DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Flow into 4D Panoptic Segmentation

Dogac Guzelkokar, Niklas Bubeck

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Flow into 4D Panoptic Segmentation

Author: Dogac Guzelkokar, Niklas Bubeck
Supervisor: Mariia Gladkova
Submission Date: 05.09.2022

I confirm that this is my own work and I have documented all sources and material used.

Munich, 05.09.2022 Dogac Guzelkokar, Niklas Bubeck

Abstract

Autonomous driving and intelligent systems nowadays are a very complex and interdisciplinary task situated in a flexible and not well described environment. Therefore, time-dependent semantic scene understanding becomes more and more important. Panoptic segmentation aims to parse both instance and semantic segmentation in a unified manner. In this work we build up on the well known Efficient LiDAR Panoptic Segmentation but exploit scene flow estimation and aligning consecutive pointclouds to match time-dependent data. Different from other scene flow models which use RGBD data and introduce color dependency, our approach aims to solely rely on lidar-based pointclouds.

Contents

1 Introduction

Temporal scene understanding is a fundamental task for autonomous and intelligent systems. To gain a complete spatio-temporal interpretation of its surroundings, panoptic segmentation (PS) combines semantic and instance segmentation. While instances are countable, such as cars and pedestrians, semantic segmentation aims to detect amorphous regions such as vegetation and roads. Measuring precise distances, LiDAR sensors have been proven to be useful for autonomous vehicles. With the accompanying increase in point cloud data, there also have to be new methods, able to process this promising data. Thus, many lidar-based methods have been developed for different tasks of dynamic scene understanding, such as segmentation and multi-object tracking. But also point-set representation learning and the rise of large-scale datasets have pushed the research in these domains. Nevertheless, the complex point distributions of LiDAR data make it difficult to perform reliably. Other than RGBD camera point clouds, lidar-based point clouds differ greatly in point size, point correspondence, and robustness. They measure data with a 360-degree field of view and therefore have lower point density for individual objects while having an overall higher amount of points. This not only causes exploding memory requirements but also generates a greater task space for the system to operate in, which is also not represented as well as for RGBD data. Furthermore, LiDAR data gets captured unordered. Thus, a bijective mapping or 1-to-1 correspondence between point clouds does not exist, other than for camera data. This reduces performance as shown by Zuanazzi et al. [1]. Another major difference is the robustness to outliers. Bidirectional reflectance disturbances caused by the sensor physics such as attenuation or scattering effects, result in a lot of outliers, where even static points don't move regarding the sensor's motion as experienced in [2]. Our approach minimizes those occurrences by using a sampling encoder/decoder strategy, and a mixture of rigid motion transformation $\mathcal{R}_t$ and scene flow $\mathcal{F}_{t\to t+1}$ that is solely learned on lidar-based point cloud data. Panoptic segmentation is then performed by exploiting the label similarity between the first point cloud $\mathcal{P}_t$ and the transferred point cloud ${\mathcal{P}'}_{t\to t+1}$ as well as the position similarity of ${\mathcal{P}'}_{t\to t+1}$ and the second point cloud $\mathcal{P}_{t+1}$. Here, an intersection over union mapping is done to find the corresponding instance labels over time.

1.1 4D Panoptic Segmentation

Panoptic segmentation is the task of predicting semantic and instance labels of 2D or 3D input and has been proposed first by [3] as a new computer vision task. This task helps provide a better scene understanding by combining the information provided by semantic and instance segmentation. 4D panoptic segmentation, introduced by [4] is the task of panoptic

segmentation, with the additional constraint on instance ids being temporally consistent. Through 4D panoptic segmentation, it is possible to provide even better scene understanding to a temporal extent through tracking instances. This is especially beneficial for autonomous driving, as temporal scene understanding is crucial for self-driving vehicles.

1.2 Scene Flow Estimation

Scene flow represents the motion of the points between two consecutive time frames, thus, creating a 3D vector field $\mathcal{F}_{t \to t+1}$. Its application enables the transformation of points recorded in the first frame $\mathcal{P}_t$ to the second frame $\mathcal{P}_{t+1}$ and allows to capture of changes in a dynamic environment. It can be seen as the 3D case of the well-known optical flow which captures pixel movement in images. Given the flow, the second point cloud can be expressed through the first point cloud and the estimated flow $\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \mathbf{f}_i$.

2 Related Work

2.1 Scene Flow

The task of scene flow estimation was first introduced by Vedula et al. [5]. They compute it from optical flow using a linear algorithm. With the emerging rise of deep learning techniques, performance was improved drastically. A lot of these methods rely either on pointnet or pointnet++ to estimate scene flow directly from point pairs [6]. But there are also methods involving voxelizations to solve the unordered point clouds. Most methods were trained on synthetic data from the FlyingThings3D dataset, or with only a small amount of annotated real-world data. Later methods like Just Go with the Flow [7] and PointPWC [8] introduce unsupervised learning to scene flow and enable training also on unlabeled data.

2.2 Point Cloud Segmentation

Point cloud segmentation describes the task of assigning points into multiple regions of different classes. Points with the same assigned class get attributed the same properties. This helps to understand the surrounding environment and has many applications in robotics, such as intelligent vehicles and autonomous navigation. In computer vision, segmenting 2D images has been a well-known task, and popular approaches have been built up on graph clustering. For the 3D case, three important properties of a segmentation algorithm should be considered. First, the algorithm should be able to take advantage of several qualitatively different kinds of features, such as trees which will have distinguished features from cars. Second, the segmentation algorithm should be able to infer the label of points that lie in sparsely sampled regions based on the information of their neighbors. Third, the segmentation algorithm should adapt to the particular 3D scanner used, because different laser scanners produce qualitatively different point cloud data, and they may have different properties even with the same scene.

With influential and fast-growing deep learning techniques, especially for pattern recognition and computer vision, handcrafted feature methods are replaced. Due to the point clouds being unordered, standard convolutions cannot be applied to LiDAR-based point clouds transforming raw point clouds essential. Depending on the data format, one differentiates between three categories. *Multiview based methods:* counts towards the early solutions. As the name suggests, the 3D data is represented by multiple 2D images, which are then processed by 2D Convolutional Neural Networks (CNN). Later on, the results can be restored into 3D space. Shortcomings are, that they introduce numerous limitations and a loss of geometric structure as they only approximate the 3D space. Furthermore, to be representative they need

proper viewpoints for the projection [9][10]. *Voxel based methods:* solve both unordered and unstructured problems of the raw point cloud. Voxelized data can then simply be further processed with 3D convolutions. Still, compared to the raw point cloud, the voxel structure is only a low-resolution form that goes along with a loss in data representation [11][12][13]. *Point based methods:* Unlike separated pre-transformation operations, in these approaches it is bound with the neural network architecture pioneering frameworks for this task are pointnet [14] and pointnet++ [15] which are elaborated in detail later.

2.2.1 Panoptic Segmentation

The panoptic segmentation task builds up on the semantic segmentation task. Here, points belonging to "things" classes such as vehicles, pedestrians, etc. are also assigned an instance id. The information of distinguishing different "things" from each other is crucial for autonomous driving, as each instance has different attributes, such as having a different motion. Mostly two of the panoptic segmentation works were of interest to us.*DS-Net:* applies cylinder convolution, then predicts for semantic labels and regressed centers separately. A multi-layer perceptron learns to shift the regressed centers to true instance centers and instance ids are assigned. Afterward, the semantic labels of instance ids are determined through majority voting. *EfficientLPS:* is another competitive work that has lower computational requirements thanks to transforming the problem into the 2D domain. After unfolding the input LiDAR scan and obtaining 2D data, it is fed through a backbone module, which is then continued by a semantic head and an instance head separately. The final panoptic segmentation result is obtained by the panoptic fusion module, which is trained to refine instance boundaries. Later on, the resulting point cloud is obtained through the KNN algorithm.

2.2.2 4D Panoptic Segmentation

4D panoptic segmentation has the additional constraint that the same instance across different scans must have the same instance id. There have been different approaches to ensure instance id consistency across scans. *4D-PLS:* is the first work on 4D panoptic segmentation. After sampling points across different scans, its encoder-decoder network learns the statistical distribution parameters for the points, assigning a probability value to points that how likely they are to belong to an instance. As it predicts for points from different scans at the same time, produced instance ids are also consistent. *4D DS-Net:* is the extension of DS-Net to the 4D panoptic segmentation task. Simply aligning and overlapping two consecutive scans with rigid transformation, DS-Net predicts labels for the combined point cloud. *Contrastive Instance Association:* is another work that is based on panoptic segmentation on 3D scans. In a contrastive learning setting, it represents the same instance appearances across different scans as positive samples and all other instances as negative samples. Consequently, instances in different scans which have similar embeddings are assigned the same instance id.

3 Preliminaries

3.1 PointNet

PointNet [14] is a deep-learning architecture that consumes a raw point cloud (set of points) without voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient, and effective approach for a number of 3D recognition tasks. The general idea of pointnet is to approximate a function f defined on a point set, that acts invariant to this point set to solve the problem of unordered points. Mathematically speaking this is the case for symmetric functions. Thus, pointnet constructs a family of symmetric functions by a neural network. Given that g is symmetric, then also f is symmetric. The smaller network h transforms each point identically and independently to a higher embedding space. Point embeddings are then aggregated by the symmetric function g and post-transformed by the function γ.

$$f\left(\{x_1, \ldots, x_n\}\right) \approx g\left(h\left(x_1\right), \ldots, h\left(x_n\right)\right)$$

Furthermore, it was shown that any symmetric function that is continuous can be constructed by pointnet. Thus, pointnet can also be described as a universal continuous set function approximator. For the function, h and γ simple multi-layer perceptrons (MLP) are used, and a max pooling layer for the embedding transformation g.

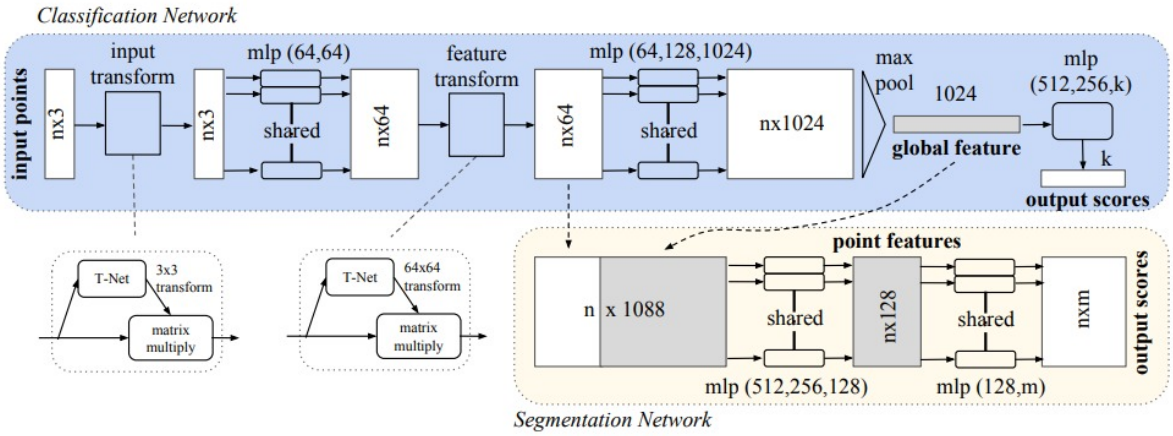

Figure 3.1: The architecture of the pointnet neural network.

To make the network more robust to geometric invariance, input point clouds are aligned to a canonical space by applying an affine transformation. This transformation is predicted

by a mini-pointnet called T-Net which gets trained end-to-end. Similarly, this is used to align points in an embedding space. To prevent local minima, the transformation is kept close to an orthogonal matrix.

3.2 PointNet++

While the first Pointnet architecture uses a single max pooling operation, PointNet++ builds a hierarchical grouping of points to abstract local regions of different sizes. This is done by set abstraction levels, which produce a new set with fewer elements and smaller resolutions. First, a sampling layer selects a set of points from its input, defining the centroids of local regions. Then the grouping layer constructs local region sets, which are then transformed into the embedding space using a mini-PointNet layer as in the original version.

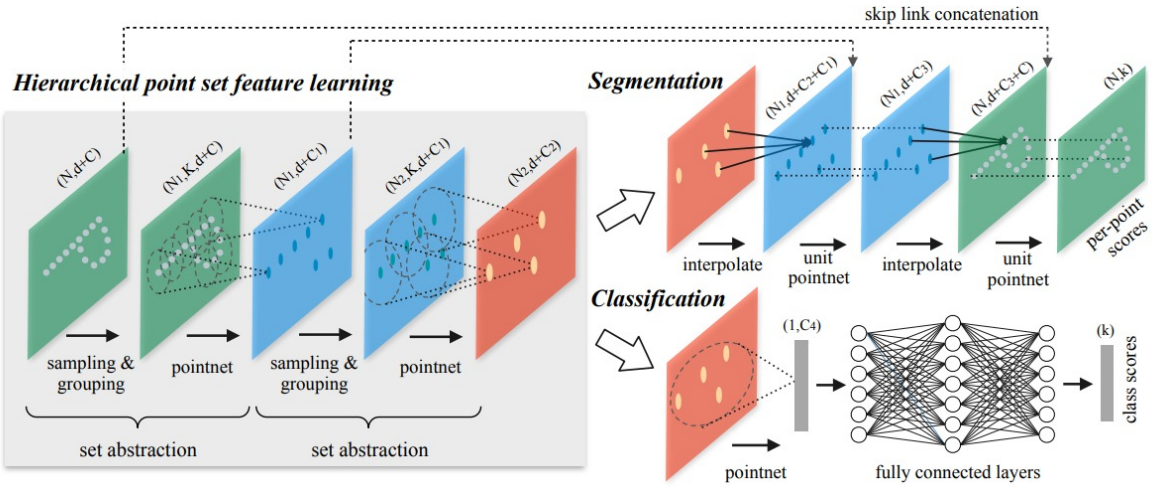

Figure 3.2: The architecture of the pointnet++ neural network.

This architecture improves robust feature learning under non-uniform sampling densities. Both PointNet and PointNet++ architectures are often used to extract global features which are then processed further.

3.3 Datasets

3.3.1 KITTI Vision Benchmark Suite

KITTI [16] is a real-world self-driving dataset. There are 150 scenes of LIDAR data in KITTI collected using seven scans of a Velodyne 64 LIDAR, augmented using 3D models, and annotated with ground truth scene flow for the camera field of view. As it is one of the first outdoor datasets with ground truth data it has been used a lot to measure the transition from indoor scene flow to outdoor.

3.3.2 FlyingThings3D

The main part of the FlyingThings3D (FT3D) collection consists of objects flying along randomized trajectories. They rely on randomness and a large pool of rendering assets to generate loads of data without running a risk of repetition or saturation [17]. The motivation for creating this dataset is to facilitate the training of large convolutional networks, which should benefit from the large variety. The data is constructed virtually and mostly used for disparity, optical flow, and scene flow estimation tasks.

3.3.3 SemanticKITTI

SemanticKITTI [18] is based on the KITTI Vision Benchmark and additionally provides pointwise annotated labels for all sequences of the Odometry Benchmark. The data was recorded at a rate of 10 Hz. Data shows inner city traffic, residential areas, but also highway scenes, and countryside roads around Karlsruhe, Germany.

3.3.4 NuScenes

The nuScenes [19] dataset is a large-scale public dataset for autonomous driving. It consists of 1000 publicly available driving scenes, each having a length of 20 seconds, from Boston and Singapore. The LIDAR data was collected using a Velodyne 32 LIDAR rotating at 20 Hz. This is in contrast to the 64-beam Velodyne rotating at 10 Hz used for the KITTI dataset. This difference in sensors leads to a difference in data sparsity that creates a distribution shift between KITTI and nuScenes. Additionally, the nuScenes-lidarseg dataset was published where each point is annotated as one of 32 possible semantic labels.

3.4 Performance Metrics

3.4.1 Scene Flow Metrics

End Point Error (EPE):

describes the mean Euclidean distance between the predicted and ground truth transformed points given by:

$$\mathcal{L} = \frac{1}{N} \sum_{i}^{N} \left\| d_i^* - \hat{d}_i \right\|^2$$

where $\hat{\mathcal{D}} = \left\{ \hat{d}_i \right\}^N$ corresponds to the predicted flow and $\mathcal{D}^* = \{ d_i^* \}^N$ to the ground truth flow.

Accuracies:

are usually measured at two threshold levels. Acc(0.05) is the percentage of scene flow prediction with an EPE smaller than 0.05 meters and Acc(0.1) is the percentage of points

having an EPE smaller than 0.1 meters.

3.4.2 Segmentation Metrics

Panoptic Quality

The panoptic quality (PQ) measures the quality of a predicted panoptic segmentation relative to the ground truth. It involves two steps, namely segment matching and the PQ computation given the matches [3]. The segment matching specifies, that a predicted and ground truth segment can only match if their intersection over union (IoU) is greater than 0.5. The panoptic quality is defined as:

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

where $\frac{1}{TP}\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)$ is simply the average IoU of matched segments, while $\frac{1}{2}|FP| + \frac{1}{2}|FN|$ is added to penalize segments without matches.

Segmentation Quality

Segmentation quality (SQ) is evaluating how closely matched our segments are with their ground truths. When this value comes closer to 1, it means that our TP predicted segments are more closely matched with their ground truths. However, it doesn't take into account any bad predictions.

$$\mathrm{SQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}$$

Recognition Quality

Recognition quality (RQ) is a combination of precision and recall, attempting to identify how effective our model is at getting a prediction right. It is defined as:

$$\mathrm{RQ} = \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

3.4.3 4D Panoptic LiDAR Segmentation Metrics

Often gets measured as LiDAR Segmentation and Tracking Quality (LSTQ) it is defined as:

$$\mathrm{LSTQ} = \sqrt{\underbrace{\frac{1}{C}\sum_{c=1}^{C} \mathrm{IoU}(c)}_{\mathrm{S_{cls}}} \times \underbrace{\frac{1}{T}\sum_{t\in T} \frac{\sum_{s\in S} \mathrm{TPA}(s,t)\mathrm{IoU}(s,t)}{|gt_{id}(t)|}}_{\mathrm{S_{assoc}}}},$$

where S_{cls} and S_{assoc} reflects the segmentation and tracking quality respectively. TPA represents the number of intersections between points that are predicted as i and the ground truth points that have the id of j.

4 Motivation

There are several approaches to the 4D LiDAR panoptic segmentation task. Some of these are based on clustering across consecutive scanscite 4dpls here, associating detected instances in consecutive scans through contrastive learning of their embeddings[20] or aligning and overlapping scans with applying rigid transformation and running inference on the final point cloud directly[21].

To bridge the panoptic LiDAR segmentation task with its 4D version, and inspired by the shortcomings of current approaches, we came up with the idea of treating consecutive scans as a single scene. To this extent, we wanted to employ scene flow alongside rigid transformation to achieve better alignment of consecutive scans and infer consistent instance ids for any consecutive scan pair. Moreover, being able to achieve this would mean that all panoptic LiDAR segmentation networks could be adapted to the 4D panoptic LiDAR segmentation task.

4.1 Dynamic Shifting Network

Dynamic Shifting Network is a panoptic segmentation network that makes its main contribution to instance prediction. After applying a common cylindrical convolution, it branches into semantic and instance prediction components, where it directly infers semantic labels. However, for the instance prediction, the regressed center's output by the instance branch is fed into a multi-layer perceptron. This multi-layer perceptron learns to shift the regressed centers to true instance centers, and each cluster is assumed to be an instance. Finally, the semantic labels of instance points are determined through majority voting.

One of the most important novelties of this work is the adaptation of a panoptic segmentation network to a 4D panoptic segmentation without significantly modifying the existing architecture but data processing. Through rigid transformation, an overlapped scene of two consecutive LiDAR scans is produced. Repeating the previous flow, panoptic segmentation is done on the combined scans and the panoptic labels of the original scans are inferred, having consistent instance ids.

The shortcoming of this approach is rigid transformation assumption failing for dynamic objects, as their motion will not match the motion of the ego-vehicle. Nevertheless, replacing rigid transformation with scene flow estimation can potentially provide better-overlapped scenes and improve this approach.

4.2 Efficient LiDAR Panoptic Segmentation

Efficient LiDAR Panoptic Segmentation improves upon both efficiency and metrics of SemanticKITTI panoptic segmentation task by reducing the LiDAR panoptic segmentation to image panoptic segmentation, with effective methods of front and back conversion between two domains. Segmenting the 2D data obtained through scan unfolding of LiDAR scans, corresponding panoptic segmentation labels of the point cloud are inferred by a KNN algorithm. As a result, the requirement for computation resources is much less due to working on the 2D domain instead of the 3D domain.

Achieving competitive metrics on panoptic segmentation and having a public code base and low resource requirements, was a fitting choice for running inference on our aligned scans.

4.3 PointPWC-Net

PointPWC-Net is a scene flow network for outdoor LiDAR scenes, that can be trained in a self-supervised manner. Leveraging a four-level-based architecture, it predicts scene flow on four different resolutions. Since it upsamples flow from a lower resolution level to a higher resolution level, PointPWC-Net can capture large motion in the scene efficiently. Introducing a self-supervised loss based on Chamfer distance and additional constraints, can be trained with no ground truth.

Given that scene flow, ground truth does not exist for SemanticKITTI, and scene PointPWC-Net was promising for training on SemanticKITTI. The only problem was that its computational requirements exceeded our resources, which required us to try and employ shrunk adaptations of the original PointPWC-Net.

5 Experiments

In the following chapter, experiments that were done will be showcased and discussed. These are methods that were the base of our design decisions, enriched the knowledge space and paved the way to the final method.

When running pretrained scene flow models like Flot [22], it immediately became clear that the size of the point clouds leads to memory issues and also introduces a great hardware dependency. To solve this problem, a voxelization sampling strategy was implemented that can adapt to different classes. This sampling strategy is described in more detail in 6. Furthermore, a switch from the remote computers to a local computer with an RTX2080 Super was performed to gain more VRAM. Thus, in the following, every model and computation was performed on this hardware.

5.1 Scene Flow Experiments

For the scene flow topic, most experiments were based either on the Flot [22], or on the PointPWC [8] network. Therefore, one scenario was taken that includes the dynamic motion of a vehicle (sequence 02, frame 70). The second scenario includes pedestrians, where we wanted to see how detailed the flow would be. Regarding the voxel scale, we applied 1cm for things classes and 1m for stuff classes. Furthermore, the following classes were removed: "unlabeled", "outlier", "road", "parking", "sidewalk", "other-ground", "lane-marking". With this sampling strategy, dynamic objects should be emphasized and the overall representation increased.

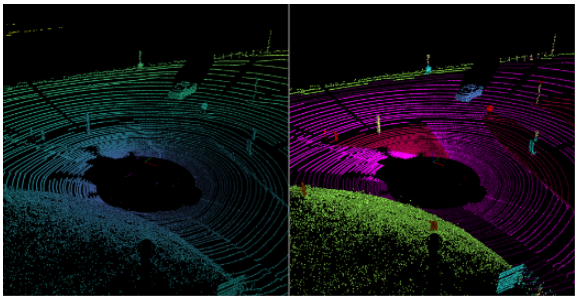

Figure 5.1: Overview of the first scenario. Left image: raw, right image: semantics

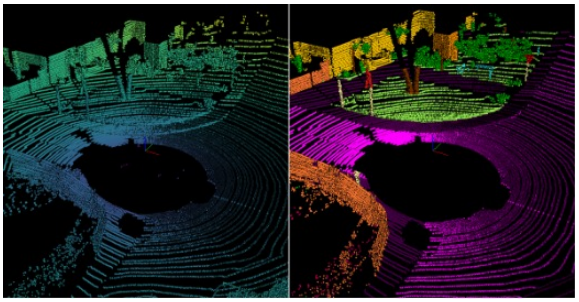

Figure 5.2: Overview of the second scenario. Left image: raw, right image: semantics

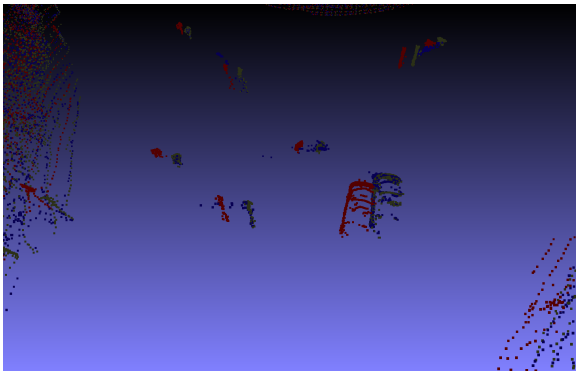

Figure 5.3: Flot on scenario 2; scene 70 and 71.

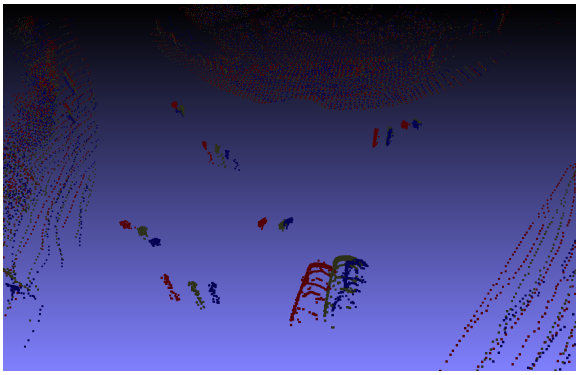

Figure 5.4: PointPWC on scenario 2; scene 70 and71.

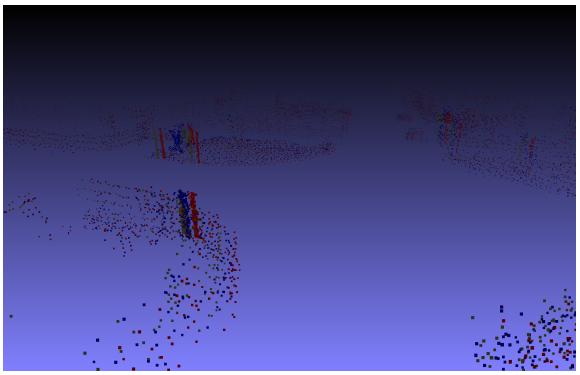

Figure 5.5: Flot on scenario 5; scene 10 and 11.

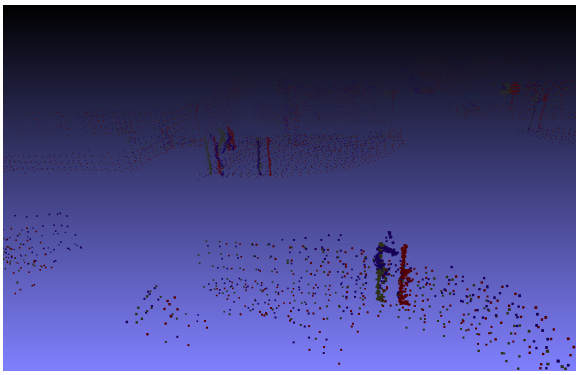

Figure 5.6: PointPWC on scenario 5; scene 10 and 11.

Red: Pointcloud1, Green: Pointcloud2, Blue: Applied flow to Pointcloud1
The images **??-??** showcase how Flot and PointPWC act on the two scenarios. It shows that Flot exceptionally works great for car structures with good precision, but it fails for vertical objects like poles and signs, ending up with cluttered points. This is especially the case if multiple objects are in the same local region. On the other hand, the PointPWC method does not have high precision but is much better at keeping local structures and identifying single objects. This can especially be seen for the pedestrian in **??**. One also has to keep in mind, that Flot only takes the positions as input, while PointPWC Operates mostly on colors and takes positional input mainly for sampling and grouping. For the images above we set all color values to zero. But we also tried with random colors, as well as lidar scan intensities instead of colors which did not give better results. Still, both methods are pretrained on FT3D and Kitti, which do not have a 360-degree field of view. The differences in the dataset are also our assumption on why the methods do not perform as well.

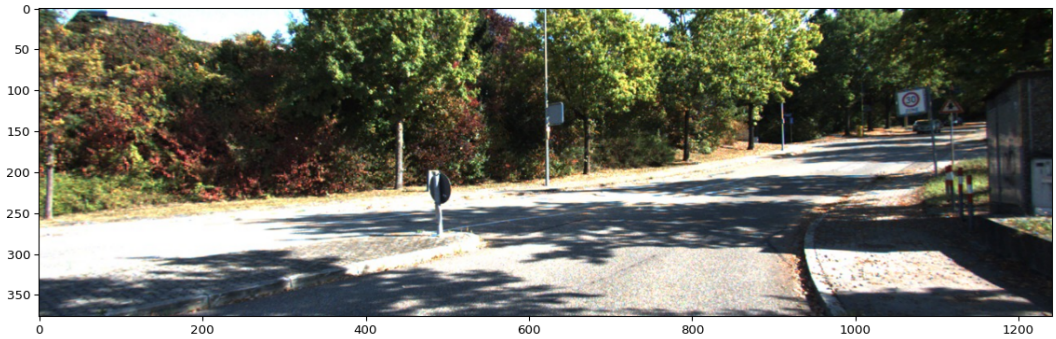

Figure 5.7: Left RGB Camera View

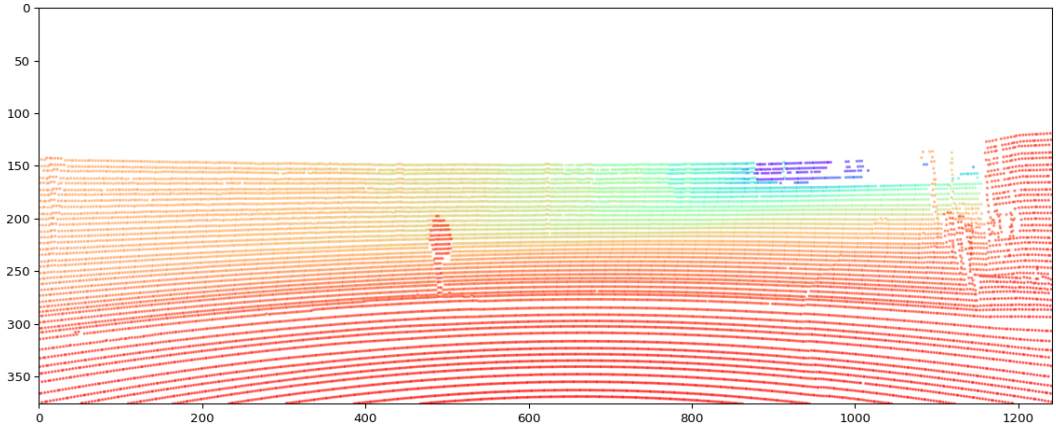

Figure 5.8: Point Cloud Projected to Left RGB Camera

5.1.1 Extracting color values

Given the Velodyne to camera coordinates transformation matrix, and the camera matrix, we process the point cloud for it to be in homogenous coordinates, then apply the transformation. This way, we project the Velodyne scan to the left RGB camera of the ego vehicle. Following the projection, point clouds outside the camera view are filtered out. This way, we end up with the points shown in 5.8 and 5.9. Finally, RGB values from pixels corresponding to the points and their indices in the Velodyne scan are output. We also planned on implementing a ball query to color points outside of the camera view, based on the colors of the points within the ball. Since usually less than a quarter of points are assigned RGB values, this would not allow us to significantly increase the number of colored points. So we did not implement it and after observing the results.

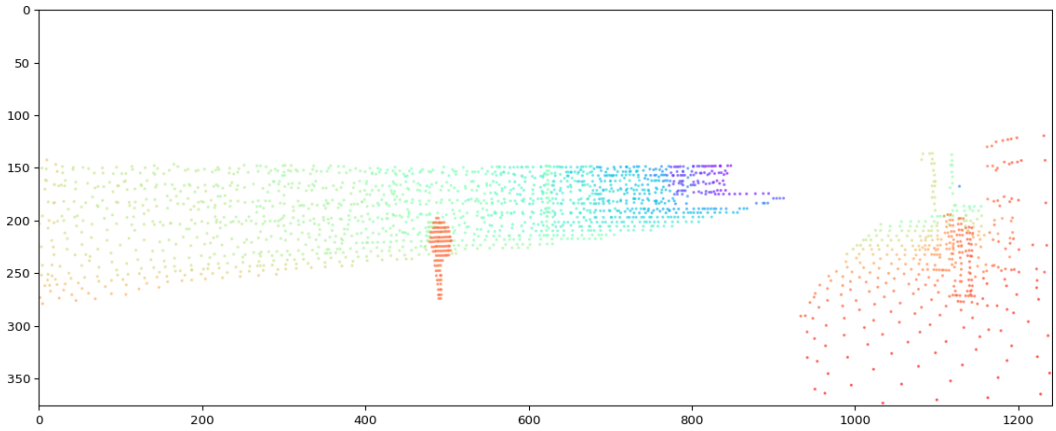

Figure 5.9: Downsampled Voxels Projected to Left RGB Camera

5.2 Losses Experiment

Our inspiration regarding losses came from PointPWC as well as Just Go with the Flow (JGWTF) [7] introducing interesting self-supervised losses for point clouds. PointPWC introduces *Chamfer distance, Smoothness constraint* , and *Laplacian regularization* also known as curvature constraint. JGWTF introduces the *Nearest Neighbor Loss* as well as the *cycle consistency loss*.

Chamfer distance: The goal of Chamfer loss is to estimate scene flow by moving the first point cloud as close as the second one. Let SF_{Θ}^{l} be the scene flow predicated at level l. Let P_w^l be the point cloud warped from the first point cloud P^l according to SF_{Θ}^{l} in level l, Q^l be the second point clout at level l. Let p_w^l and q^l be points in P_w^l and Q^l. The Chamfer loss l_C^l can be written as:

$$P_w^l = P^l + SF_{\Theta}^{l}$$

$$\ell_C^l\left(P_w^l, Q^l\right) = \sum_{p_w^l \in P_w^l} \min_{q^l \in Q^l} \left\| p_w^l - q^l \right\|_2^2 + \sum_{q^l \in Q^l} \min_{p_w^l \in P_w^l} \left\| p_w^l - q^l \right\|_2^2$$

The chamfer loss does not penalize degenerate solutions where all points of the warped point cloud map to the same point in the second point cloud.

Smoothness Constraint: Enforces local spatial smoothness and assumes that the predicted scene flow in local regions should be similar to other flows in the same region.

$$\ell_S^l\left(SF^l\right) = \sum_{p_i^l \in P^l} \frac{1}{\left|N\left(p_i^l\right)\right|} \sum_{p_j^l \in N\left(p_i^l\right)} \left\| SF^l\left(p_j^l\right) - SF^l\left(p_i^l\right) \right\|_2^2$$

Laplacian Regularization: Tries to approximate the local shape characteristics of the surface.

The laplacian coordinate vector is computed as:

$$\delta^l\left(p_i^l\right) = \frac{1}{\left|N\left(p_i^l\right)\right|} \sum_{p_j^l \in N\left(p_i^l\right)} \left(p_j^l - p_i^l\right)$$

Nearest Neighbor (NN) Loss: Can be seen as the normalized first part of the chamfer loss. For each transformed point, the distance to the nearest neighbor of the second point cloud is added.

$$\mathcal{L}_{NN} = \sum_{p_w^l \in P_w^l} \min_{q^l \in Q^l} \left\| p_w^l - q^l \right\|_2^2$$

The loss has problems when point clouds are sufficiently sparse. Then the position of the transformed point cloud may not correspond with any point of the second point cloud. Furthermore, similar to the chamfer loss, also the nearest neighbor loss does not penalize degenerate solutions.

Cycle Consistency Loss: As the constraints losses for the PointPWC network, the cycle loss tries to make up for the issues of the nearest neighbor loss. Therefore, a forward, as well as a backward flow, is applied. If both flows are accurate the new estimated point $\hat{x}_i''$ should be similar to the original point x_i. The error between these points is referred to as the cycle consistency loss given by:

$$\mathcal{L}_{\text{cycle}} = \sum_{i}^{N} \left\| \hat{x}_i'' - x_i \right\|^2$$

We first had a look at the PointPWC losses and wanted to see the impact of the constraint losses. Therefore, we overfitted a scenario and took intermediate inference results along the training. Putting the images together we gained a video, that clearly shows how the points are behaving dependent on each other and the mesh rather acts as a cloth. This can be seen in the video Click Here. Different from when using only the chamfer loss, where the movement seems to be freer, and also able to drift away, which can be seen Click Here.

5.3 Architectures

Hardware limitations were a major obstacle for both training and evaluating models for LiDAR scans. Therefore, we modified PWC-Net to be able to work with less computational resources. Overall, three different versions of PWC-Net were tried. Originally having around 7.8 million parameters, the first version had 3.3 million parameters having the same architecture with less channels. The two other versions had a level removed from the original PWC-net, resolutions were modified and the number of channels was decreased. The only difference between these two versions was the number of channels, as one had 2.3 million parameters and the other, had 2 million parameters. Finally, the original PWC-Net architecture with decreased number of channels seemed appropriate for the task and computational resources at hand.

5.4 Overfitting Experiments

Before starting the training, we tried to overfit the original PointPWC and our implementations to see whether the model capability is sufficient for our task.

5.4.1 Original PWC: Overfit to sequence 00 frames 0-10

As the first step, we wanted to have a comparison between the original architecture and our adapted one with fewer parameters. Thus we trained both on the same dataset, with the same hyperparameters.

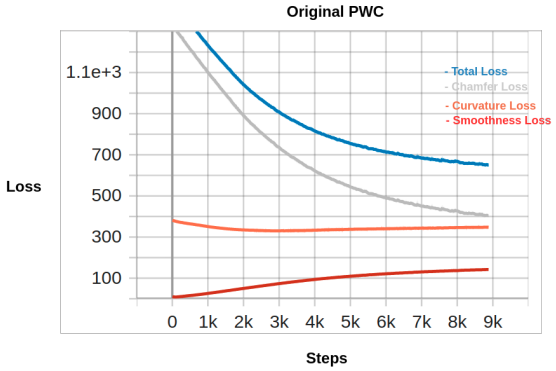

Figure 5.10: Loss functions for the original architecture

We found out, that the loss functions and thus also the training behaves very similarly to our comparison at 5.4.2. Therefore, we concluded that our adaptions to the network seem to work out fine.

5.4.2 Linear Downsampling with no class removed: Overfit to sequence 00 frames 0-10, Loss: All PWCNet losses

With this first check, we wanted to see if we can overfit to a small set of data. Furthermore, we wanted to see the behavior of the loss functions and how it acts on the point clouds.

From the loss functions, one can see that the curvature loss and the smoothing loss work as some kind of constraint in this case. Also, we found that the network can estimate greater changes in the position of the point cloud but mostly struggles with points that classify as buildings, vegetation, or ground. As most of the points belong to one of those classes, they create a lot of noise.

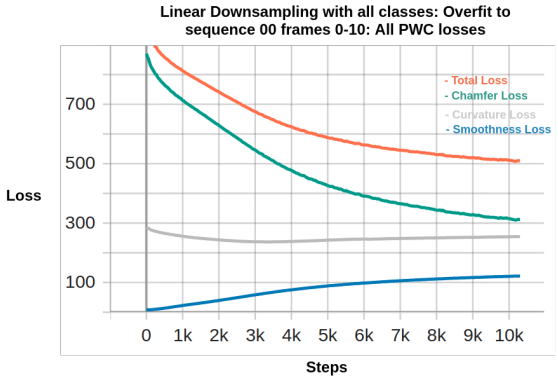

Figure 5.11: Loss functions for the adapted PWCNet architecture

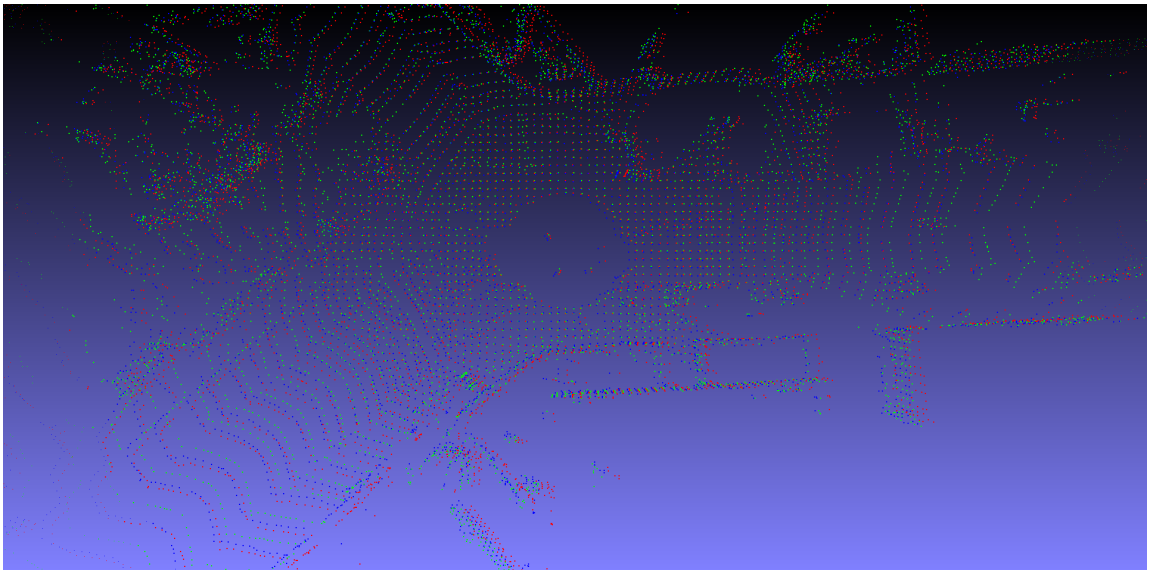

Figure 5.12: Endresult for the adapted pwc architecture with all PointPWC losses

5.4.3 Linear Downsampling with no class removed: Overfit to sequence 00 frames 0-10, Loss: Only Chamfer Loss

To see if we are right with our assumptions of the curvature and smoothing loss, we trained a network only using the chamfer loss.

From the image, we can see, that the network became more flexible. The lines in the lower left that scanned the road are matching better and gained more flexibility to transform to the curve. But also we gained scenarios, in which the points start to clutter and do not represent the initial shape well. This is the case for the wall in the lower mid, as well as for the building in the upper right corner. The Overall loss is a weighted sum of all other losses. So it can be controlled by seeing the weights as a hyperparameter.

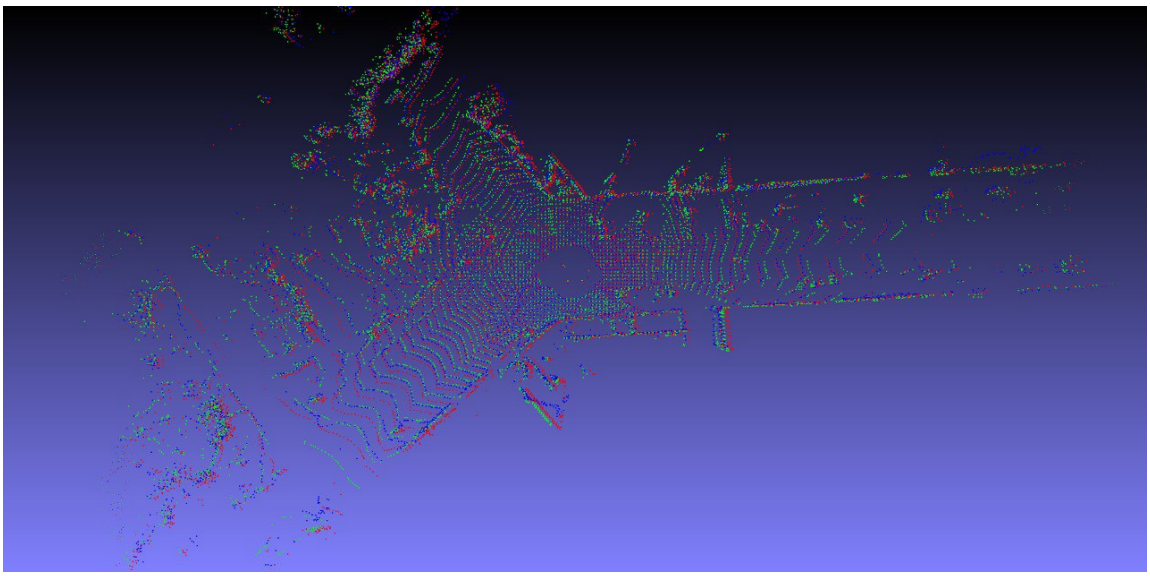

Figure 5.13: Endresult for the adapted PWCNet architecture using only the chamfer loss

5.4.4 Adaptive Downsampling with only dynamic classes: Overfit to sequence 02 frames 65-75, Loss: JGWTF Cycle Loss

We were thinking of using the JGWTF cycle consistency loss as a refinement loss that could be applied in the end, since it directly measures the goodness of the forward, as well as the backward flow.

As it can be seen in **??** we get a very bad result. We reimplemented the function from tensorflow to pytorch and think that a mistake was made in the implementation.

5.4.5 Adaptive Downsampling with only dynamic classes, sequence 02, Loss: All PWC losses

To see how things behave when we remove unwanted classes/noise we set up a model that is only overfitted on dynamic classes. This gives us an additional problem which is size instability. Meaning depending on the scene we have a higher variance of points that we can use. As the PWC network can handle different sizes, but the creation of batches does not (cant stack torches with different sizes). We evaluated two solutions for this problem after first trying to overfit. First, we trained only with one batch and variable sizes. Second, we used zero padding to scale all point clouds to a size of 8192.

Overfit to frames 65-75

Overfitting on different sizes with batch size 1, went great and resulted in almost perfect flow.

Looking especially at the curvature and smoothing loss, we can see that they are more stable

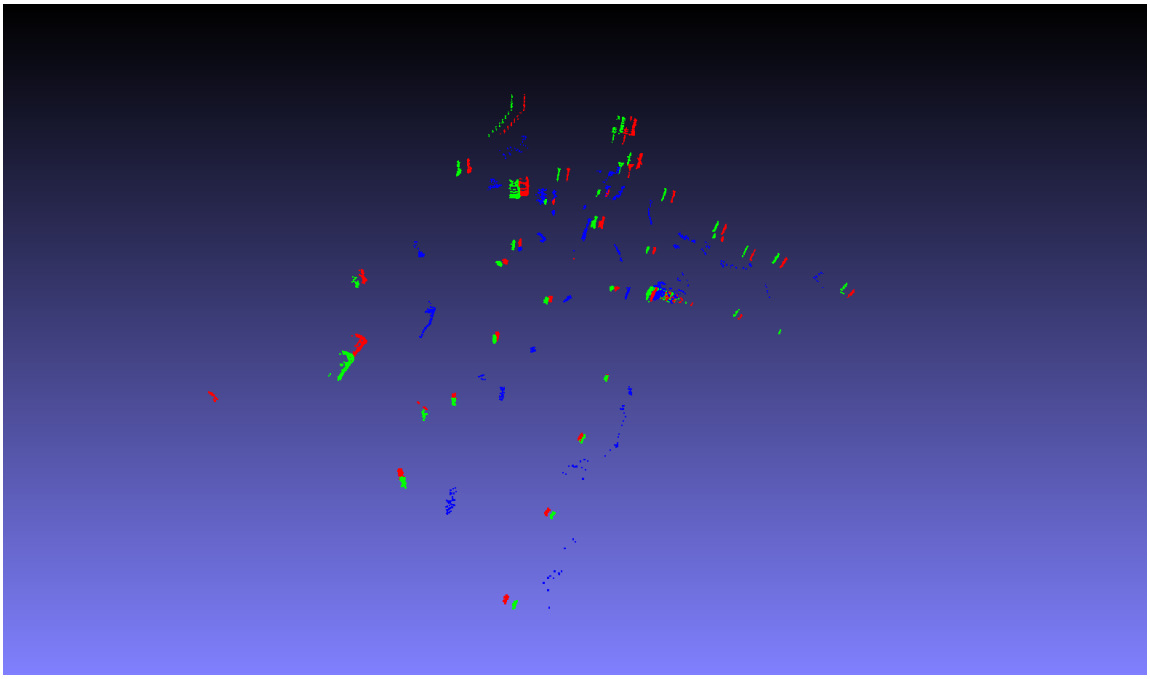

Figure 5.14: Endresult for the adapted PointPWC architecture using only the nearest neighbor loss and the cycle consistency loss

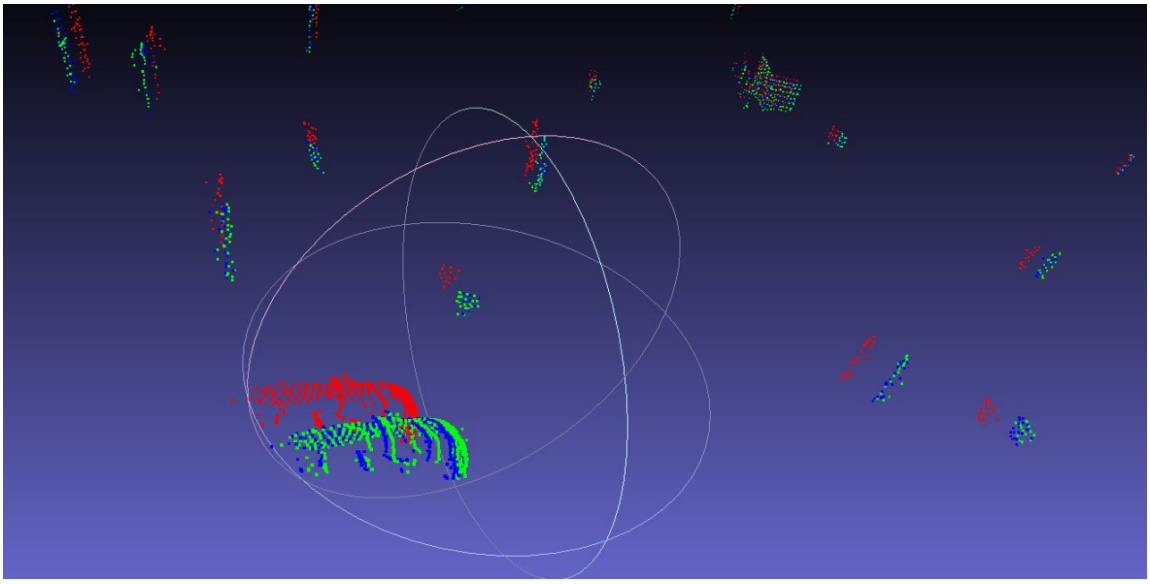

Figure 5.15: Endresult for the adapted PWCNet architecture using only the dynamic classes

and also contribute to a lower overall error. We conclude that non-dynamic classes are responsible for a lot of noise in the curvature and smoothing loss.

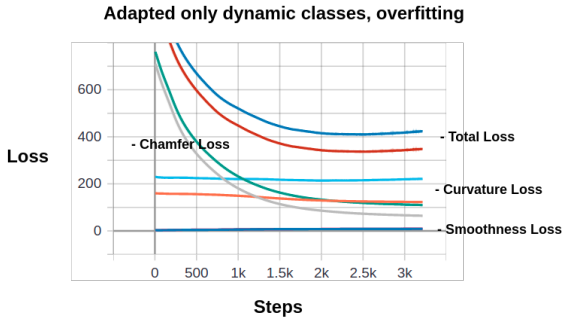

Figure 5.16: Loss functions for overfitting dynamic classes

Whole sequence 2, with no padding

When applying the idea from the overfit to the whole sequence, we had a lot of issues to generalizing with the validation set.

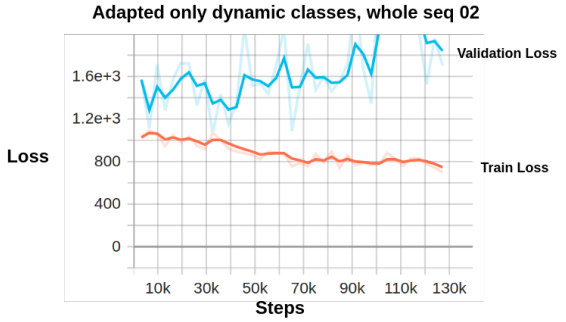

Figure 5.17: Loss functions for training on seq 02 and dynamic classes

Just using one batch is too unstable for the training process of a lot of data.

Whole sequence 2, with padding

By using padding, using the RTX2080 we can scale at least to a batch size of three. This gave some stability in the training process until we also greatly overfitted to this case.

From the knowledge gained from our overfitting experiments, it became more clear, that we should use all PointPWC losses. Also, noise reduction is more important than spatial representation. Furthermore, running a higher batch size while using padding seems to be more stable in the training process while giving similar results.

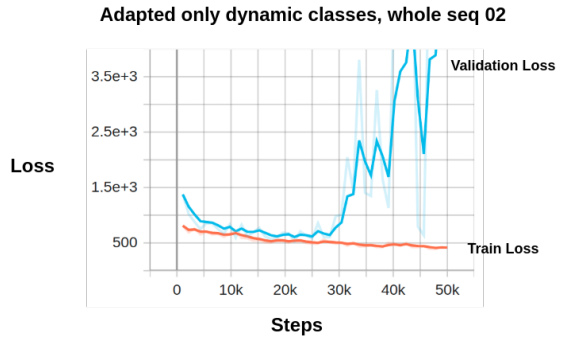

Figure 5.18: Loss functions for training on seq 02 and dynamic classes with using padding

5.5 Pretrained Scene Flow Models

With this work, three pretrained scene flow models are provided, which will be discussed in the following.

SemanticKitti Base Model

The first model is the semanticKitti base model. It was trained on all sequences of the train and validation set of the semanticKiTTI dataset. Namely, those are sequences are [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. The model was trained with a random 70/30 train/val split and a learning rate of 0.000005. For the base model we removed the following classes: ["unlabeled", "road", "parking", "other-ground", "building", "vegetation", "terrain", "other-object"]. As we knew we would have too less data, we included the sidewalk to get a better working space representation. As a sampling strategy, we gave things classes a voxel size of $0.01m$ and stuff classes a voxel size of $0.5m$. If we get too few points (<8192) we use padding.

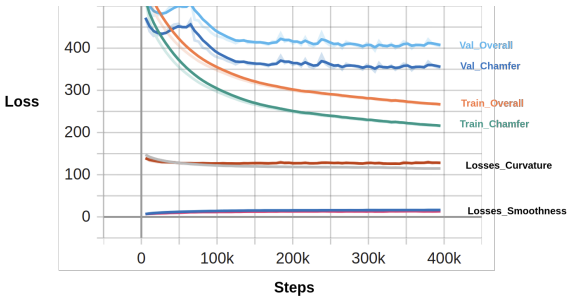

Figure 5.19: Training and validation loss function for the base model

One can easily see, that we are overfitting due to having not enough data. Therefore, we refined the base model with our next model.

Semantic Kitti Forth and Back

We applied the same settings as before just with the difference that we additionally train on also with the backwards flow. Meaning we estimate the flow not only from $t \rightarrow t+1$ but also from $t \rightarrow t-1$. In addition to giving us more data, this also helps to generalize better, and the model to not only learn forward motion.

Semantic Kitti and NuScenes Model

We again applied the same configurations as for the two other models. But we leave out the validation set (sequence 08), to make inference on unseen semantic kitti data. Additionally, we also train on the first 25000 scans of the NuScenes dataset. Different from the models above, we make the assumption of having enough data and also remove the "sidewalk" class from the point cloud, to further reduce noise.

5.6 Panoptic Experiments

Motivated by the DS-Net approach, we tried to apply EfficientLPS inference on an aligned point cloud. This did not work well, as the second pointcloud did not get any labels assigned. We think the reason for this is the backprojection from the image to the point cloud again.

5.7 IoU Matching Experiments

Different from our description in 6.5, where we have no restrictions for the search of the nearest neighbor in the warped point cloud, we initially thought that it would make much more sense to only do the nearest neighbor search for instances. So what we did first is filter for instance labels in the warped point cloud and then apply the matching. What we experienced is, that many of the predictions from the EfficientLPS network do not have exact semantic labels like they are given in the semanticKITTI config file. At this time, we did not know how to make sense of them and had very weird results. It's only now that we know that it takes some remapping of the labels given the learning map before we apply our IoU matching.

6 Method: Flow into 4D Panoptic Segmentation

6.1 Overall Workflow

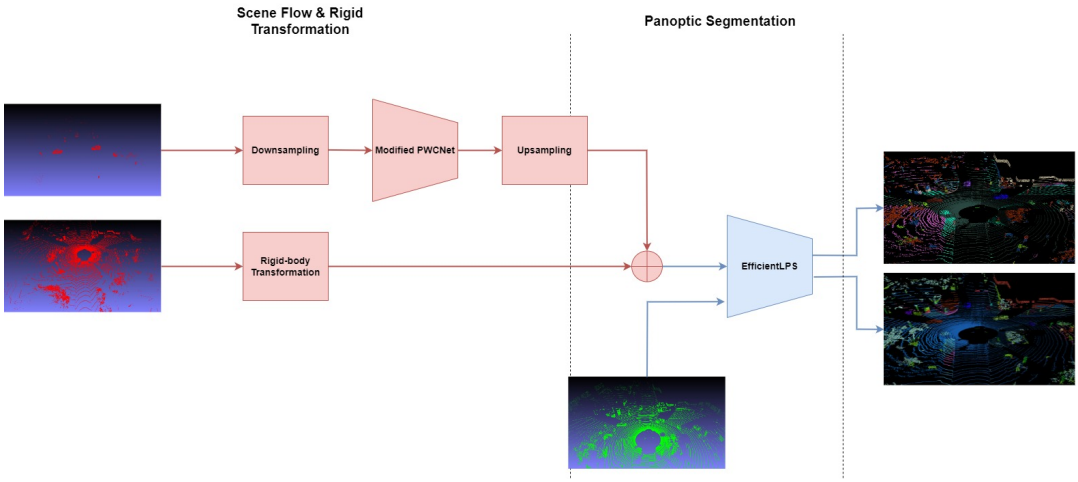

Figure 6.1: Overall Workflow Part 1

The overall workflow includes scene alignment, panoptic segmentation and instance association. In the scene alignment part, dynamic objects are downsampled, fed to the modified PWC-Net and the output is upsampled. For the points that do not belong to dynamic objects, rigid transformation is applied. As a result, any scan is aligned to the next scan. EfficientLPS predicts panoptic segmentation labels for the aligned and the next scans separately. Consequently, two aligned scans have inconsistent label ids. Based on the overlapping of instances from different scans, instance ids are associated and consistency is achieved.

6.2 Sampler

As the raw point cloud data has too much data and results in an exploding memory requirement the number of points needs to be downsampled. This is done via an adaptive voxelization scheme. Therefore, for each semantic class, a specific voxel size can be predefined allowing it to emphasize different labels and shift the representation of different objects by the

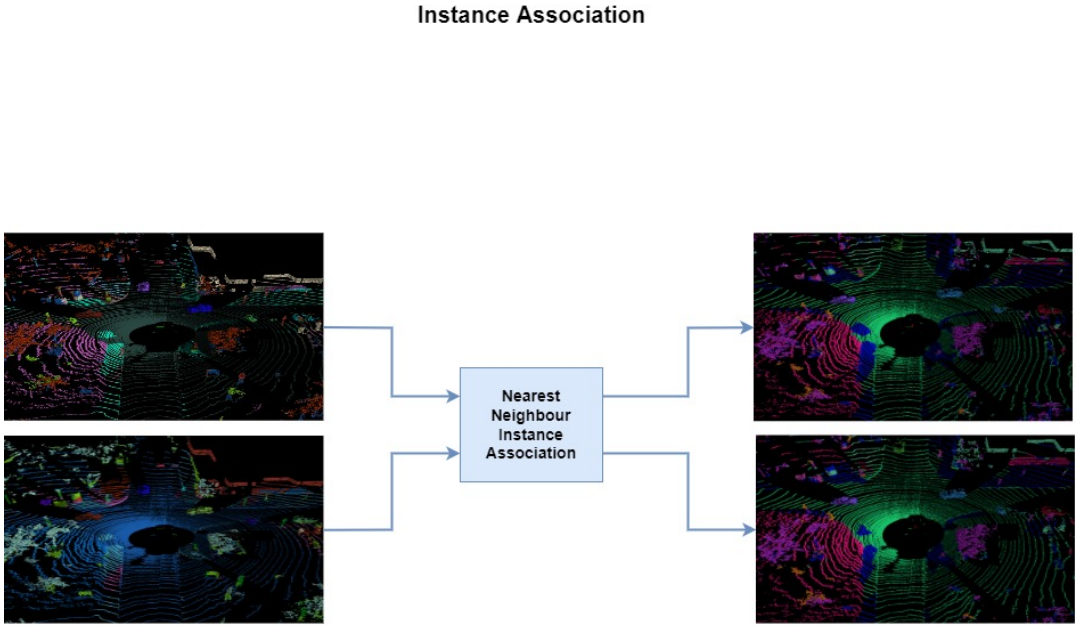

Figure 6.2: Overall Workflow Part 2

points. The sampler gets the raw point cloud and the labels as input. Additionally, specific classes can be defined that should get completely left out. The overall point cloud then gets split into k independent point clouds, where k is the number of unique classes. For each point cloud, a voxelization is performed with the predefined voxel size. Within a voxel only one point is left which is the mean point of all points that fell into the voxel, thus, representing the voxel space. All k split point clouds are then stacked back together leading to our final voxelized point cloud.

After doing inference on this point cloud, a flow value for each point of the raw point cloud is wanted. Therefore, for each point of the original point cloud, the nearest voxel representative of the same class is searched for. Or one could consider that this point is within the voxel. Then the same flow of the representative value is also applied to the raw point. As a voxel matches the L1-norm in a three-dimensional space, the same metric is used to measure the distance to the nearest voxel.

6.3 Scene Flow

Modified PWCNet with less parameters is used for dynamic objects instead of rigid transformation. This is due to the fact that dynamic objects other than the ego vehicle cannot be aligned with rigid transformation. The network is trained on the downsampled SemanticKITTI dataset and outputs the aligned locations of dynamic objects.

6.4 Panoptic Segmentation

Panoptic segmentation is done on each scan separately. Since the two scans are aligned, the assumption is that EfficientLPS will make similar inferences on both scans, that is, the label distributions will be similar except for the instance id inconsistency, which is resolved in the next and final step of the workflow.

6.5 IoU Matching

To solve the time dependency over consecutive scans and get consistent instance labels, an intersection over union matching method was used. Therefore, the assumption is made to see the points of a specific instance as a set representing a certain space. Therefore, the label similarity between the first point cloud $\mathcal{P}_t$ and the transferred point cloud ${\mathcal{P}'}_{t \to t+1}$ as well as the position similarity of ${\mathcal{P}'}_{t \to t+1}$ and the second point cloud $\mathcal{P}_{t+1}$ is exploited. First, the points of $\mathcal{P}_{t+1}$ are clustered into k subsets. Then for each point of a set, the nearest neighbor in ${\mathcal{P}'}_{t \to t+1}$ is searched for. Thus, every point of a set votes for an instance of the previous scan. This gives a distribution of matches, from which we choose the highest match if its distribution is over 50 percent.

7 Evaluation

7.1 Scene Flow

For the first two models we did the mistake and trained on all available data, which does not give us a test set for a meaningful evaluation. Therefore, we will evaluate only the Semantic Kitti and NuScenes model. As described earlier the general metric to measure the goodness of the scene flow is the end point error. As we don't have any ground truth metrics we cannot evaluate our results in this way. Instead we introduce our own metric which we think comes closest to the end point error. The metric with which we measure our results is the mean scaled chamfer loss (MSCL). It is defined as the chamfer loss divided by two times the number of points for which a flow value exists (e.g. is nonzero). Similar to the accuracy metrics, we also introduce the Acc(0.1) and the Acc(0.05) metric, which is based on the mean scaled chamfer loss. We do know that we cannot make a one-to-one comparison this way with other methods. But we want to create a precise measurment that is easy to understand. Other self-supervised methods do their metrics on scene flow datasets as FT3D. But as we are only interested in building a model for the semanticKitti dataset, this is not an alternative. Thus we evaluate our scene flow on a selection of frames of the unseen sequence 08 of the semanticKitti dataset.

Scene Flow Evaluation			
Data	MSCL	Acc(0.05)	Acc(0.1)
sequence 08: 0-50	0.0319	0.86	0.96
sequence 08: 200-225	0.0410	0.6	0.96
sequence 08: 700-725	0.1472	0.0	0.08
sequence 08: 1700-1725	0.0985	0.64	0.76

Table 7.1: Evaluation of scene flow based on different snippets of the semanticKitti dataset.

With the data shown in the table above, we can show that despite a few scenarios, the scene flow shows good results with a nearest neighboring distance of usually less than 10cm.
The images show how the flow acts on unseen data of the semanticKitti dataset. One can see, that in general it does work well for cars, but is also is depending on the point structure. With more data and data augmentation. The network surely could perfome even better.

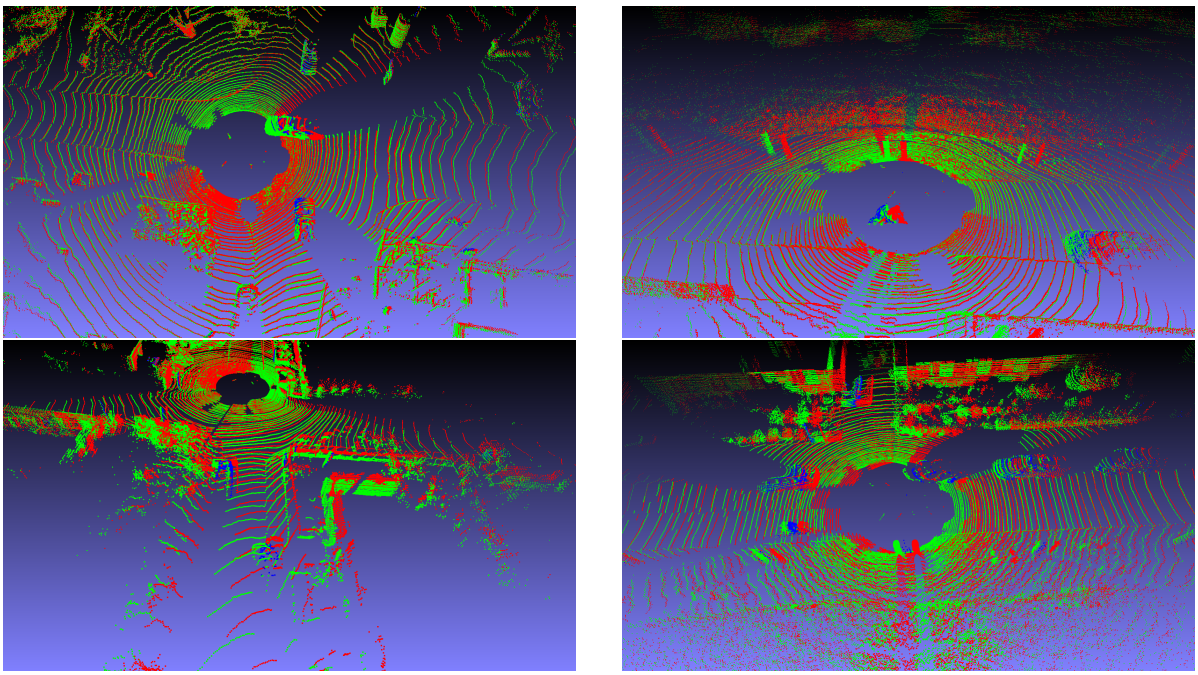

Figure 7.1: Example images taken from the test set to showcase the results of the scene flow.

7.2 4D Panoptic Segmentation

The results given in the videos that can be seen HERE and HERE, show that the time dependency issue is solvable with our concept. Things classes like cars get tracked for a certain amount of time. As we currently are also doing nearest neighbor for stuff classes, the labels change as soon as the instance has more matches to the semantic class, which is due to that cars far away get smaller and thus the relation of points for things to stuff classes changes. One would need to only compare to things classes, after applying the remapping.

8 Discussion

The main contribution of this work was to build a common approach for adapting panoptic segmentation networks to 4D panoptic segmentation task. Through scene flow and rigid transformation, we wanted to align consecutive scans and infer a similar result for the aligned scans. On this purpose, we have encountered many bottlenecks and came up with our solutions to achieve the final results. As of now, despite producing temporally consistent instance ids for a subset of the dataset, we have not yet finalized the testing and competition submission parts, as a few more steps are needed. Also, a list of ideas still remains that are not tried out yet due to time and hardware constraints.

8.1 Further work

8.1.1 Retraining FLOT on SemanticKITTI

Many scene flow models are not optimized for semanticKITTI's 360-degree scans. Even if they are trained on semanticKITTI, the datasets are preprocessed to be similar to the benchmark dataset, such as the kittiFlow dataset. FLOT follows this case, and could not perform well for semanticKITTI's 360-degree scans. Therefore, training FLOT in a self-supervised setting and seeing if it is better than PointPWC-Net is a next step for us.

8.1.2 Retraining and Debugging EfficientLPS on Aligned Scans

We initially wanted to run EfficientLPS on the combined scenes, which are usually twice as denser as the single scans. Unfortunately, EfficientLPS had a bug regarding the second point cloud in the combined scene. Also, making the input scans twice as denser may require further adjustment of model parameters and retraining of the model. Consequently, training and debugging EfficientLPS is a potential next step for our work.

8.1.3 Use more data and data augmentation

As we have shown, the scene flow model tends to overfit the data. The first reason for this is simply too little data. Another reason for this is, that the data is repeating a lot. The car mostly drives on a road, thus the network sees the same things multiple times from only a small deviation than before. Therefore, one could make use of data augmentation like random rotation and translation of point clouds, to generate more unique point clouds.

8.1.4 Exploring Other Panoptic Segmentation Models

As our approach is supposed to be model-agnostic, replacing EfficientLPS with other panoptic segmentation models and experimenting is always a task. A further advantage of this step would be, however, the current existence of more performant models that have easy to approach code bases. Incorporating such models could help us get better metrics.

8.1.5 Completion of Test Pipeline

Currently, we need another step for generating the labels for the test sequences. An immediate step would be to implement a two iteration process. First, EfficientLPS infers semantic labels of the test sequences, and then, adaptive downsampling is applied based on the EfficientLPS output, which prepares the test sequences for scene flow inference and the rest of our workflow. In consequence, submission-ready labeled scans can be generated.

8.2 Encountered Problems

8.2.1 Hardware Limitations

Unfortunately, depending on the method, 3D datasets usually demand more computation resources. This limited us from retraining existing methods given time limitations, or completely experimenting with the given method at all. We overcame this problem by mostly experimenting with efficiency-focused methods. When it was the best option, we modified the model to work with less parameters as in the case of PointPWC-Net. For EfficientLPS, we tried out different resolutions for 2D data, but the given GPUs were not able to work with a resolution that had reasonable inferences. Therefore, we mitigated this problem by moving on to a personal GPU that was sufficient for running EfficientLPS with a batch size 1, with the original resolution.

8.2.2 Dependency Issues

A lot of methods that have published code bases use legacy library versions or have conflicting requirements. Therefore, trying to reproduce others' work was time consuming in the beginning, and we were not able to work with some of the existing methods. We simply ran through several methods, and stick to the ones we were able to get working with. We have not yet tried out all the promising methods. However, we have acquired experience in assessing which methods would be easier to work with, based on their requirements, which would accelerate this process.

8.2.3 EfficientLPS on Combined Scenes

EfficientLPS had problems inferring on combined scenes, where the labeling of the points belonging to the second point cloud was bugged. We investigated the model thoroughly, however, have not found the reason yet. Since the model works on 2D domain and only

projects and back-projects from and to 3D domain, change in point cloud density should not cause such a bug. Since we were limited by time to retrain EfficientLPS or replace it with another method, we designed a workaround where EfficientLPS infers on the aligned scans separately, and we ensure label consistency with a nearest neighbour method.

8.2.4 Scene Flow Issues

Experimentations with scene flow networks were done in different settings. Since FLOT and PointPWC-Net work on a fixed number of points, we first downsampled the 360-degree scans naively and run inference. Unsurprisingly, this yielded poor results. Following this observation, we decided on applying rigid transformation to non-dynamic classes and focus scene flow on dynamic objects. Accordingly, we implemented adaptive downsampling that samples from dynamic objects. We obtained promising yet not sufficient results, so we continued with retraining PointPWC-Net. The sampling strategy was too important, as downsampling too much required us to pad the scans with less than 8192 points and the generalization was abysmal, so we adapted our sampling strategy and managed to obtain decent scene flow results.

9 Conclusion

As a result of this project, we implemented a 4D panoptic segmentation workflow with replaceable scene flow and panoptic segmentation components. For this purpose, we retrained a scene flow model, differing from their usual setting, on 360-degree point clouds and implemented necessary pre-processing steps to adapt it for our work flow. Combining scene flow on dynamic objects and rigid transformation on the rest, we showed that it is possible to align consecutive scans and improve upon the alignment of dynamic objects by using scene flow instead of rigid transformation. Finally, we showed that, to some degree, it is possible to obtain similar predictions for consecutive scans through the previously explained alignment process, and the instance consistency can be provided with a nearest neighbour method. We further defined the process to extend the current workflow to test sequences.

List of Tables

Bibliography

[1] V. Zuanazzi, J. van Vugt, O. Booij, and P. Mettes. "Adversarial self-supervised scene flow estimation". In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 1049–1058.

[2] I. Tishchenko, S. Lombardi, M. R. Oswald, and M. Pollefeys. "Self-supervised learning of non-rigid residual flow and ego-motion". In: *2020 international conference on 3D vision (3DV)*. IEEE. 2020, pp. 150–159.

[3] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. "Panoptic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9404–9413.

[4] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixé. "4d panoptic lidar segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5527–5537.

[5] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. "Three-dimensional scene flow". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. IEEE. 1999, pp. 722–729.

[6] X. Liu, C. R. Qi, and L. J. Guibas. "Flownet3d: Learning scene flow in 3d point clouds". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 529–537.

[7] H. Mittal, B. Okorn, and D. Held. "Just go with the flow: Self-supervised scene flow estimation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11177–11185.

[8] W. Wu, Z. Wang, Z. Li, W. Liu, and L. Fuxin. "Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds". In: *arXiv preprint arXiv:1911.12408* (2019).

[9] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. "Multi-view convolutional neural networks for 3d shape recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 945–953.

[10] A. Boulch, J. Guerry, B. Le Saux, and N. Audebert. "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks". In: *Computers & Graphics* 71 (2018), pp. 189–198.

[11] G. Riegler, A. Osman Ulusoy, and A. Geiger. "Octnet: Learning deep 3d representations at high resolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3577–3586.

[12] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. "Segcloud: Semantic segmentation of 3d point clouds". In: *2017 international conference on 3D vision (3DV)*. IEEE. 2017, pp. 537–547.

[13] D. Maturana and S. Scherer. "Voxnet: A 3d convolutional neural network for real-time object recognition". In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 922–928.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems* 30 (2017).

[16] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[17] N. Mayer, E. Ilg, P. Hausser, 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: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048.

[18] 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: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9297–9307.

[19] 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: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[20] R. Marcuzzi, L. Nunes, L. Wiesmann, I. Vizzo, J. Behley, and C. Stachniss. "Contrastive Instance Association for 4D Panoptic Segmentation Using Sequences of 3D LiDAR Scans". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1550–1557. DOI: 10.1109/LRA.2022.3140439.

[21] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu. *LiDAR-based Panoptic Segmentation via Dynamic Shifting Network*. 2020. DOI: 10.48550/ARXIV.2011.11964. URL: https://arxiv.org/abs/2011.11964.

[22] G. Puy, A. Boulch, and R. Marlet. "Flot: Scene flow on point clouds guided by optimal transport". In: *European conference on computer vision*. Springer. 2020, pp. 527–544.