP does for the regular weight of smoothness on the stereoKITTI dataset, and we similarly set $\alpha_{cyc} = 10$. For Neural Prior, we use $\alpha_{surf} = 1$ since there is already a regularization included in the network structure, and we keep $\alpha_{cyc} = 10$ for consistency in all experiments.

For the stereoKITTI dataset, we use $k = 32$ for k-nn computations as in the original SCOOP configuration and $k = 4$ for LiDAR datasets since LiDAR point clouds are much more sparse. Lastly, we compute normals for surface smoothness based on $k_n = 5$ nearest neighbors for all datasets.

5.4.1 Evaluation on StereoKITTI

In Table 5.1, we show the results of our method with tested architectures on the stereoKITTI dataset in comparison to baselines. Coupled with existing self-supervised architectures, our losses outperform the fully-supervised models trained on large synthetic FT3D datasets and the self-supervised and optimization-based methods. Mainly, our self-supervised loss terms improve the state-of-the-art architectures on both KITTI_o and KITTI_t splits. On these two splits, our proposed loss terms deliver relative *EPE* gains of 35.1% and 27.5%, respectively, on top of state-of-the-art SCOOP architecture. Performance is boosted in all other metrics as well. These results demonstrate the merit of the proposed losses: they bring current state-of-the-art scene flow models to new performance levels.

5.4.2 Generalization over various LiDAR datasets

In Table 5.2, we demonstrate the benefit of our loss terms applied in combination with Neural Prior architecture on different LiDAR datasets. While we keep the proposed parameters unchanged for

Regularization	*EPE*[m]	*AS*[%]	*AR*[%]
$\mathcal{L}_{cycle}$	0.168	49.36	76.56
$\mathcal{L}_{cyc}$	0.163	50.04	76.47
$\mathcal{L}_{cycle} + \mathcal{L}_{sm}$	0.149	53.08	79.30
$\mathcal{L}_{cyc} + \mathcal{L}_{sm}$	0.145	53.59	79.57
$\mathcal{L}_{cycle} + \mathcal{L}_{cyc} + \mathcal{L}_{sm}$	0.132	55.41	80.76

Table 5.4: Result of cycle losses on nuScenes, without $\mathcal{L}_{surf}$ to isolate the effect of $\mathcal{L}_{cyc}$. Standard losses in blue and our proposed component in red.

all the datasets in the paper, we achieve consistent improvement in all cases with various sensor configurations, even on much sparser nuScenes LiDAR. We improved the performance of Neural Prior architecture in all of the evaluation metrics. Such consistent gains demonstrate that the assumptions of surface rigidity and regularity that underpin our losses are agnostic to the sensor domain and sampling pattern. See Figure 5.4 for a qualitative sample.

5.4.3 Ablations studies

Influence of loss terms. We examine the performance of various combinations for proposed losses with Neural Prior architecture on the nuScenes dataset; see Table 5.3. We observe the benefit of adding the smoothness loss to the existing prior regularization. Our proposed terms $\mathcal{L}_{cyc}$ and $\mathcal{L}_{surf}$ increase the performance even further when adding components separately, where $\mathcal{L}_{cyc}$ has a large impact on *EPE* and $\mathcal{L}_{surf}$ greatly increases *AS*. Note that $\mathcal{L}_{surf}$ substitutes the $\mathcal{L}_{smooth}$, and therefore combination of both at the same time is omitted.

Cyclic Smoothness vs. Cycle Consistency The name of our proposed Cyclic Smoothness implies similarities with Cycle Consistency loss introduced in JGWTF [81]. We demonstrate differences in Figure 5.3 and experimentally verify gains in Table 5.4. Both losses regularize the flow predicted in the blue source point cloud by transferring the knowledge about the object rigidity observable in the target point cloud, see Figure 5.3. The main difference between proposed cyclic smoothness $\mathcal{L}_{cyc}$ and standard cycle consistency $\mathcal{L}_{cycle}$ stems from the way the rigid object in the red point cloud is discovered. Both approaches can prevent trivial failure cases in which the flow of two distinct blue points ends up in a single red point; however, in more complicated cases, the behavior is different. The implicit rigidity propagation is extremely dependent on the presence of a high-density point cloud and the small noise of the predicted (green) point cloud in order to discover the actual rigid object successfully. We experimentally demonstrate that both proposed losses are complementary; see Table 5.4. The best results are achieved with the combination of both, while the second best results are achieved with the proposed explicit propagation.

Influence of Surface Smoothness on stereoKITTI We ablate the number k_n of nearest neighbor points needed to construct normals for $\mathcal{L}_{surf}$ on KITTI_t dataset in Figure 5.5. We observe

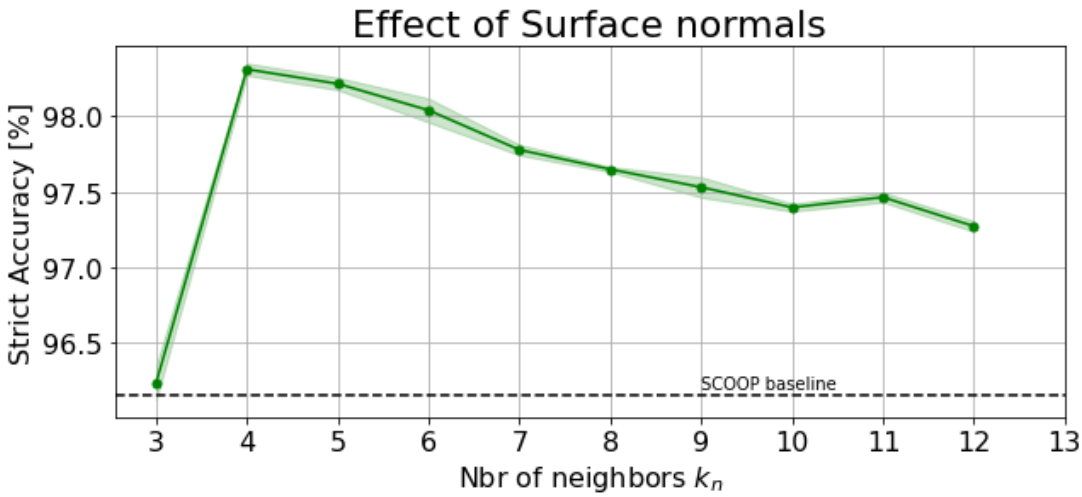


Figure 5.5: **Ablation of normal estimation on KITTI$_t$.** Influence on the performance of the number of neighboring points used to compute surface normals. The best performance for the SCOOP model is obtained with $k_n = 4$, and further increase of neighborhood diminishes the performance.

that the best performance is reached on the stereo-based dataset when four neighbors are used to compute normals. Adding more points diminishes the performance yet still surpasses SCOOP; see Figure 5.6 for an example of separating flows of vehicle and falsely segmented ground. The performance is exceptionally worse when we utilize the minimum number of points for normal estimation, i.e. *three*. We attribute this to the sampling bias of stereo camera data, where the three points most likely lie on a line that comes from adjacent pixels, making normal estimation ill-posed.

5.5 Discussion to Cyclic and Surface Smoothness

Conclusion. Self-supervised learning of scene flow without ground-truth signals is prone to degenerative solutions. We have introduced surface awareness and cyclic smoothness into the self-supervised learning framework, which proved essential for scene flow regularization and improved the performance of state-of-the-art models. The experiments demonstrated that the method works in real-world LiDAR settings, with point clouds from a standard stereo-based dataset, and with multiple network architectures, proving its generalization ability. Since our method aims for an improved selection of the object-centric rigid clusters, the proposed loss terms could also be adapted to instance segmentation in the future. We also plan to extend the cycle smoothness to work across multiple frames while seeking a broader temporal consistency.

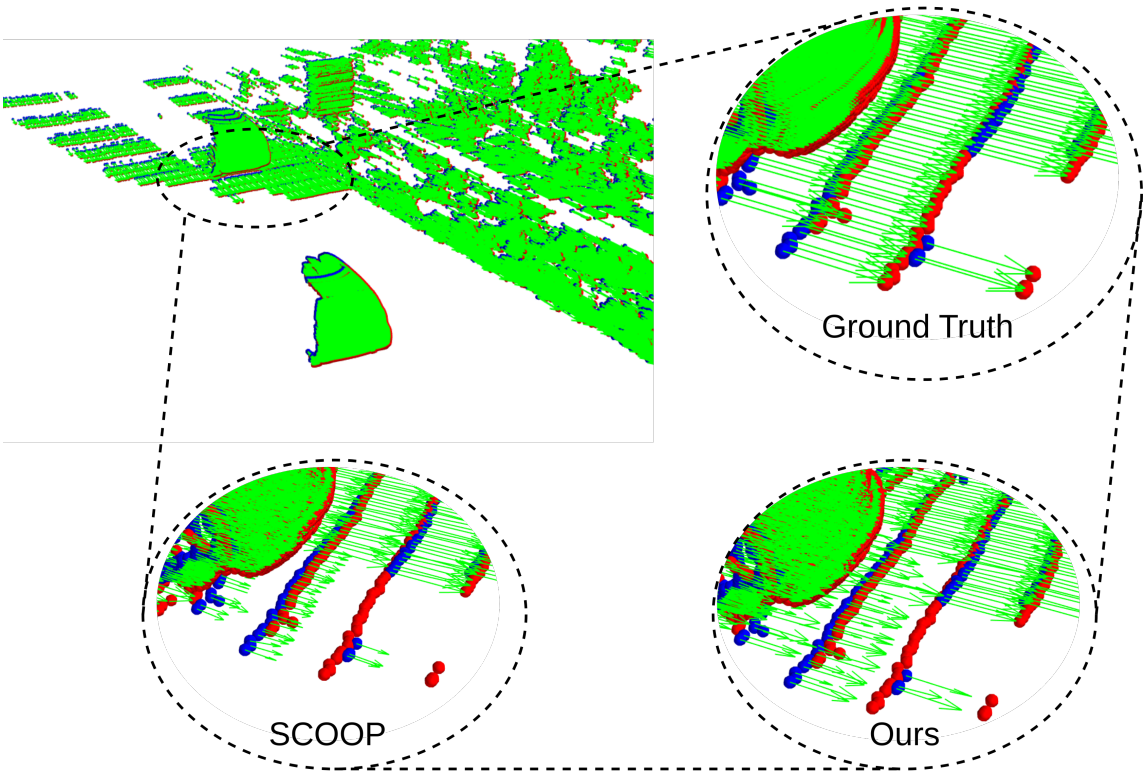


Figure 5.6: **Qualitative example on KITTI$_t$.** Top-left viewpoint shows a point cloud with flow after ground filtering with some ground falsely segmented. As a result, the ground with flow induced by ego movement shares the same neighborhood in smoothness loss. Our surface loss, however, groups points, taking into account the associated normals, thus separating the planar ground from the vehicle for a much more consistent flow.

Limitations. The cycle smoothness loss works best when there are multiple corresponding points found in the target point cloud neighborhood $N_Y(\mathbf{y}^\star_\mathbf{x})$. When there are no due to the sudden occlusion in the target frame or the points become out-of-range, the proposed cyclic smoothness converges into the regular smoothness loss. By increasing the number of points in rigid clusters in $\mathcal{L}_{cyc}$, we implicitly enforce the flow to be a purely translational movement rather than a more realistic rotation and translation. However, due to the high framerate of the sensor, the translation movement is a sufficient approximation for most of the applications. Note also that this issue is not specific to our approach as it is already a weak point of the standard smoothness loss.

If the point cloud is very sparse, $\mathcal{L}_{surf}$ would only introduce noise to the features for neighborhood calculation in final smoothness. Flow estimation also would not benefit from $\mathcal{L}_{surf}$ on planar objects.

Future work. We show improvement of surface-aware component even on nuScenes dataset, which has much sparser point clouds compared to other LiDAR datasets and, especially, the stereo dataset. However, the surface-aware component of our method is more precise when measurements are dense. The additional improvement may consist of adding future temporal horizon and shape reconstruction to densify the measurements or instance segmentation as a input to loss terms. We chose to follow-up with explicitly model the point grouping as object instance assignment rather than neighborhoods, which is described in following sections.

5.6 Joint Flow and Object Instance Optimization

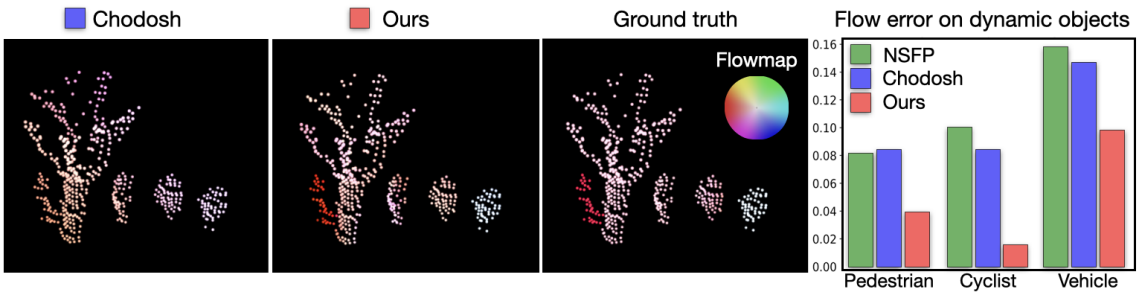


Figure 5.7: Comparison of the proposed method with self-supervised competitors on Argoverse2 Dataset [11]. Our method is able to distinguish the different motion patterns and separate objects, while other methods tend to under-segment objects and fit incorrect rigid motion.

Self-supervision in 3D flow estimation always stems from strong prior assumptions about the scene structure. State-of-the-art methods [37, 62, 105, 20, 119] typically construct spatially compact clusters [27], that are assumed to correspond to rigidly moving objects as a first step and then apply rigid flow regularization for these clusters. Such premature clustering inevitably suffers from *under-segmentation* (i.e. merging several rigid objects into a single cluster) and over-segmentation (i.e. disintegration of a single rigid object to multiple clusters). To partially suppress the under-segmentation issue, the outlier rejection technique has recently been proposed [119]. Nevertheless, the quality of the initial clustering remains the main bottleneck, which prevents estimating the accurate flow, especially in situations where multiple small objects are moving independently close to each other. In contrast to existing approaches, we generate many small *overlapping spatio-temporal rigid-cluster hypotheses* and then jointly optimize the flow with the rigid-body segmentation. We quantitatively and qualitatively demonstrate that such an approach mitigates the over and under-segmentation issues and consequently yields superior results, especially on dynamic objects; see Figure 5.7 for the comparison.

In the following self-supervised framework, we propose to employ two losses that are visualized using the analogy of a mechanical machine[1] with springs, see Figure 5.8. The equilibrium of this machine corresponds to the globally optimal flow. The machine consists of two consecutive pointclouds in time (black and red crosses) which are not allowed to move, flow arrows attached to blue points by swivel telescopic joints, and two losses (rigid and distance) represented by springs. The scene depicts two rigid vertical objects that are moving in different directions. We employ standard *distance loss* (red springs), which attracts the flow of black points toward the corresponding red points from the consecutive pointcloud. Since objects in black pointcloud are close to each other,

[1]Since both losses minimize the L2-norm of some quantities, we can visualize their influence on the estimated flow by ideal springs. In the ideal spring, the total conserved energy is proportional to the square of its deformation; therefore, the value of L2-loss corresponds to its energy.

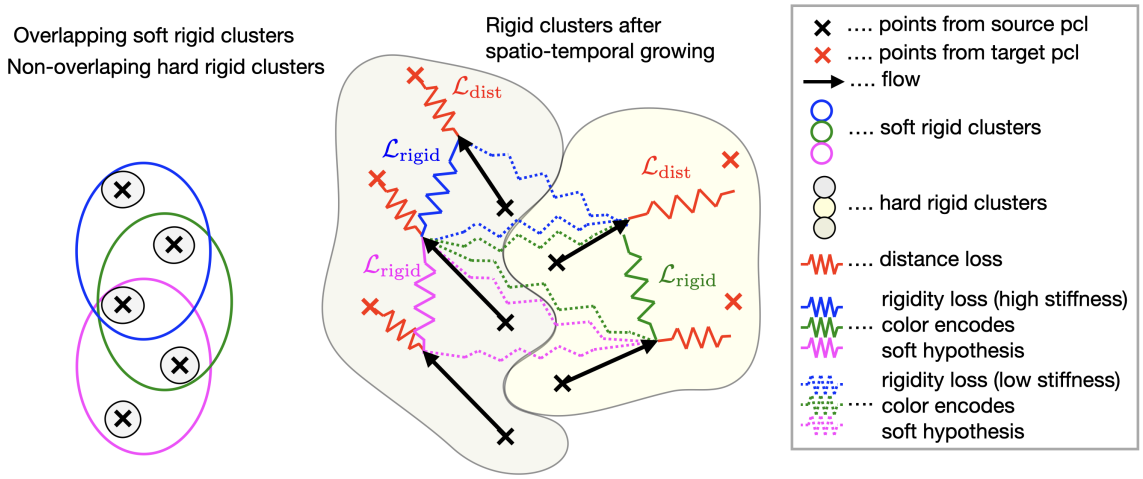


Figure 5.8: **Outline of the proposed losses:** The left image shows pointcloud represented by black crosses that contains two vertical rigid objects. There is no spatial clustering that would segment them correctly; therefore any independent flow estimation will be strongly biased by the incorrect clustering. In contrast, we cover the pointcloud by (i) non-overlapping hard rigid clusters and (ii) overlapping soft rigid clusters. The right image demonstrates losses, visualized by springs, that are used for the flow estimation. The resulting flow is used for merging the hard clusters on spatio-temporal domain. The procedure is repeated until convergence. The resulting hard rigid clusters delivers rigid object segmentation.

there is no spatial clustering that would separate them correctly. In contrast to existing methods, we instead introduce overlapping clusters (blue/green/magenta regions), within which the rigidity is encouraged through *rigidity loss* (blue/green/magenta springs). This rigid loss constructs springs among the arrow end-points in each source rigid-clusters, which prevents the flow from deforming the rigid cluster. The color of the springs matches the color of the cluster.

We also enable outlier rejection through spectral clustering as proposed in [119]. In terms of the mechanical machine analogy, the intuition of outlier rejection is that it adjusts the stiffness of springs in order to weaken the springs that connect outlier points (i.e. the points, the motion of which is non-rigid with respect to the rest of the cluster). To further regularize the flow, a structural regularization through Neural prior [63, 119, 64, 20] has also been proposed in recent works. We identified that the Neural prior regularizer helps mostly with static scenes flow estimation with a small number of dynamic objects; however, it over-regularizes the flow in highly dynamic scenes due to the limited capacity of the Neural-prior network. Therefore, we decided to drop this regularization in our proposed approach.

The main novelty of our approach stems from replacing the *large spatially compact non-overlapping clusters* delivered by an independent clustering algorithm, such as DBSCAN [27], with two different types of clusters: (i) *soft mutually overlapping clusters*, which are assumed to spread over multiple rigid objects and (ii) *hard non-overlapping rigid clusters*, which are expected to cover only the points

from a single rigid object or its part. The soft clusters are optimized by the rigidity loss with outlier rejection, and the hard clusters are optimized by the rigidity loss without the outlier rejection. While soft clusters remain fixed, the hard clusters are progressively growing towards consistently moving points in the spatial and temporal domain.

We argue that while the flow estimated from large non-overlapping clusters [119] heavily suffers from over and under-segmentation, usage of the proposed overlapping growing clusters significantly suppresses this issue, see Figure 5.7. In particular, it mitigates the *over-segmentation* by spatially propagating the rigidity through the overlap; see Figure 5.8, for example, that the blue and magenta clusters deliver strong rigidity springs between the points of the left object, which consequently grows the hard rigid cluster over the whole rigid object. Mitigation of the *under-segmentation* is achieved by using only small rigid clusters combined with outlier rejection techniques; for example, even though all three soft clusters contain an outlier, the resulting rigidity springs between left and right rigid objects are weakened, and the hard rigid cluster grows only within the rigid object. Consequently, the only assumption required in order to correctly segment inconsistently moving rigid objects is that each point of the rigid object appears as an inlier in at least one soft cluster hypothesis.

5.6.1 Method Overview

Point clouds consist of 3D points $\mathbf{x}_i \in X \subset \mathbb{R}^3$ and $\mathbf{y}_i \in Y \subset \mathbb{R}^3$. Since real-world scenes consist mostly of a static background, we follow a good practice of compensating the ego-motion [20] first before running our overlapping cluster hypotheses. We calculate the ICP [121] as an estimate of ego-motion and use it to transform the source point cloud X. In the rest of this section, we focus on the estimation of the remaining flow, which corresponds to the motion between transformed source point cloud X and target point cloud Y. Consequently, the flow of point $\mathbf{x}_i \in X$ is a 3D vector $\mathbf{f}_i \in F \subset \mathbb{R}^3$ that is non-zero only on dynamic objects if the ego-motion is correctly estimated.

The method is summarized by pseudo-code in Algorithm 2. The proposed method first initialize two sets of clusters: hard and soft. *Hard* rigid clusters are non-overlapping small clusters that are assumed to cover only a single rigid object or its part. Each rigid cluster $H \in \mathcal{H}$ is a small compact cluster delivered through spatial-temporal segmentation on X including the adjacent temporal point clouds. In contrast, the *soft* rigid clusters are small overlapping clusters that are expected to overflow into neighboring objects. Each point $\mathbf{x} \in Y$ is associated with the one soft cluster $S \in \mathcal{S}$, which is defined as the set of its k nearest neighbors $N_k(\mathbf{x}, X)$ from the point cloud X. Given these two sets of clusters, we optimize the flow to minimize (i) hard rigidity loss on hard clusters, (ii) soft rigidity loss on soft clusters and (iii) distance loss on all points. Both rigidity losses enforce rigid flow on points within the cluster. The main difference is that the soft rigidity loss allows for outlier rejection. All losses are detailed in the following paragraphs. Once the optimized flow is available, we merge rigid clusters in X, whose flow goes into the same rigid cluster in Y. Since the algorithm

is iteratively called on all consecutive pairs of point clouds, the rigidity is propagated throughout the temporal domain.

Algorithm 2 Joint flow estimation and rigid object segmentation

Require: pointcloud X, Y
Ensure: flow F

1. Initialize set of *hard* rigid clusters $\mathcal{H}$.
2. Initialize set of *soft* rigid clusters $\mathcal{S}$.
3. Optimize
$$\mathcal{L}(F) = \alpha \sum_{\mathbf{f} \in F} \mathcal{L}_{\text{dist}}(\mathbf{f}) + \beta \sum_{H \in \mathcal{H}} \sum_{\mathbf{f} \in H} \mathcal{L}_{\text{hard}}(\mathbf{f}) + \gamma \sum_{S \in \mathcal{S}} \sum_{\mathbf{f} \in S} \mathcal{L}_{\text{soft}}(\mathbf{f}),$$
where α, β and γ are hyper-parameters of the proposed method.
4. If the flow of two different rigid clusters $H_i, H_j \in \mathcal{H}$ goes into the same rigid cluster in point cloud Y, then merge H_i, H_j into the same cluster.
5. Repeat from 3 until convergence or reaching maximum number of iterations.

5.6.2 Distance loss

Similarly to existing approaches, we assume that the motion of objects is sufficiently small with respect to the spatio-temporal resolution of the sensor. This assumption transforms into so-called *distance loss*, which attracts the flow of points from the point cloud X toward the consecutive point cloud Y.

$$\mathcal{L}_{\text{dist}}(\mathbf{f}) = \sum_{i \in P} \|\mathbf{x}_i + \mathbf{f}_i - N_1(\mathbf{x}_i + \mathbf{f}_i, Y)\| \tag{5.9}$$

where $N_1(\mathbf{x}, Y)$ is the nearest neighbor of point $\mathbf{x}$ from point cloud Y. In practice, we use it bidirectionally for both point sets X, Y, which is equivalent to Chamfer distance.

5.6.3 Hard rigidity loss

Since the distance loss is typically insufficient for a reliable flow estimation, the additional prior assumption that takes into account the rigidity of objects is considered. In contrast to existing approaches [27, 37, 105, 119], we do not explicitly model a fixed number of independently moving rigid objects, but we simultaneously optimize flow with the rigid clusters as describe in Algorithm 2. Given a cluster C consisting of k points from point cloud X, we construct the complete undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with k vertices $\mathcal{V} = C$ and $k(k-1)/2$ edges $\mathcal{E}$ corresponding to all possible pair-wise connections among points (without self-loops). Each edge is associated with a reward function that encourages the flow in incident vertices to preserve the mutual distance between the corresponding points, *i.e.*, encourage the rigid motion. The reward for preserving rigidity is defined as follows:

$$r_{ij} = 1 - \sum_u \frac{(d_{ij}^u - \hat{d}_{ij}^u)^2}{\theta}, \tag{5.10}$$

where $d_{ij}^u = |\mathbf{x}_i^u - \mathbf{x}_j^u|$ is the distance between the u-th dimension of points $\mathbf{x}_i, \mathbf{x}_j \in C$ and similarly $\hat{d}_{ij}^u = |(\mathbf{x}_i^u + \mathbf{f}_i^u) - (\mathbf{x}_j^u + \mathbf{f}_j^u)|$ is the distance after applying the estimated flow on these points, and θ is a hyper-parameter of the proposed method. If the motion is rigid, the distance difference is zero, and the reward equals one; if the distance is non-zero the reward is proportionally smaller. Given this notation, we introduce *hard rigidity loss*

$$\mathcal{L}_{\text{hard}}(\mathbf{f}) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{V}} -\log(r_{ij}). \tag{5.11}$$

5.6.4 Soft rigidity loss

Intuitively, when the cluster C is incorrect (e.g. it overflows into neighboring objects), the optimization of the flow through the hard rigid loss delivers inaccurate flow. In order to enable outlier rejection, we introduce k-dimensional soft clustering vector $\mathbf{v} \in [0,1]^{(k+1)}, \|\mathbf{v}\| = 1$ that is supposed to softly selects high-reward edges. This vector models how much the points are likely to be in the rigid object that is dominant within the cluster. Given this notation, we define the soft rigidity score induced by the point $\mathbf{x}_m$ as

$$s_m(\mathbf{v}) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{V}} r_{ij} \mathbf{v}_i \mathbf{v}_j = \mathbf{v}^\top \mathbf{A}_m \mathbf{v}, \tag{5.12}$$

where matrix $\mathbf{A}_m$ consists of elements $[\mathbf{A}_m]_{i,j} = r_{ij}$. Product $\mathbf{v}_i \mathbf{v}_j$ corresponds to the spring stiffness between point $\mathbf{x}_i$ and $\mathbf{x}_j$ in the mechanical analogy from Figure 5.7.

Given the score matrix $\mathbf{A}_m$, the optimal soft clustering is, by Raleigh's ratio theorem, the principal eigenvector of matrix $\mathbf{A}_m$

$$\text{eig}(\mathbf{A}_m) = \arg\max_{\|\mathbf{v}\|=1} \mathbf{v}^\top \mathbf{A}_m \mathbf{v}. \tag{5.13}$$

The score (5.12) for the optimal soft clustering (which is equal to the principal eigenvalue of $\mathbf{A}_m$) describes how much the flow is consistent with the rigid motion. In order to make the estimated flow more rigid under the optimal soft clustering, we introduce the *soft rigidity loss*

$$\mathcal{L}_{\text{soft}}(\mathbf{f}) = \sum_m -\log\Big(s_m\big(\text{eig}(\mathbf{A}_m(\mathbf{f}))\big)\Big), \tag{5.14}$$

which pushes the principal eigenvalue up and consequently makes the flow on spatially compact clusters more rigid.

Table 5.5: Performance on Argoverse2 dataset in foreground aware metrics, i.e. Threeway EPE metric.

		Dynamic Foreground			**Static Foreground**			**Static Background**		
Methods	*Avg. EPE* [m]↓	*EPE* [m]↓	*AS* [%]↑	*AR* [%]↑	*EPE* [m]↓	*AS* [%]↑	*AR* [%]↑	*EPE* [m]↓	*AS* [%]↑	*AR* [%]↑
MBNSFP [119]	0.159	0.393	9.325	25.72	**0.034**	**88.54**	**96.98**	0.051	84.96	92.59
NSFP [63]	0.083	0.141	39.85	71.69	0.059	75.15	91.14	0.048	81.96	93.56
Chodosh [20]	0.070	0.132	41.80	75.49	0.049	77.29	93.74	0.028	87.97	95.30
Ours	**0.047**	**0.079**	**67.90**	**85.35**	0.035	86.26	95.78	**0.026**	**93.02**	**96.30**

Table 5.6: Performance on Argoverse2 dataset in object class aware EPE.

	Pedestrian			**Cyclist**			**Vehicles**		
Methods	*Avg.*↓	*Dyn.*↓	*Stat.*↓	*Avg.*↓	*Dyn.*↓	*Stat.*↓	*Avg.*↓	*Dyn.*↓	*Stat.*↓
MBNSFP [119]	0.071	0.115	0.026	0.302	0.570	0.034	0.245	0.457	**0.034**
NSFP [63]	0.062	0.080	0.044	0.058	0.099	0.017	0.111	0.156	0.065
Chodosh [20]	0.068	0.083	0.052	0.047	0.083	0.011	0.100	0.145	0.054
Ours	**0.031**	**0.039**	**0.023**	**0.012**	**0.016**	**0.009**	**0.068**	**0.097**	0.039

5.7 Experiments

Implementation Details. We use $k = 16$ and set α and β loss weights to 1. We use the learning rate of 0.004. The θ distance threshold for outlier rejection is set to 0.03 as in [15, 119]. We perform hard clustering by spatio-temporal DBSCAN over horizon of 5 point clouds and set epsilon parameter to 0.3 and minimal samples to 1, resulting in clustering the points to the same cluster if the Euclidian distance is lower than 0.3. The flows are optimized until a fixed number of iterations (1500) or until reaching the convergence of the loss function. For other comparative methods, we use the same optimized parameters as reported in their original papers and official implementations.

5.7.1 Comparison with State-of-the-Art Methods

We benchmark our method against the top-performing methods, such as NSFP [63], MBNSFP [119] and Chodosh [49], which all share the same structural optimization-based regularization, i.e Neural Prior [63, 64]. For all methods, we use the official implementation provided in papers. For Chodosh [20], we used the codebase published in [118]. For each method, we compensate the ego-motion with KISS-ICP [121] first, then estimate the flow by methods.

In Table 5.5, we show results on Argoverse2, the main benchmark for 3D scene flow estimation. Our method dominates the dynamic foreground metrics, while the MBNSFP [119] has better static foreground. We suggest, that their reliance on Neural prior architecture coupled with under-segmented rigidity tends to overfit on the static classes, since they share a single movement, caused by ego-motion, and the MLP layers in Neural Prior do not have expressive capacity to catch multiple motion patterns. Chodosh [20] do not optimize rigidity and Neural prior jointly, but fine-tune the rigidity as a post-processing step, resulting in more separable object motion. Our method without Neural Prior and merging clusters achieves the best average EPE over the dynamic and static

metrics.

When zooming in to per-class scene flows, we observe a trend of overfitting to larger dynamic objects, i.e. vehicles. We suggest, that fitting one structural neural prior to multiple objects in the scene leads to the local minima of solving the objects with the most of points, and sacrificing the smaller ones. In Table 5.6, we show the performance on Pedestrians, Cyclists and Vehicles separately. For example, we can see, that the performance of the other methods on Pedestrian class is halved, compared to ours, where the over-segmentation of the scene usually safely cluster the whole Pedestrian as one hard cluster and estimate rigid flow without structural neural constraints. We see that in Cyclist category, the performance of ours is even higher. We do not lose the ability estimate flows on Vehicles, as we can separate them in our merging clustering as well. We have worse static vehicles compared to MBNSFP[119], which we explain by overfitting into static flow.

5.7.2 DBSCAN Clustering in Scene Flow

In order to achieve the best possible clustering in competing methods, the authors of [119, 20] tune The DBSCAN algorithm to catch the bigger objects, which is visible in parameter epsilon (distance grouping threshold) set to 0.8 and minimal number of point samples in one density-based cluster set to 30 in MBNSFP [119]. Such parametrization allows for successful grouping of points in whole bodies of large objects such as vehicles, but simultaneously group multiple smaller objects together. Another issue stems of points, that falls below such threshold and therefore do not form the cluster and are treated without rigidity regularization in the methods. See example of such DBSCAN clustering in Figure 5.9, where the multiple pedestrians are grouped together with a tree or a vehicle in the middle of point cloud has blue points (points not assigned to any cluster).

We conclude that such clustering parametrization leads to over-fiting on large objects a therefore we designed our hard clustering approach to assign all points to clusters in a "over-segmentation" manner and gradually merge the over-segmented clusters based on flow optimization.

5.7.3 Neural Prior with Hard and Soft Clusters.

In order to compare our proposed rigidity via hard and soft clusters to the most similar methods (MBNSFP [119] as it also use spatial consistency with outlier rejection), we change the direct optimization of flows for neural prior architecture as in other methods [63, 119, 20]. We also perform experiments under the metrics proposed in MBNSFP [119], i.e. overall end point error without foreground/background split.

In Table 5.7, we see the results on Waymo [106] and Argoverse1 [11] datasets on splits used in [119]. We can see, that the better performance is reached with our rigidity regularization. The MBNSFP [119] is the second top-performing methods with regular Neural prior [63] with cycle consistency and structural regularization behind.

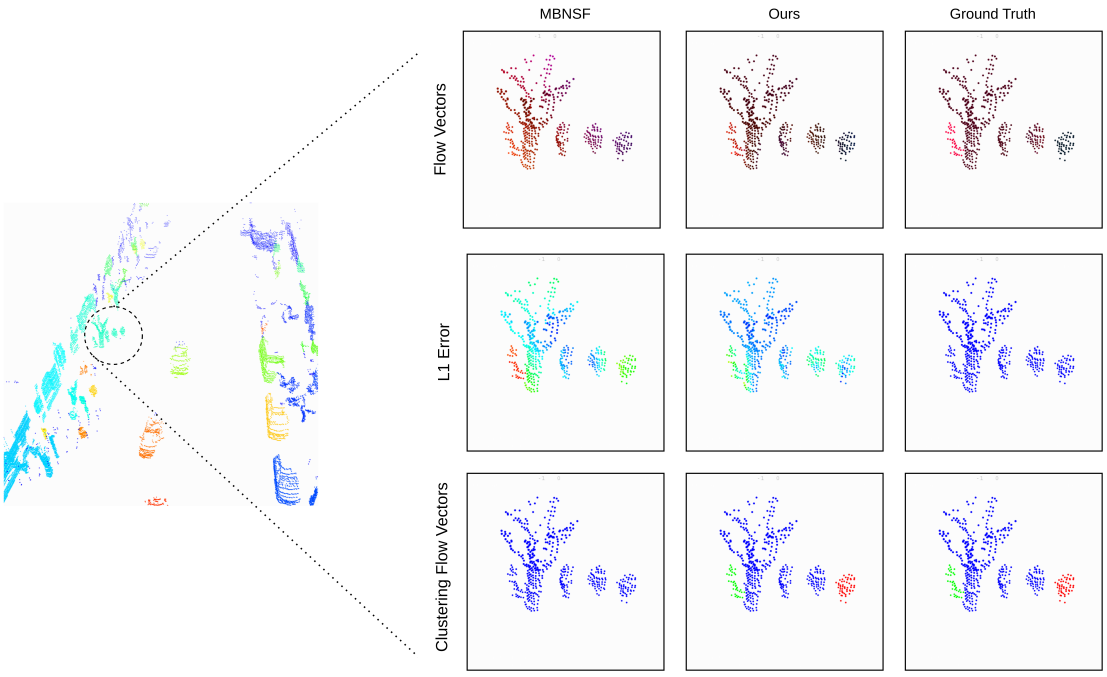


Figure 5.9: Qualitative example on Waymo Dataset. We show, that by DBSCAN under-segmentation (left part of the image, where color denotes cluster id), it is impossible to separate motions of multiple objects in one cluster. When the MBNSFP [119] method converges, the flow vectors on a single cluster looks uniform, resembling similar motion patterns on separate objects, while with our gradual clustering, we are able to separate motions and assign correct headings of flows. In the last row, we applied DBSCAN clustering to **only** flow vectors to show, that we can separate the objects based on our method output flows. We could not find parameters to separate objects with MBNSFP [119] flows, as they are too close to each other.

Table 5.7: Performance on LiDAR datasets. We show performance on standard LiDAR benchmarks with overall metrics (without dynamic/class split). Results are averages over 3 runs.

Dataset	Method	*EPE* [m]↓	*AS* [%]↑	*AR* [%]↑	θ [rad]↓
Argoverse1	JGF [81]	0.542	8.80	20.28	0.715
	PointPWC-Net [131]	0.409	9.79	29.31	0.643
	NSFP	0.065	77.89	90.68	**0.230**
	MBNSFP [119]	0.057	86.76	92.46	0.273
	NSFP + **Ours**	**0.050**	**87.06**	**94.26**	0.269
Waymo	R3DSF[37]	0.414	35.47	44.96	0.527
	NSFP[63]	0.087	78.96	89.96	0.300
	MBNSFP[119]	0.066	82.29	92.44	**0.277**
	NSFP + **Ours**	**0.039**	**88.96**	**95.65**	0.297

Runtime Benefit of Implementing Soft Clusters as Neighborhoods. We analyze the performance-time trade-off in Figure 5.10 by acquiring the *EPE* based on inference time for our method on Argoverse1 dataset while using Neural Prior architecture with proposed Soft Clusters. The speed was measured on NVIDIA Tesla V100 GPU on complete point clouds. We compare with optimization-based methods FastNSF [64] and MBNSF [119] with distance transform acceleration proposed by [64] and DBSCAN clustering, and Neural prior [63] without distance transform acceleration [64]. Compared to the MBNSFP [119], our method is approximately **10**× faster while stopping at the same prediction error. Even though their spatial consistency regularization uses the same algorithm of computing the point-to-point displacements with outlier rejection, our method is significantly faster because we design the clusters on the same-sized neighborhoods, which can be efficiently parallelized on GPUs. In contrast to our method, MBNSFP computes eigen vectors of matrix with variable-sized clusters [27]. Thus, MBNSFP is looping over c clusters $\{N_0, N_1, ..., N_c\}$ one-by-one, whereas our method allows GPU to parallel the $\mathbb{R}^{N\times k\times k}$ tensor computation for N points in point cloud and k neighbors simultaneously.

5.7.4 Results on StereoKITTI benchmark.

We also evaluate our method on both the full StereoKITTI and its testing subset KITTI_t following the experimental setting in [59, 81, 57]. We compare our method to the top-performing methods in self-supervised 3D scene flow and also to recent fully-supervised methods trained on FT3D [32] dataset. Our self-supervised method without any training data is on par with the most recent and top performing fully-supervised method, the IHNet [127] in terms of *AS* (Ours 98.0% and fully-supervised IHNet 97.8%), see Figure 5.11.

Our main baselines for performance comparison are state of the art self-supervised methods such as SCOOP [59], SLIM [2], FastNSF [64], Rigid Flow [62]. Our method achieves the new self-supervised state-of-the-art performance on stereoKITTI benchmark with the relative improvement

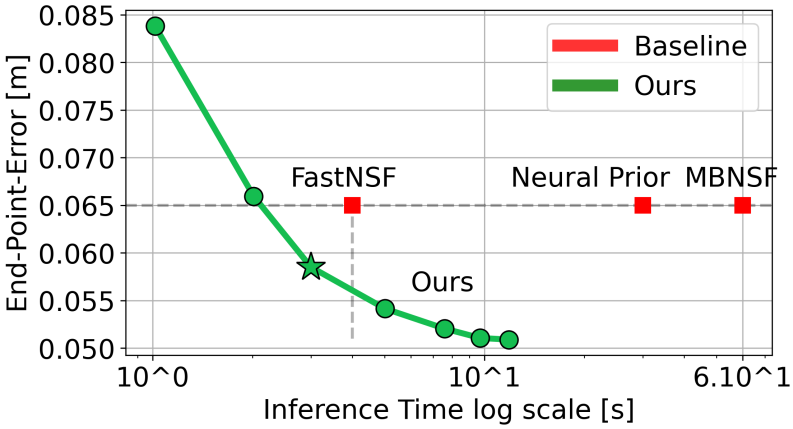


Figure 5.10: Flow estimation error vs. run time on Argoverse1 dataset. Our method in green offers speed-accuracy trade-off that outperforms the converged baselines on overall *EPE* metric.

of 38% of *EPE* compared to top-performing SCOOP [59] on StereoKITTI and also achieves the improvement of 42% of *EPE* compared to the top-performing NSFP [64] on KITTI_t.

5.7.5 Ablation Study

Method components In Table 5.9, we show metrics when adding one enhancement per time. We observe the benefits of components on final performance on the Argoverse2 dataset. Without rigidity regularization and clusters **a)**, the methods resembles flow only to the nearest neighbors in target point cloud and therefore fails to produce meaningful motions. By including the hard cluster rigidity **b)**, we observe physically consistent motions. However, since we enforce rigidity on over-segmented clusters, the objects still deforms. With addition of soft clusters **c)**, we are able to connect nearby points from different hard segments to rigidity term and reject outliers, expanding the object rigidity. By guiding the hard clusterization with flows **d)**, we acquire small boost by connecting lonely clusters divided by occlusions or separate by LiDAR sampling rate back to the major rigid body.

What really matters for rigidity regularization In Table 5.10, we list the configuration of hard clustering and soft neighborhoods (kNN) with researched rigidity regularizations on KITTI_t data. KNN rigid neighborhoods provide the best results with both regularizations. When SCOOPs regularization is applied to the hard DB instances, the performance diminishes compared to our kNNs.

In this matter, our soft rigidity clusters constructed from neighborhoods are generalizing to the out-of-distribution datasets and does not require careful parameter manipulation as density-based clustering.

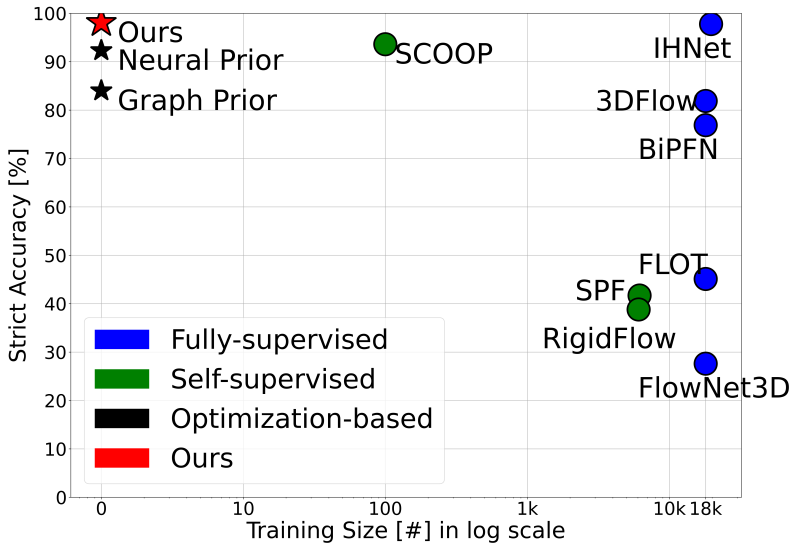


Figure 5.11: Comparison of the proposed method with self-supervised [59, 62, 61, 63, 84] and fully-supervised [70, 85, 127] competitors on StereoKITTI benchmark.

Table 5.8: **Comparison with State-of-the-Art on KITTI dataset.** We evaluate scene flow based on standard metrics *EPE*, *AS*, *AR* and *Out.* for StereoKITTI dataset. Previous state of the art performance is underlined and current best is in bold. Our results averaged over 5 runs.

Test data	Method	$EPE(m)\downarrow$	$AS(\%)\uparrow$	$AR(\%)\uparrow$	$Out.(\%)\downarrow$
$KITTI_o$	FlowStep3D [57]	0.102	70.8	83.9	24.6
	RigidFlow [62]	0.102	48.4	75.6	44.2
	SLIM [2]	0.067	77.0	93.4	24.9
	MBNSFP [119]	0.112	80.7	86.3	14.5
	NSFP [64]	0.051	89.8	95.1	14.3
	SCOOP [59]	<u>0.047</u>	<u>91.3</u>	<u>95.0</u>	<u>18.6</u>
	Ours	**0.029**	**98.0**	**98.7**	**11.7**
$KITTI_t$	JGF [81]	0.105	46.5	79.4	-
	RigidFlow [62]	0.117	38.8	69.7	-
	MBNSFP [119]	0.112	81.6	87.3	15.0
	SCOOP [59]	0.039	<u>93.6</u>	<u>96.5</u>	15.2
	NSFP [64]	<u>0.036</u>	92.3	96.2	<u>13.2</u>
	Ours	**0.021**	**98.7**	**99.1**	**11.0**

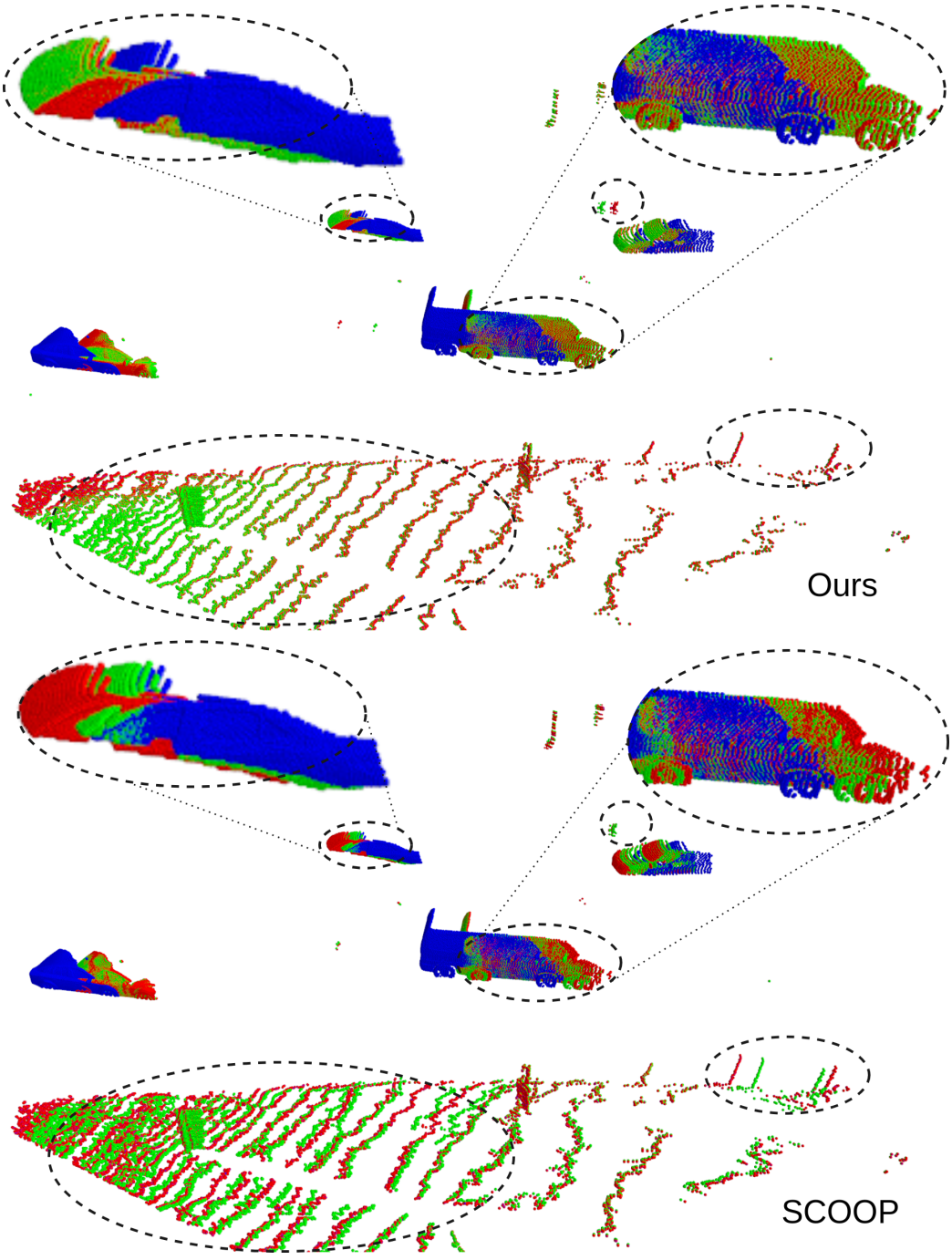


Figure 5.12: Visual comparison of scene flow estimation of our method (left) with the SCOOP [59] (right). The data sample is taken from the StereoKITTI data set. The blue points denote the clouds X, while the red points are clouds Y. The green points correspond to warped cloud $X + F$, where F is the estimated flow from prediction models. SCOOP exhibits a deviation from the target, whereas our method maintains the structural integrity of the source point cloud and rigidity of objects.

Table 5.9: Ablation study for individual method components on Argoverse2.

Module	Avg. $EPE(m)\downarrow$	Dyn. Fore. $EPE(m)\downarrow$	Stat. Fore. $EPE(m)\downarrow$	Stat. Back. $EPE(m)\downarrow$
a) w/o Rigidity	0.244	0.504	0.103	0.124
b) w/ Hard Clusters Rigidity	0.052	0.089	0.040	0.028
c) w/ Hard Clusters and Soft Clusters Rigidity	0.047	0.079	**0.035**	0.026
d) w/ Rigidity and Flow Guided Hard Clusters	**0.046**	**0.077**	**0.035**	**0.025**

Table 5.10: Rigidity regularization with different clusters tested on KITTI_t.

Clusters	Regularization	$EPE(m)\downarrow$	$AS(\%)\uparrow$
N/A	N/A	0.036	92.3
DB[37, 120]	MBNSF[120]	0.112	81.6
DB[37, 120]	SCOOP[59]	0.035	96.3
kNN	SCOOP[59]	0.024	96.7
kNN	**Ours**	**0.021**	**98.7**

5.8 Discussion

Conclusion. We presented a novel self-supervised method for 3D scene flow prediction inspired by rigid motion of multiple objects in the scene. The method introduced overlapping cluster rigidity regularization, that achieve state-of-the-art on standard benchmarks and is tuned towards the complex dynamic objects rather then static backgrounds. The framework is part of the optimization-based models family a therefore does not require any training data. Extensive experiments showed, that our method surpass self-supervised state-of-the-art models on scene flow benchmarks with unique sensor configurations, and also is on par with fully-supervised counterparts. The proposed solution cleverly adapts rigidity mechanism to discover long tail object flows showing that it is important, *how* are the points paired into the regularization mechanism.

Limitations. Even though our method is reasonably fast, it still does not, as the other methods, achieve real-time performance while keeping high accuracy. Potential direction for improvement is shown in [118]. Secondly, our method does not deal with the local minima of the distance loss, resulting in sub-optimal flows even when the object is correctly clustered. These two issues are worth of the future research to improve scene flow further. One way to achieve real-time performance for our method, while maintaining accuracy, is via distillation to neural network [118]. Secondly, our method pairs the rigid points by geometrical neighborhoods, which can transfer rigidity properties to larger area on sparse point clouds. Thus, enforcing flows on multiple paired dynamic objects to be the same. In other words, the sparse measurements might limit the performance gain of higher values of parameter k.

Future Work. One way how to compensate in sparse measurement scenarios is to include the points from other frames along the temporal domain. An other potential improvement is the

introduction of structural prior for rigid cluster detection, i.e. the instance segmentation neural network.

Chapter 6

Conclusion

6.1 Conclusion

Data-driven 3D perception solutions require scalability and generalization to keeping the low cost during the development [116, 47, 86]. The field of 3D computer vision is evolving rapidly with better and more efficient algorithms, but the aforementioned problems still exists as the fully-supervised methods are practically intractable and the unsupervised methods did not yet surpass the fully-supervised in terms of performance [26, 105, 86].

We have contributed to the issues by proposing more realistic simulator [116] and data augmentation framework [149] as it brings models closer to the desired performance and ease the annotation costs. The thesis further focused on leveraging motion and temporal domain to improve the scalability of semi-supervised [35] and self-supervised [114, 115] models as a general source of supervision signals. The contributions in scene flow methods raised the bar in self-supervised state-of-the-art models, that can support other tasks in 3D perception and robotics [114, 115].

This work has built upon general paradigms, that are used even in the recent state-of-the-art methods [74, 47, 106, 7, 11] and forecast the importance of temporal and motion domain in future models for 3D perception. As the field rapidly evolves, some methods become deprecated, but the simulated and augmented environments, pseudo-labeling, temporal consistency, propagation along the time is observable even in the most recent research advancements and foundational vision models [56, 74]. With the emerging transformer-based architectures [56, 74, 75] and vision foundational models [148, 56], we still need to scale the supervision. The researched paradigms in the thesis contribute to making the 3D perception models adapt and run on practical application.

6.2 Future Work

Despite the contributions in related tasks, the thesis did not establish the unified framework of each individual novelties to build upon each other. On Figure 6.1, we propose a direction of combining the proposed self-supervision losses from scene flow to improve the tasks of object detection and semantic segmentation on sequences of point clouds. Based on our findings, the motion vectors can discover objects (as shown in Figure 5.9 and accumulated points together with trajectory and shape of the object provides additional features to supervision signal. By accumulating such instances, one can apply contrastive framework [132, 12] to learn discriminative features in shared backbone of multi-task architecture [75]. Such approach will lead to more scalable models, which would require less annotation of dynamic objects, their semantics and geometrical properties. As a result, the general concept of motion can improve the 3D perception pipelines using unlabeled data.

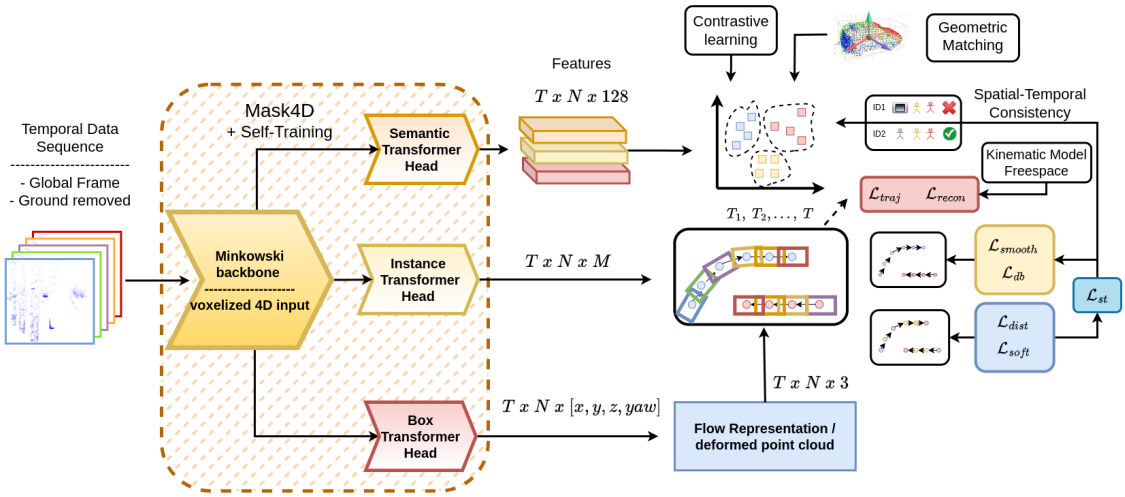


Figure 6.1: **Possible future direction of leveraging motion for 3D perception tasks**

Appendix A

Additional Contributions

A.1 Successfully supervised student theses:

- Improving Detection by Exploiting Dynamics in the Lidar Data; Vojtech Bartek
- Convolutional Neural Networks for Object Classification from LiDAR Data; Jiri Zacha
- Learning the Musical Instrument Amplifier Model with Neural Networks; Jakub Lukes
- Ensemble Detection Models for LiDAR Point Clouds; Simon Pokorny
- Motion Feature Self-Supervised Learning in 3D Point Cloud Data; Simon Pokorny

A.2 Lab tutor of bachelor courses:

- Vision for Robotics - award for outstanding tutor
- Robot Learning

A.3 Collaboration with Industry

- Research with Valeo.ai
- Application of proposed methods with Valeo production team

Appendix B

Author Publication Response

Patrik Vacek, Otakar Jašek, Karel Zimmermann, and Tomáš Svoboda. Learning to predict lidar intensities. *IEEE Transactions on Intelligent Transportation Systems*, 23(4):3556–3564, 2022

- Liu, Y., Shen, Y., Tian, Y., Ai, Y., Tian, B., Wu, E. and Chen, L., 2022. Radarverses in metaverses: A CPSI-based architecture for 6S radar systems in CPSS. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 53(4), pp.2128-2137.
- Liu, Y., Shen, Y., Fan, L., Tian, Y., Ai, Y., Tian, B., Liu, Z. and Wang, F.Y., 2022. Parallel radars: from digital twins to digital intelligence for smart radar systems. Sensors, 22(24), p.9930.
- Guillard, B., Vemprala, S., Gupta, J.K., Miksik, O., Vineet, V., Fua, P. and Kapoor, A., 2022, October. Learning to simulate realistic LiDARs. In IROS. IEEE.
- Korus, K., Czerniawski, T. and Salamak, M., 2023. Visual programming simulator for producing realistic labeled point clouds from digital infrastructure models. Automation in Construction, 156, p.105126.
- Christian, G., Woodlief, T. and Elbaum, S., 2023, May. Generating Realistic and Diverse Tests for LiDAR-Based Perception Systems. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2604-2616). IEEE.
- Haghighi, H., Wang, X., Jing, H. and Dianati, M., 2024. Review of the Learning-based Camera and Lidar Simulation Methods for Autonomous Driving Systems. arXiv preprint arXiv:2402.10079.

- Li, B., Li, J., Chen, G., Wu, H. and Huang, K., 2022, May. De-snowing LiDAR point clouds with intensity and spatial-temporal features. In ICRA (pp. 2359-2365). IEEE.
- Wang, H., Liang, H., Li, Z., Zheng, X., Xu, H., Zhou, P. and Kong, B., 2024. InLIOM: Tightly-Coupled Intensity LiDAR Inertial Odometry and Mapping. IEEE Transactions on Intelligent Transportation Systems.
- Hanzálek, Z., Záhora, J. and Sojka, M., 2023. Cone Slalom with Automated Sports Car–Trajectory Planning Algorithm. IEEE Transactions on Vehicular Technology.
- Chen, H. and Müller, S., 2022, June. Analysis of real-time lidar sensor simulation for testing automated driving functions on a vehicle-in-the-loop testbench. In 2022 IEEE Intelligent Vehicles Symposium (IV) (pp. 1605-1614). IEEE.
- Haghighi, H., Dianati, M., Debattista, K. and Donzella, V., 2023. Contrastive Learning-Based Framework for Sim-to-Real Mapping of Lidar Point Clouds in Autonomous Driving Systems. arXiv preprint arXiv:2312.15817.
- Yang, D., Liu, Z., Jiang, W., Yan, G., Gao, X., Shi, B., Liu, S. and Cai, X., 2023. Realistic Rainy Weather Simulation for LiDARs in CARLA Simulator. arXiv preprint arXiv:2312.12772.
- Yang, X., Zhang, Y., Wang, Y., Dai, K., Qin, W. and Yin, C., 2023, September. High-fidelity LiDAR Simulation System Based on Real Pointcloud Features. In 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC) (pp. 656-662). IEEE.
- Rubel, R., Clark, N. and Dudash, A., 2023. SurfaceAug: Closing the Gap in Multimodal Ground Truth Sampling. arXiv preprint arXiv:2312.03808.
- Ochs, S., Percin, T., Samuelis, C., Schörner, P., Zofka, M.R. and Zöllner, J.M., 2023, September. What can we Learn from Virtual Sensor Models for Self Localization and Mapping for Autonomous Mobile Systems?. In 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC) (pp. 820-826). IEEE.
- Sánchez-Morales, D., Bochkati, M., Schütz, A., Zhao, S. and Pany, T., 2021, September. 3D LiDAR-IMU Integration for State Estimation and Verification Using a GNSS/INS/LiDAR Simulation Chain. In Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021) (pp. 2602-2620).
- Marez, D., Nans, L. and Borden, S., 2021, April. Multi-modal dataset generation using domain randomization for object detection. In Geospatial Informatics XI (Vol. 11733, pp. 67-80). SPIE.

- Zhang, W., Kiran, B.R., Gauthier, T., Mazouz, Y. and Steger, T., 2022. Simulation-to-Reality domain adaptation for offline 3D object annotation on pointclouds with correlation alignment. arXiv preprint arXiv:2202.02666.

Petr Šebek, Šimon Pokorný, Patrik Vacek, and Tomáš Svoboda. Real3d-Aug: Point Cloud Augmentation by Placing Real Objects with Occlusion Handling for 3D Detection and Segmentation. In *Computer Vision Winter Workshop (CVWW)*, 2023

- Kharroubi, A., Poux, F., Ballouch, Z., Hajji, R. and Billen, R., 2022. Three dimensional change detection using point clouds: A review. Geomatics, 2(4), pp.457-485.
- Hu, X., Duan, Z., Huang, X., Xu, Z., Ming, D. and Ma, J., 2023, October. Context-aware data augmentation for lidar 3d object detection. In 2023 IEEE International Conference on Image Processing (ICIP) (pp. 11-15). IEEE.
- Reuse, M., Amende, K., Simon, M. and Sick, B., 2024, February. Exploring 3D Object Detection for Autonomous Factory Driving: Advanced Research on Handling Limited Annotations with Ground Truth Sampling Augmentation. In Computer Sciences & Mathematics Forum (Vol. 9, No. 1, p. 5). MDPI.
- Lee, J., Lee, J.H., Lee, J.K., Kim, J.S., Kwon, S. and Kim, S., 2023, October. Dual Adaptive Data Augmentation for 3D Object Detection. In 2023 14th International Conference on Information and Communication Technology Convergence (ICTC) (pp. 1732-1737). IEEE.

Awet Haileslassie Gebrehiwot, Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Teachers in Concordance for Pseudo-Labeling of 3D Sequential Data. *IEEE Robotics and Automation Letters (R-AL)*, 8(2):536–543, 2023

- Kong, L., Ren, J., Pan, L. and Liu, Z., 2023. Lasermix for semi-supervised lidar semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 21705-21715).
- Gebrehiwot, A.H., Hurych, D., Zimmermann, K., Pérez, P. and Svoboda, T., 2023, October. T-UDA: Temporal Unsupervised Domain Adaptation in Sequential Point Clouds. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 7643-7650). IEEE.

Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Regularizing Self-supervised 3D Scene Flows with Surface Awareness and Cyclic Consistency. *Submitted to peer-reviewed Journal*, 2024

- Zhang, Y., Wandt, B., Magnusson, M. and Felsberg, M., 2024. DiffSF: Diffusion Models for Scene Flow Estimation. arXiv preprint arXiv:2403.05327.

Bibliography

[1] Sérgio Agostinho, Aljosa Osep, Alessio Del Bue, and Laura Leal-Taixé. (Just) A spoonful of refinements helps the registration error go down. In *ICCV*, pages 6108–6117, 2021.

[2] Stefan Baur, David Emmerichs, Frank Moosmann, Peter Pinggera, Bjorn Ommer, and Andreas Geiger. SLIM: Self-Supervised LiDAR Scene Flow and Motion Segmentation. In *ICCV*, pages 13126–13136, 2021.

[3] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. PointFlowNet: Learning representations for rigid motion estimation from point clouds. In *CVPR*, pages 7962–7971, 2019.

[4] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, 2019.

[5] M. Berman, A. R. Triki, and M. B. Blaschko. The Lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, 2018.

[6] Igor Bogoslavskyi and Cyrill Stachniss. Efficient Online Segmentation for Sparse 3D Laser Scans. *Photogrammetrie - Fernerkundung - Geoinformation*, 85:41–52, 12 2016.

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[8] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. Fast LIDAR-based road detection using fully convolutional neural networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1019–1024. IEEE, 2017.

[9] Ángela Casado-García and Jónathan Heras. Ensemble methods for object detection. In *European Conference on Artificial Intelligence (ECAI)*, 2020.

[10] Nino Cauli and Diego Reforgiato Recupero. Survey on videos data augmentation for deep learning models. *Future Internet*, 14(3), 2022.

[11] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3D tracking and forecasting with rich maps. In *CVPR*, pages 8748–8757, 2019.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[13] X. Chen, B. Mersch, L. Nunes, R. Marcuzzi, I. Vizzo, J. Behley, and C. Stachniss. Automatic Labeling to Generate Training Data for Online LiDAR-Based Moving Object Segmentation. *IEEE Robotics and Automation Letters (RA-L)*, page 6107–6114, 2022.

[14] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D object proposals using stereo imagery for accurate object class detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(5):1259–1272, 2017.

[15] Zhi Chen, Kun Sun, Fan Yang, and Wenbing Tao. SC^2-PCR: A second order spatial compatibility for efficient and robust point cloud registration. In *CVPR*, pages 13221–13231, 2022.

[16] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. Liu. (AF)2-S3Net: Attentive Feature Fusion With Adaptive Feature Selection for Sparse Semantic Segmentation Network. In *CVPR*, 2021.

[17] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, Quoc V. Le, Jonathon Shlens, and Dragomir Anguelov. Improving 3D Object Detection through Progressive Population Based Augmentation. In *ECCV*, 2020.

[18] Wencan Cheng and Jong Hwan Ko. Bi-PointFlowNet: Bidirectional learning for point cloud based scene flow estimation. In *ECCV*, 2022.

[19] J. H. Cho and B. Hariharan. On the efficacy of knowledge distillation. In *ICCV*, 2019.

[20] Nathaniel Chodosh, Deva Ramanan, and Simon Lucey. Re-Evaluating LiDAR Scene Flow for Autonomous Driving. *WACV*, abs/2304.02150, 2024.

[21] C. B. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019.

[22] Phuong Chu, Seoungjae Cho, Simon Fong, and Kyungeun Cho. Enhanced ground segmentation method for LiDAR point clouds in human-centric autonomous robot systems. *Human-centric Computing and Information Sciences*, 9, 12 2019.

[23] Lihe Ding, Shaocong Dong, Tingfa Xu, Xinli Xu, Jie Wang, and Jianan Li. FH-Net: A fast hierarchical network for scene flow estimation on real-world point clouds. In *ECCV*, 2022.

[24] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CORL*, pages 1–16, 2017.

[25] Khaled Elmadawi, Moemen Abdelrazek, Mohamed Elsobky, Hesham M. Eraqi, and Mohamed Zahran. End-to-end sensor modeling for LiDAR Point Cloud. *Conference on Intelligent Transportation Systems (ITSC)*, 2019.

[26] Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Pınar Topçam, Maximilian Listl, Yılmaz Kaan Çaylı, and Alois Knoll. 3D Object Detection with a Self-supervised LiDAR Scene Flow Backbone. In *ECCV*, 2022.

[27] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[28] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, June 2010.

[29] Jin Fang, Feilong Yan, Tongtong Zhao, Feihu Zhang, Dingfu Zhou, Ruigang Yang, Yu Ma, and Liang Wang. Simulating LIDAR point cloud for autonomous driving using real-world scenes and traffic flows. *CoRR*, abs/1811.07112, 2018.

[30] Jin Fang, Xinxin Zuo, Dingfu Zhou, Shengze Jin, Sen Wang, and Liangjun Zhang. LiDAR-Aug: A General Rendering-based Augmentation Framework for 3D Object Detection. In *CVPR*, pages 4708–4718, 2021.

[31] Lex Fridman, Daniel E Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Aleksandr Patsekin, Julia Kindelsberger, Li Ding, et al. Mit advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation. *IEEE Access*, 7:102021–102038, 2019.

[32] Christian Fruhwirth-Reisinger, Michael Opitz, Horst Possegger, and Horst Bischof. Fast3d: Flow-aware self-training for 3d object detectors. In *BMVC*, 2021.

[33] Alireza G. Kashani, Michael Olsen, Christopher Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15:28099–28128, 11 2015.

[34] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[35] Awet Haileslassie Gebrehiwot, Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Teachers in Concordance for Pseudo-Labeling of 3D Sequential Data. *IEEE Robotics and Automation Letters (R-AL)*, 8(2):536–543, 2023.

[36] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, pages 3354–3361, 2012.

[37] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J. Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3D scene flow. In *CVPR*, pages 5692–5703, June 2021.

[38] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. BlenSor: Blender Sensor Simulation Toolbox. In *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part II*, ISVC'11, pages 199–208, Berlin, Heidelberg, 2011. Springer-Verlag.

[39] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds. In *CVPR*, pages 3254–3263, 2019.

[40] Mokrane Hadj-Bachir and Philippe De Souza. LIDAR sensor simulation in adverse weather condition for driving assistance development. working paper or preprint, January 2019.

[41] Martin Hahner, Dengxin Dai, Alexander Liniger, and Luc Van Gool. Quantifying Data Augmentation for LiDAR based 3D Object Detection. *arXiv:2004.01643*, 1:1–7, 2020.

[42] Jun Han, Jun Tao, and Chaoli Wang. FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Trans. Vis. Comput. Graph.*, 26(4):1732–1744, 2020.

[43] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2015.

[44] Martin Friedrich Holder, Philipp Rosenberger, Felix Bert, and Hermann Winner. Data-driven Derivation of Requirements for a LiDAR Sensor Model. In *Grazer Symposium Virtuelles Fahrzeug*, Graz, May 2018.

[45] Y. Hou, X. Zhu, Y. Ma, C. C. Loy, and Y. Li. Point-to-Voxel Knowledge Distillation for LiDAR Semantic Segmentation. In *CVPR*, 2022.

[46] P. Hu, J. Ziglar, D. Held, and D. Ramanan. What You See is What You Get: Exploiting Visibility for 3D Object Detection. In *CVPR*, 2020.

[47] Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, and Konrad Schindler. Dynamic 3D scene analysis by point cloud accumulation. In *ECCV*, 2022.

[48] M. Jaritz, T. Vu, R. d. Charette, E. Wirbel, and P. Pérez. xMUDA: Cross-modal unsupervised domain adaptation for 3D semantic segmentation. In *CVPR*, 2020.

[49] Cansen Jiang, Danda Pani Paudel, David Fofi, Yohan Fougerolle, and Cédric Demonceaux. Moving Object Detection by 3D Flow Field Analysis. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 22(4):1950–1963, 2021.

[50] L. Jiang, S. Shi, Z. Tian, X. Lai, S. Liu, C. Fu, and J. Jia. Guided point contrastive learning for semi-supervised point cloud semantic segmentation. In *ICCV*, 2021.

[51] P. Jiang and S. Saripalli. LiDARNet: A Boundary-Aware Domain Adaptation Model for Point Cloud Semantic Segmentation. In *ICRA*, 2021.

[52] M. Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *ICRA*, pages 1–8, 2017.

[53] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1589–1596, 2022.

[54] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32:922–923, 1976.

[55] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[56] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, pages 4015–4026, October 2023.

[57] Yair Kittenplon, Yonina C. Eldar, and Dan Raviv. FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation. In *CVPR*, pages 4112–4121, 2021.

[58] Xingyu L., M. Yan, and J. Bohg. MeteorNet: Deep learning on dynamic 3D point cloud sequences. In *ICCV*, 2019.

[59] Itai Lang, Dror Aiger, Forrester Cole, Shai Avidan, and Michael Rubinstein. SCOOP: Self-Supervised Correspondence and Optimization-Based Scene Flow. In *CVPR*, pages 5281–5290, 2023.

[60] D. Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. In *ICML. Workshop Challenges in Representation Learning*, 2013.

[61] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-Point-Flow: Self-Supervised Scene Flow Estimation From Point Clouds With Optimal Transport and Random Walk. In *CVPR*, pages 15577–15586, 2021.

[62] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. RigidFlow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *CVPR*, pages 16959–16968, 2022.

[63] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *NeurIPS*, 34:7838–7851, 2021.

[64] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast Neural Scene Flow. In *ICCV*, pages 9878–9890, 2023.

[65] Y. Li, J. Chen, X. Xie, K. Ma, and Y. Zheng. Self-loop uncertainty: A novel pseudo-label for semi-supervised medical image segmentation. In *MICCAI*, 2020.

[66] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, pages 541–556, Cham, 2020. Springer International Publishing.

[67] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[68] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 143:39–47, 2018.

[69] I. J. Liu, J. Peng, and A. G. Schwing. Knowledge flow: Improve upon your teachers. In *ICLR*, 2019.

[70] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3D point clouds. *CVPR*, pages 529–537, 2019.

[71] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. MeteorNet: Deep learning on dynamic 3D point cloud sequences. In *ICCV*, 2019.

[72] Y. Liu, C. Ma, Z. He, C. Kuo, K. Chen, P. Zhang, B. Wu, Z. Kira, and P. Vajda. Unbiased teacher for semi-supervised object detection. In *ICLR*, 2021.

[73] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased Teacher for Semi-Supervised Object Detection. In *ICLR*, 2021.

[74] Youquan Liu, Lingdong Kong, Jun Cen, Runnan Chen, Wenwei Zhang, Liang Pan, Kai Chen, and Ziwei Liu. Segment any point cloud sequences by distilling vision foundation models. In *NeurIPS*, 2023.

[75] Rodrigo Marcuzzi, Lucas Nunes, Louis Wiesmann, Elias Marks, Jens Behley, and Cyrill Stachniss. Mask4D: End-to-End Mask-Based 4D Panoptic Segmentation for LiDAR Sequences. *IEEE Robotics and Automation Letters (R-AL)*, 8(11):7487–7494, 2023.

[76] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016.

[77] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016.

[78] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015.

[79] Andres Milioto, Ignacio Vizzo, Jens Behley, and C. Stachniss. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. *IROS*, pages 4213–4220, 2019.

[80] S. Mirzadeh, M. Farajtabar, A. Li, and H. Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. In *AAAI*, 2019.

[81] Himangi Mittal, Brian Okorn, and David Held. Just Go With the Flow: Self-Supervised Scene Flow Estimation. In *CVPR*, pages 11177–11185, June 2020.

[82] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R. Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *ECCV*, pages 424–443, 2022.

[83] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss. SegContrast: 3D point cloud feature representation learning through self-supervised segment discrimination. *IEEE Robotics and Automation Letters (R-AL)*, 2022.

[84] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *3DV*, pages 261–270, 2020.

[85] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *ECCV*, 2020.

[86] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov. Offboard 3D object detection from point cloud sequences. In *CVPR*, 2021.

[87] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NeurIPS*, 2017.

[88] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, abs/1612.00593, 2017.

[89] Matěj Račinský. 3D map estimation from a single RGB image. *M.S. thesis*, 2018.

[90] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501*, 2020.

[91] ReinventingParking. Cars are parked 95% of the time. `https://www.reinventingparking.org/2013/02/cars-are-parked-95-of-time-lets-check.html`, 2013.

[92] Yuan Ren, Siyan Zhao, and Liu Bingbing. Object insertion based data augmentation for semantic segmentation. In *ICRA*, pages 359–365. IEEE, 2022.

[93] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *ECCV*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[94] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[95] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Nader Essam. LiDAR sensor modeling and data augmentation with GANs for autonomous driving. *arXiv preprint arXiv:1905.07290*, 2019.

[96] Ahmad El Sallab, Ibrahim Sobh, Mohamed Zahran, and Mohamed Shawky. Unsupervised neural sensor models for synthetic lidar data augmentation. *NeurIPS Workshop*, 2019.

[97] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*, 2017.

[98] J. Shen, X. Tang, X. Dong, and L. Shao. Visual object tracking by hierarchical attention siamese network. *IEEE Transactions on Cybernetics*, pages 1–13, 2019.

[99] Yaqi Shen, Le Hui, Jin Xie, and Jian Yang. Self-supervised 3D scene flow estimation guided by superpoints. In *CVPR*, pages 5271–5280, 2023.

[100] Zhihao Shen, Huawei Liang, Linglong Lin, Zhiling Wang, Weixin Huang, and Jie Yu. Fast Ground Segmentation for 3D LiDAR Point Cloud Based on Jump-Convolution-Process. *Remote Sensing*, 13(16), 2021.

[101] H. Shi, G. Lin, H. Wang, T. Hung, and Z. Wang. SpSequenceNet: Semantic segmentation network on 4D point clouds. In *CVPR*, 2020.

[102] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019.

[103] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *CVPR*, June 2020.

[104] W. Shi, Y. Gong, C. Ding, Zhiheng M. Tao, and N. Zheng. Transductive semi-supervised deep learning using min-max features. In *ECCV*, 2018.

[105] Ziyang Song and Bo Yang. OGC: Unsupervised 3D object segmentation from rigid dynamics of point clouds. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

[106] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, pages 2446–2454, June 2020.

[107] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *ECCV*, pages 685–702. Springer, 2020.

[108] A. Teichman and S. Thrun. Tracking-based semi-supervised learning. *IJRR*, 31(7):804–818, 2012.

[109] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. KPconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[110] Ivan Tishchenko, Sandro Lombardi, Martin R. Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *3DV*, pages 150–159, 2020.

[111] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique Signatures of Histograms for Local Surface Description. In *ECCV*, 2010.

[112] Jiayin Tu, Ping Wang, and Fuqiang Liu. PP-RCNN: Point-Pillars Feature Set Abstraction for 3D Real-time Object Detection. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[113] F. Tung and G. Mori. Similarity-preserving knowledge distillation. In *ICCV*, 2019.

[114] Patrik Vacek, David Hurych, Karel Zimmermann, Patrick Pérez, and Tomáš Svoboda. Regularizing Self-supervised 3D Scene Flows with Surface Awareness and Cyclic Consistency. *Submitted to peer-reviewed Journal*, 2024.

[115] Patrik Vacek, David Hurych, Karel Zimmermann, and Tomas Svoboda. Let It Flow: Simultaneous Optimization of 3D Flow and Object Clustering. *Submitted to peer-reviewed Journal*, 2024.

[116] Patrik Vacek, Otakar Jašek, Karel Zimmermann, and Tomáš Svoboda. Learning to predict lidar intensities. *IEEE Transactions on Intelligent Transportation Systems*, 23(4):3556–3564, 2022.

[117] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5):544–557, 2009.

[118] Kyle Vedder, Neehar Peri, Nathaniel Chodosh, Ishan Khatri, Eric Eaton, Dinesh Jayaraman, Yang Liu, Deva Ramanan, and James Hays. ZeroFlow: Scalable Scene Flow via Distillation. *arXiv:2305.10424*, 2023.

[119] Kavisha Vidanapathirana, Shin-Fang Chng, Xueqian Li, and Simon Lucey. Multi-body neural scene flow. In *3DV*. IEEE, 2024.

[120] Kavisha Vidanapathirana, Shin-Fang Chng, Xueqian Li, and Simon Lucey. Multi-body neural scene flow. *3DV*, 2024.

[121] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[122] H. Wang, Y. Cong, O. Litany, Y. Gao, and L. J. Guibas. 3DIoUmatch: Leveraging IoU prediction for semi-supervised 3D object detection. In *CVPR*, 2021.

[123] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, pages 2598–2597, June 2018.

[124] W. Wang and J. Shen. Deep visual attention prediction. *IEEE Transactions on Image Processing, 27(5)*, pages 2368–237, 2017.

[125] W. Wang, J. Shen, and H. Ling. A deep network solution for attention and aesthetics aware photo cropping. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 1531–1544, 2018.

[126] Yuan Wang, Tianyue Shi, Peng Yun, Lei Tai, and Ming Liu. PointSeg: Real-Time Semantic Segmentation Based on 3D LiDAR Point Cloud. *CoRR*, abs/1807.06288, 2018.

[127] Yun Wang, Cheng Chi, Min Lin, and Xin Yang. IHNet: iterative hierarchical network guided by high-resolution estimated information for scene flow estimation. In *ICCV*, pages 10073–10082, October 2023.

[128] Z. Wang, H. Fu, L. Wang, L. Xiao, and B. Dai. SCNet: Subdivision Coding Network for Object Detection Based on 3D Point Cloud. *IEEE Access*, 7:120449–120462, 2019.

[129] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, and James Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In *NeurIPS Datasets and Benchmarks*, 2021.

[130] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud. In *ICRA*, 2019.

[131] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *ECCV*, pages 88–107, 2020.

[132] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany. Pointcontrast: Unsupervised pre-training for 3D point cloud understanding. In *ECCV*, 2020.

[133] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. K., and M. Tomizuka. SqueezeSegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *ECCV*, 2020.

[134] Xianfa Xu, Zhe Chen, and Fuliang Yin. CutResize: Improved data augmentation method for RGB-D Object Recognition. *IEEE Robotics and Automation Letters (R-AL)*, 7(1):183–190, 2022.

[135] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[136] Bin Yang, Ming Liang, and Raquel Urtasun. HDNET: Exploiting HD Maps for 3D Object Detection. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 29–31 Oct 2018.

[137] Jihan Yang, Shaoshuai Shi, Zhe Wang, Hongsheng Li, and Xiaojuan Qi. ST3D: Self-Training for Unsupervised Domain Adaptation on 3D Object Detection. In *CVPR*, pages 10368–10378, June 2021.

[138] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *ACM Trans. Graph.*, 37(6), dec 2018.

[139] Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *ICMR*, pages 458–464. ACM, 2018.

[140] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *CVPR*, 2020.

[141] N. Zhao, T. Chua, and G. H. Lee. SESS: Self-Ensembling Semi-Supervised 3D Object Detection. In *CVPR*, 2020.

[142] Q. Zhou, C. Yu, Z. Wang, Q. Qian, and H. Li. Instant-teaching: An end-to-end semi-supervised object detection framework. In *CVPR*, 2021.

[143] Wei Zhou, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Automated evaluation of semantic segmentation robustness for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1951–1963, 2020.

[144] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, pages 4490–4499, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[145] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin. Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. In *CVPR*, 2021.

[146] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation. In *CVPR*, pages 9934–9943, 2021.

[147] M. R. Zofka, M. Essinger, T. Fleck, R. Kohlhaas, and J. M. Zöllner. The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality LiDAR stimulation. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 151–157, May 2018.

[148] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. In *NeurIPS*, July 2023.

[149] Petr Šebek, Šimon Pokorný, Patrik Vacek, and Tomáš Svoboda. Real3d-Aug: Point Cloud Augmentation by Placing Real Objects with Occlusion Handling for 3D Detection and Segmentation. In *Computer Vision Winter Workshop (CVWW)*, 2023.