OGC: Unsupervised 3D Object Segmentation from Rigid Dynamics of Point Clouds

Ziyang Song **Bo Yang**

vLAR Group, The Hong Kong Polytechnic University

ziyang.song@connect.polyu.hk bo.yang@polyu.edu.hk

Abstract

In this paper, we study the problem of 3D object segmentation from raw point clouds. Unlike all existing methods which usually require a large amount of human annotations for full supervision, we propose the first unsupervised method, called OGC, to simultaneously identify multiple 3D objects in a single forward pass, without needing any type of human annotations. The key to our approach is to fully leverage the dynamic motion patterns over sequential point clouds as supervision signals to automatically discover rigid objects. Our method consists of three major components, 1) the object segmentation network to directly estimate multi-object masks from a single point cloud frame, 2) the auxiliary self-supervised scene flow estimator, and 3) our core object geometry consistency component. By carefully designing a series of loss functions, we effectively take into account the multi-object rigid consistency and the object shape invariance in both temporal and spatial scales. This allows our method to truly discover the object geometry even in the absence of annotations. We extensively evaluate our method on five datasets, demonstrating the superior performance for object part instance segmentation and general object segmentation in both indoor and the challenging outdoor scenarios. Our code and data are available at https://github.com/vLAR-group/OGC

1 Introduction

Identifying 3D objects from point clouds is vital for machines to tackle high-level tasks such as autonomous planning and manipulation in real-world scenarios. Inspired by the seminal work PointNet [56], a plethora of sophisticated models [70; 76; 37] have been developed to accurately detect and segment individual objects from the sparse and irregular point clouds. Although these methods have achieved excellent performance on a wide range of public datasets, they primarily rely on a huge amount of human annotations for full supervision. However, it is extremely costly to fully annotate every objects in point clouds due to the irregularity of data format.

Very recently, a few works start to address 3D object segmentation in the absence of human annotations. By analysing 3D scene flows from sequential point clouds, Jiang *et al*. [28] apply the conventional subspace clustering optimization technique to identify moving objects from raw point cloud sequences. With the self-supervised learning of 3D scene flow, SLIM [3] is the first learning-based work to showcase that the set of moving points can be effectively learned as an object against the stationary background. Fundamentally, their design principle shares the key spirit of Gestalt theory [73; 69] developed exactly 100 years ago: the raw sensory data with similar motion are likely to be organized into a single object. This is indeed true in the real world where solid objects usually have strong correlation in rigid motions. However, these methods cannot learn to simultaneously segment multiple interested 3D objects from a single point cloud in one go.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

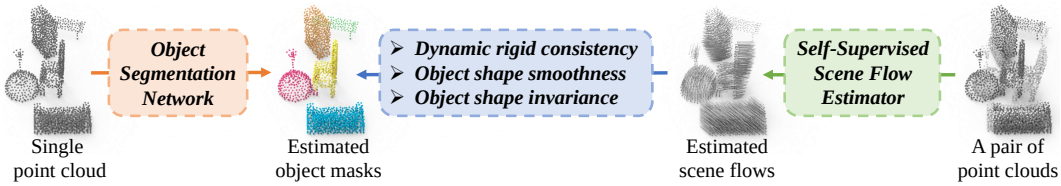

Figure 1: The general workflow and components of our framework.

Motivated by the potential of motion dynamics, this paper aims to design a general neural framework to simultaneously segment multiple 3D objects, without requiring any human annotations but the inherent object dynamics in training. To achieve this, a naïve approach is to train a neural network to directly cluster motion vectors into groups from sequential point clouds, which is widely known as motion segmentation [77; 78]. However, such design requires that the input data points are sequential in both training and testing phases, and the trained model cannot infer objects from a single point cloud. Fundamentally, this is because the learned motion segmentation strategies simply cluster similar motion vectors instead of discriminating object geometries, and therefore such design is not general enough for real-world applications.

In this regard, we design a new pipeline which takes a single point cloud as input and directly estimates multiple object masks in a single forward pass. Without needing any human annotations, our pipeline instead leverages the underlying dynamics of sequential point clouds as supervision signals. In particular, as shown in Figure 1, our architecture consists of three major components: 1) an object segmentation network to extract per-point features and estimate all object masks from *a single point cloud*, as indicated by the orange block; 2) an auxiliary self-supervised network to estimate per-point motion vectors from a pair of point clouds, as indicated by the green block; 3) a series of loss functions to fully utilize the motion dynamics to supervise the object segmentation backbone, as indicated by the blue block. For the first two components, it is actually flexible to adopt any of existing neural feature extractors [57] and self-supervised motion estimators [34]. Nevertheless, the third component is particularly challenging to design, primarily because we need to take into account not only the consistency of diverse dynamics of multiple objects in a sequence, but also the invariance of object geometry irregardless of different moving patterns.

To tackle this challenge, we introduce three key losses to end-to-end train our object segmentation network from scratch: 1) a multi-object dynamic rigid consistency loss, which aims to evaluate how coherently all estimated object masks (shapes) can fit the motion via rigid transformations; 2) an object shape smoothness prior, which regularizes all points of each estimated object to be spatially continual instead of fragmented; 3) an object shape invariance loss, which drives multiple estimated masks of a particular object to be invariant given different (augmented) rigid transformations. These losses together force all estimated **o**bjects' **g**eometry to be **c**onsistent and represented by high-quality masks, purely from raw 3D point clouds without any human annotations. Our method is called **OGC** and our contributions are:

- We introduce the first unsupervised multi-object segmentation pipeline on single point cloud frames, without needing any human annotations in training or multiple frames as input.
- We design a set of geometry consistency based losses to fully leverage the object rigid dynamics and shape invariance as effective supervision signals.
- We demonstrate promising object segmentation performance on five datasets, showing significantly better results than classical clustering and optimization baselines.

Difference from Scene Flow Estimation: We do not aim to design a new scene flow estimation method such as [74; 34; 20; 71]. Instead, we use unsupervised learning based per-point scene flow as supervision signals for single-frame multi-object segmentation.

Difference from Motion Segmentation: We neither aim to segment motion vectors such as [71; 63] which require multiple successive frames as input in both training and testing. Instead, our network directly estimates object masks from single frames, and therefore is more flexible and general.

Scope: This paper does not intend to replace fully-supervised approaches because the never-moving objects are unlikely to be discovered due to the lack of supervision signals. In addition, estimating object categories or non-rigid objects such as articulated buses and semi-truck with trailers is also out of the scope of this paper.

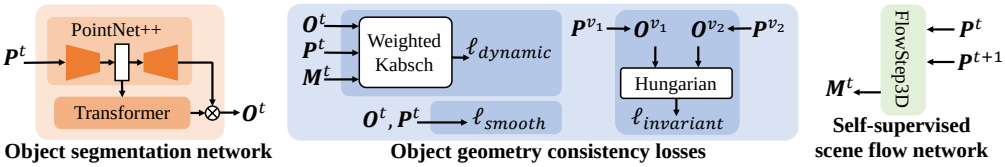

Figure 2: Components of our pipeline. The object segmentation network consists of PointNet++ and Transformer decoders. FlowStep3D is adopted as the self-supervised scene flow network.

2 Related Works

Fully-supervised 3D Object Segmentation: To identify 3D objects from point clouds, existing fully-supervised solutions can be divided as 1) bounding box based object detection methods [84; 37; 60] or 2) mask based instance segmentation pipelines [70; 76; 68]. Thanks to the dense human annotations and the well developed backbones including projection-based [38; 8; 37], point-based [57; 65; 26] and voxel-based [22; 12] feature extractors, these methods achieve impressive performance on both indoor and outdoor datasets. However, manually labelling every object in large-scale point clouds is costly. To alleviate this burden, we aim to pioneer 3D object segmentation without human labels.

3D Scene Flow Estimation and Motion Segmentation: Given sequential point clouds, per-point 3D motion vectors, also known as scene flow, can be accurately estimated. Early works focus on fully-supervised scene flow estimation [42; 4; 24; 43; 55; 53; 72], whereas recent methods start to explore self-supervised motion estimation [50; 74; 34; 46; 82; 39]. Taking the scene flow as input, a number of works [80; 27; 64; 3] aim to group similar motion vectors, and then obtain bounding boxes or masks only for dynamic objects. Although achieving encouraging results, they either rely on ground truth segmentation for supervision or can only segment simple foreground and background objects, without being able to simultaneously segment multiple objects. In this paper, we leverage the successful self-supervised scene flow estimator as our auxiliary neural network to provide valuable supervision signals, so that multiple objects can be identified in a single forward pass.

Unsupervised 2D Object Segmentation: Inspired by the early work AIR [16], a large number of generative models have been proposed to discover objects from single images without needing human annotations, including MONet [6], IODINE [23], Slot-Att [44], *etc.*. These methods are further extended to segment objects from video frames [35; 25; 49; 29; 85; 15]. However, as investigated by the recent work [79], all these approaches can only process simple synthetic datasets, and cannot discover objects from complex real-world images yet. It is still elusive to apply these ideas on 3D point clouds where 3D objects are far more complicated and diverse in terms of geometry.

2D Scene Flow Estimation and Motion Segmentation: Given image sequences, pixel-level 2D scene flow, also known as optical flow, have been extensively studied in literature [18; 83]. The estimated flow field can be further grouped as objects [61; 62; 10; 45; 77; 41; 32]. Drawing insights from these works, this paper aims to segment multiple diverse objects in the complex 3D space.

3 OGC

3.1 Overview

As shown in Figure 2, given a single point cloud $\boldsymbol{P}^t$ with N points as input, *i.e.*, $\boldsymbol{P}^t \in \mathbb{R}^{N\times 3}$, where each point only has a location $\{x, y, z\}$ without color for simplicity, the **object segmentation network** extracts per-point features and directly reasons a set of object masks, denoted as $\boldsymbol{O}^t \in (0,1)^{N\times K}$, where K is a predefined number of objects that is large enough for a specific dataset. In particular, we firstly adopt PointNet++ [57] to extract the per-point local features. Then we employ Transformer decoders [67] to attend to the point features and yield all object masks in parallel. The whole architecture can be regarded as a 3D extension of the recent MaskFormer [9] which shows excellent performance in object segmentation in 2D images. Thanks to the powerful Transformer module, each inferred object mask is effectively modeled over the entire point cloud. Implementation details are in Appendix A.1

In the meantime, we have the corresponding sequence of point clouds for supervision, denoted as $\{\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \cdots\}$. For simplicity, we only use the first two frames $\{\boldsymbol{P}^t, \boldsymbol{P}^{t+1}\}$ and feed them into

the **auxiliary self-supervised scene flow network**, obtaining satisfactory motion vectors for every point in the first point cloud frame, denoted as $\boldsymbol{M}^t \in \mathbb{R}^{N\times 3}$, where each motion vector represents point displacement $\{\Delta x, \Delta y, \Delta z\}$. Among the existing self-supervised scene flow methods, we choose the recent FlowStep3D [34] which shows excellent scene flow estimation in multiple datasets. Implementation details are in Appendix A.1. To train the object segmentation network from scratch, the key component is the supervision mechanism as discussed below.

3.2 Object Geometry Consistency Losses

Given the input point cloud $\boldsymbol{P}^t$ and its output object masks $\boldsymbol{O}^t$ and motion $\boldsymbol{M}^t$, we introduce the following objectives to satisfy the geometry consistency on both frames $\boldsymbol{P}^t$ and $(\boldsymbol{P}^t + \boldsymbol{M}^t)$. Note that, the masks $\boldsymbol{O}^t$ are meaningless at the very beginning and need to be optimized appropriately.

(1) Geometry Consistency over Dynamic Object Transformations

From time t to $t+1$, the rigid objects in point cloud frame $\boldsymbol{P}^t$ usually exhibit different dynamic transformations which can be described by matrices belonging to $SE(3)$ group. For the kth object, we firstly retrieve its (soft) binary mask $\boldsymbol{O}_k^t$, and then feed the tuple $\{\boldsymbol{P}^t, \boldsymbol{P}^t + \boldsymbol{M}^t, \boldsymbol{O}_k^t\}$ into the differentiable weighted-Kabsch algorithm [31; 21], estimating its transformation matrix $\boldsymbol{T}_k \in \mathbb{R}^{4\times 4}$.

In order to drive all raw object masks to be more and more accurate, so as to fully explain the corresponding motion patterns within all masks, the following dynamic rigid loss is designed to minimize the discrepancy of per-point scene flow between time t and $t+1$ for each point in $\boldsymbol{P}^t$:

$$\ell_{dynamic} = \frac{1}{N}\sum_{n=1}^{N}\Big\|\Big(\sum_{k=1}^{K} o_{nk}\cdot(\boldsymbol{T}_k \circ \boldsymbol{p}_n)\Big) - (\boldsymbol{p}_n + \boldsymbol{m}_n)\Big\|_2 \tag{1}$$

where $o_{nk} \in (0,1)$ represents the probability of being assigned to the kth object for a specific n^{th} point $\boldsymbol{p}_n$, and $\boldsymbol{m}_n \in \mathbb{R}^3$ represents the motion of $\boldsymbol{p}_n$. The operation $\circ$ applys the rigid transformation to the point. Intuitively, if one inferred object mask includes two sets of points with two different moving directions, the transformed point cloud can only favor one moving direction, thereby resulting in higher errors. Therefore, $\ell_{dynamic}$ can push all object masks to fit the dynamic and diverse motion patterns. However, here arises a critical issue: a single rigid object may be assigned to multiple masks, *i.e.* oversegmentation. We alleviate this issue by a simple smoothness regularizer discussed below.

We observe that such rigid constraint concept is also applied in recent scene flow estimation method [20]. However, their objective is to push the scene flow to be consistent given object masks (estimated by DBSCAN clustering), while our objective is to learn high-quality masks from given flows.

(2) Geometry Smoothness Regularization

The primary reason why a single object may be oversegmented is the lack of spatial connectivity between individual points. However, our common observation is that physically neighbouring points usually belong to a single object. In this regard, we simply introduce a geometry smoothness regularizer. Particularly, for a specific n^{th} 3D point $\boldsymbol{p}_n$ in the point cloud $\boldsymbol{P}^t$, we firstly search H points from its neighbourhood using either KNN or spherical querying methods, and then force their mask assignments to be consistent with the center point $\boldsymbol{p}_n$. Mathmatically, it is defined as:

$$\ell_{smooth} = \frac{1}{N}\sum_{n=1}^{N}\Big(\frac{1}{H}\sum_{h=1}^{H} d(\boldsymbol{o}_n, \boldsymbol{o}_{n_h})\Big) \tag{2}$$

where $\boldsymbol{o}_n \in (0,1)^K$ represents the object assignment of center point $\boldsymbol{p}_n$, and $\boldsymbol{o}_{n_h} \in (0,1)^K$ represents the object assignment of its hth neighbouring point. The distance function $d()$ is flexible to choose $L1$ / $L2$ or a more aggressive cross-entropy function.

Note that, such local smoothness prior is successfully used for scene flow estimation [42; 34]. Here, we instead demonstrate its effectiveness for object segmentation.

(3) Geometry Invariance over Scene Transformations

With the above geometry constraints designed in (1)(2), the shapes of dynamic objects can be reasonably segmented. However, the learned object geometry may not be general enough. For example, a moving car can be well segmented, yet another similar parked car may not be discovered. To this end, we introduce an object geometry invariance constraint as follows:

- Firstly, given $\boldsymbol{P}^t$, we apply two transformations to get augmented point clouds $\boldsymbol{P}^{v_1}$ and $\boldsymbol{P}^{v_2}$.
- Secondly, we feed $\boldsymbol{P}^{v_1}$ and $\boldsymbol{P}^{v_2}$ into our object segmentation network, obtaining two sets of object masks $\boldsymbol{O}^{v_1}$ and $\boldsymbol{O}^{v_2}$. Because the per-point locations in two point clouds are transformed differently, the position sensitive PointNet++ [57] features generate two different sets of masks.
- Thirdly, we leverage the Hungarian algorithm [36] to one-one match the individual masks in $\boldsymbol{O}^{v_1}$ and $\boldsymbol{O}^{v_2}$ according to the object pair-wise IoU scores. Basically, this is to address the issue that there is no fixed order for predicted object masks from the two augmented point clouds.
- At last, we reorder the masks in $\boldsymbol{O}^{v_2}$ to align with $\boldsymbol{O}^{v_1}$, and design the invariance loss as follows.

$$\ell_{invariant} = \frac{1}{N}\sum_{n=1}^{N} \hat{d}\big(\hat{\boldsymbol{o}}_n^{v_1}, \hat{\boldsymbol{o}}_n^{v_2}\big) \tag{3}$$

where $\hat{\boldsymbol{o}}_n^{v_1}$ and $\hat{\boldsymbol{o}}_n^{v_2}$ are the reordered object assignments of the two augmented point clouds for the nth point. The distance function $\hat{d}()$ is flexible to use $L1$, $L2$ or cross-entropy. Ultimately, this loss drives the estimated object masks to be invariant with different views of input point clouds.

Notably, unlike existing self-supervised learning [11] which usually uses invariance prior for better latent representations, here we aim to generalize the segmentation strategy to similar yet static objects.

3.3 Iterative Optimization of Object Segmentation and Motion Estimation

With the designed geometry consistency loss functions, the object segmentation network is optimized from scratch by the combined loss: $\ell_{seg} = \ell_{dynamic} + \ell_{smooth} + \ell_{invariant}$. For efficiency, the auxiliary self-supervised scene flow network FlowStep3D [34] is independently trained by its own losses until convergence. Intuitively, with better and better object masks estimated, the estimated scene flow is also expected to be improved further if we use the masks properly. To this end, we propose the following Algorithm 1 to iteratively improve object segmentation and motion estimation.

Algorithm 1 Iterative optimization of object segmentation and scene flow estimation. Assume the whole train split has S point cloud pairs: $\{(\boldsymbol{P}^t, \boldsymbol{P}^{t+1})_1 \cdots (\boldsymbol{P}^t, \boldsymbol{P}^{t+1})_S\}$.

Stage 0: Initial scene flow estimation.

• Independently and fully train the self-supervised scene flow network on the whole training data split, and obtain reasonable scene flow estimations: $\{(\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t)_1 \cdots (\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t)_S\}$.

for number of iteration rounds R **do**

Stage 1: Object segmentation optimization.

• Train the object segmentation network using ℓ_{seg} for a total E epochs on the whole training split: $\{(\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t)_1 \cdots (\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t)_S\}$.

• Estimate reasonable object masks: $\{(\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{O}^t, \boldsymbol{O}^{t+1})_1 \cdots (\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{O}^t, \boldsymbol{O}^{t+1})_S\}$.

Stage 2: Scene flow improvement.

• For each pair of data $(\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t, \boldsymbol{O}^t, \boldsymbol{O}^{t+1})$, by drawing insights from the classical ICP [2], we propose an **object-aware ICP** algorithm to estimate new scene flow $\hat{\boldsymbol{M}}^t$ for point cloud $\boldsymbol{P}^t$.

• Update the new scene flow for next round training:

$\{(\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t)_1 \cdots (\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t)_S\} \leftarrow \{(\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \hat{\boldsymbol{M}}^t)_1 \cdots (\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \hat{\boldsymbol{M}}^t)_S\}$

Empirically, setting the total number of rounds R to be 2 or 3 has a good trade off between accuracy and training efficiency. Due to the limited space, details of object-aware ICP algorithm are in Appendix A.2. We exclude the invariance loss $l_{invariant}$ from object segmentation optimization stage in the early rounds so that the networks can focus on moving objects in training and produce better scene flows, and then add $l_{invariant}$ back in the final round. Detailed analysis is in Appendix A.5.

4 Experiments

Our method is evaluated on four different application scenarios: 1) part instance segmentation of articulated objects on SAPIEN dataset [75], 2) object segmentation of indoor scenes on our own synthetic dataset, 3) object segmentation of real-world outdoor scenes on KITTI-SF dataset [48], and 4) object segmentation on the sparse yet large-scale LiDAR based KITTI-Det [19] and SemanticKITTI [5] datasets. For evaluation metrics, we follow [51] and report the **F1-score**, **Precision**, and **Recall**

Table 1: Quantitative results of our method and baselines on the SAPIEN dataset.

		AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
Supervised Methods	PointNet++ [57]	-	-	-	-	-	51.2	65.0
	MeteorNet [43]	-	-	-	-	-	45.7	60.0
	DeepPart [80]	-	-	-	-	-	53.0	67.0
	MBS [27]	-	-	-	-	-	67.3	77.0
	OGC$_{sup}$	66.1	48.7	62.0	54.6	71.7	66.8	77.1
Unsupervised Motion Segmentation	TrajAffn [52]	6.2	14.7	22.0	16.3	34.0	45.7	60.1
	SSC [51]	9.5	20.4	28.2	20.9	43.5	50.6	65.9
Unsupervised Methods	WardLinkage [30]	17.4	26.8	40.1	36.9	43.9	49.4	62.2
	DBSCAN [17]	6.3	13.4	20.4	13.9	37.9	34.2	51.4
	OGC(Ours)	**55.6**	**50.6**	**65.1**	**65.0**	**65.2**	**60.9**	**73.4**

with an IoU threshold of 0.5. In addition, we report the Average Precision (**AP**) score following COCO dataset [40] and the Panoptic Quality (**PQ**) score defined in [33]. The mean Intersection over Union (**mIoU**) score and the Rand Index (**RI**) score implemented in [27] are also included. Note that, all metrics are computed in a class-agnostic manner.

4.1 Evaluation on SAPIEN Dataset

The SAPIEN dataset [75] provides 720 simulated articulated objects with part instance level annotations. Each object has 4 sequential scans. The part instances have different articulating (moving) states. We follow [27] to use the training data generated from [81]. In particular, there are 82092 pairs of point clouds for training, 2880 single point cloud frames for testing. Each point cloud is downsampled to 512 points in both training and testing.

Since there is no existing unsupervised method for multi-object segmentation on 3D point clouds, we firstly implement two classical clustering methods: WardLinkage [30] and DBSCAN [17] to directly group 3D points from single point clouds into objects. Secondly, we implement two classical motion segmentation methods: TrajAffn [52] and SSC [51]. Note that, these two methods take the same estimated scene flows of FlowStep3D as input, while our method uses the estimated scene flows during training only, but takes single point clouds as input during testing. In addition, we also include the excellent results of several fully-supervised methods (PointNet++ [57], MeteorNet [43], DeepPart [80]) reported in MBS [27]. Their experimental details can be found in MBS [27]. Lastly, we train our object segmentation network using single point clouds with full annotations, denoted as OGC$_{sup}$. All implementation details are in Appendix A.4.

Analysis: As shown in Table 1, our OGC surpasses the classical clustering based and motion segmentation methods by large margins on all metrics, showing the advantage of our method in fully leveraging both the motion patterns and various types of geometry consistency. Compared with the fully supervised baselines, our method is only inferior to the strong MBS [27] and OGC$_{sup}$. However, we observe that our OGC actually shows a higher precision score than OGC$_{sup}$, primarily because our method tends to learn better objectness thanks to a combination of motion pattern and smoothness constraints and avoid dividing a single object into pieces. Figure 3 shows qualitative results.

4.2 Evaluation on OGC-DR / OGC-DRSV Datasets

We further evaluate our method to segment objects in indoor 3D scenes. Considering that the existing dataset FlyingThings3D [47] tends to have unrealistically cluttered scenes with severely fragmented objects and it is originally introduced for scene flow estimation, we turn to synthesize a new dynamic room dataset, called **OGC-DR**, that suits both scene flow estimation and object segmentation. In particular, we follow [54] to randomly place $4 \sim 8$ objects belonging to 7 classes of ShapeNet [7] {chair, table, lamp, sofa, cabinet, bench, display} into each room. In total, we create 3750, 250, and 1000 indoor rooms (scenes) for training/validation/test splits. In each scene, we create rigid dynamics by applying continuous random transformations to each object and record 4 sequential frames for evaluation. Each point cloud frame is downsampled to 2048 points. Note that, we follow [13] to split different object instances for train/val/test sets.

Table 2: Quantitative results of our method and baselines on our OGC-DR/OGC-DRSV dataset.

		AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
Supervised Method	OGC_{sup}	90.7 / 86.3	82.6 / 78.8	87.6 / 85.0	83.7 / 82.2	92.0 / 88.0	89.2 / 83.9	97.7 / 97.1
Unsupervised Motion Segmentation	TrajAffn [52]	42.6 / 39.3	46.7 / 43.8	57.8 / 54.8	69.6 / 63.0	49.4 / 48.4	46.8 / 45.9	80.1 / 77.7
	SSC [51]	74.5 / 70.3	79.2 / 75.4	84.2 / 81.5	92.5 / 89.6	77.3 / 74.7	74.6 / 70.8	91.5 / 91.3
Unsupervised Methods	WardLinkage [30]	72.3 / 69.8	74.0 / 71.6	82.5 / 80.5	**93.9 / 91.8**	73.6 / 71.7	69.9 / 67.2	94.3 / 93.3
	DBSCAN [17]	73.9 / 71.9	76.0 / 76.3	81.6 / 81.8	85.8 / 79.1	77.8 / 84.8	74.7 / 80.1	91.5 / 93.5
	OGC(Ours)	**92.3 / 86.8**	**85.1 / 77.0**	**89.4 / 83.9**	85.6 / 77.7	**93.6 / 91.2**	**90.8 / 84.8**	**97.8 / 95.4**

Based on our OGC-DR dataset, we collect single depth scans every time step on the mesh models to generate another dataset, called Single-View OGC-DR (**OGC-DRSV**). All object point clouds in OGC-DRSV are severely incomplete due to self- and/or mutual occlusions, resulting in the new dataset significantly more challenging than OGC-DR. Each point cloud frame in OGC-DRSV is also downsampled to 2048 points. More details of these two datasets are in Appendix A.3.

Analysis: As shown in Table 2, our method outperforms all classical unsupervised methods including the clustering based and the motion segmentation based methods on OGC-DR. Since the synthetic rooms in OGC-DR all have complete 3D objects, and the generated point cloud sequences are of high quality. Therefore, our OGC even surpasses the supervised OGC_{sup}. This shows that the rigid dynamic motions can indeed provide sufficient supervision signals to identify objects. On OGC-DRSV, our method still achieves superior performance and demonstrates robustness to incomplete point clouds, although the scores are slightly lower than that on the full point cloud dataset OGC-DR (AP: 86.8 vs 92.3). Figure 3 shows qualitative results.

4.3 Evaluation on KITTI Scene Flow Dataset

We additionally evaluate our method on the challenging real-world outdoor KITTI Scene Flow (KITTI-SF) dataset. Officially, KITTI-SF dataset [48] consists of 200 (training) pairs of point clouds from real-world traffic scenes and an online hidden test for scene flow estimation. In our experiment, we train our pipeline on the first 100 pairs of point clouds, and then test on the remaining 100 pairs (200 single point clouds). We observe that in the 100 training pairs, the moving objects are only cars and trucks. Therefore, in the testing phase, we only keep the human annotations [1] of cars and trucks in every single frame to compute the scores. All other objects are treated as part of background. Note that, the whole background is not ignored, but counted as one object in our evaluation, and the cars and trucks can be static or moving. We find KITTI-SF is too challeging for the classical unsupervised methods, due to the extreme imbalance of 3D points between objects and background. Besides, the background and objects in KITTI-SF are always connected because of the Earth's gravity, while clustering-based WardLinkage and DBSCAN favor spatially separated objects. Therefore, we leverage the prior about ground planes in KITTI-SF to assist these methods. We detect and specially handle the ground planes, leaving above-ground points only for these methods to handle. Implementation details are in Appendix A.4.

Analysis: As shown in Table 3, our method obtains superior segmentation scores on the KITTI-SF dataset, being very close to our fully-supervised counterpart OGC_{sup}. This demonstrates the excellence of our method on real-world scenes. Figure 3 shows qualitative results.

Table 3: Quantitative results of our method and baselines on the KITTI-SF dataset.

		AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
Supervised Method	OGC_{sup}	62.4	52.7	65.1	63.4	67.0	67.3	95.0
Unsupervised Motion Segmentation	TrajAffn [52]	24.0	30.2	43.2	37.6	50.8	48.1	58.5
	SSC [51]	12.5	20.4	28.4	22.8	37.6	41.5	48.9
Unsupervised Methods	WardLinkage [30]	25.0	16.3	22.9	13.7	**69.8**	60.5	44.9
	DBSCAN [17]	13.4	22.8	32.6	26.7	42.0	42.6	55.3
	OGC(Ours)	**54.4**	**42.4**	**52.4**	**47.3**	58.8	**63.7**	**93.6**

4.4 Generalization to KITTI Detection and SemanticKITTI Datasets

Given our well trained model on KITTI-SF in Section 4.3, we directly test it on the popular KITTI 3D Object Detection (KITTI-Det) [19] and SemanticKITTI [5] benchmarks. Unlike the stereo-based

point clouds in KITTI-SF, point clouds in these two datasets are collected by LiDAR sensors and thus more sparse.

- **KITTI-Det** officially has 3712 point cloud frames for training, 3769 for validation. We only keep the ground truth object masks obtained from bounding boxes for the car category in each frame. All other objects are treated as part of background. For comparison, we download the official pretrained models of three fully-supervised methods PointRCNN [59], PV-RCNN [58] and Voxel-RCNN [14] to directly test on the validation split using the same settings as ours. In addition, we use the well trained OGC$_{sup}$ on KITTI-SF to directly test for comparison, denoted as OGC*$_{sup}$. We also train OGC$_{sup}$ on the training split (3712 frames) from scratch and test it on the remaining 3769 frames using the same evaluation settings, denoted as OGC$_{sup}$.
- **SemanticKITTI** officially has 11 sequences with annotations for training and another 11 sequences for online hidden test. We only keep ground truth objects of car and truck categories. The total 11 training sequences (23201 point cloud frames) are used for testing. Compared with KITTI-Det, SemanticKITTI holds $6\times$ more testing frames and covers more diverse scenes. Following the official split in [5], we also report the results on: i) sequences 00$\sim$07 and 09$\sim$10 (19130 frames), and ii) the sequence 08 (4071 frames) separately.

Table 4: Quantitative results on KITTI-Det (* denotes the model trained on KITTI-SF).

		AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
Supervised Methods	PointRCNN [59]	95.7	80.1	88.9	81.3	98.0	91.4	97.2
	PV-RCNN [58]	95.4	77.3	84.4	73.7	98.8	92.7	97.1
	Voxel-RCNN [14]	95.8	79.6	87.3	78.1	98.9	92.6	97.3
	OGC$_{sup}$	80.0	68.5	78.3	72.7	84.8	84.0	96.9
	OGC*$_{sup}$	51.4	41.0	49.1	43.7	56.0	66.2	91.0
Unsupervised Method	**OGC*(Ours)**	40.5	30.9	37.0	30.8	46.5	60.6	86.4

Analysis: As shown in Tables 4&5, our method can directly generalize to 3D object segmentation on sparse LiDAR point clouds with satisfactory results, also being close to the fully-supervised counterpart OGC*$_{sup}$. It is understandable that the other three fully-supervised models have a clear advantage over ours on KITTI-Det, because they are fully supervised and trained on the KITTI-Det training split (3712 frames) while ours does not. We hope that our method can serve the first baseline and inspire more advanced unsupervised methods in the future to close the gap. Figure 3 shows qualitative results.

Table 5: Quantitative results on SemanticKITTI (* denotes the model trained on KITTI-SF).

Sequences	Methods	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
00$\sim$10	OGC*$_{sup}$	53.8	41.3	48.1	40.1	60.0	68.3	90.0
	OGC*(Ours)	42.6	30.2	35.3	28.2	47.3	60.3	86.0
00$\sim$07 & 09$\sim$10	OGC*$_{sup}$	55.3	41.8	48.4	40.1	61.1	69.9	90.3
	OGC*(Ours)	43.6	30.5	35.5	28.1	48.2	62.1	86.3
08	OGC*$_{sup}$	49.4	39.2	46.6	40.0	55.8	60.3	88.3
	OGC*(Ours)	38.6	29.1	34.7	28.6	44.0	51.8	84.3

4.5 Ablation Study

(1) Geometry Consistency Losses: To validate the choice of our design, we firstly conduct three groups of ablative experiments on the SAPIEN dataset [75]: 1) only remove the dynamic rigid loss $\ell_{dynamic}$, 2) only remove the smoothness loss ℓ_{smooth}, and 3) only remove the invariance loss $\ell_{invariant}$. As shown in Table 6, combining the proposed three losses together gives the highest segmentation scores. Basically, the dynamic rigid loss serves to discriminate multiple objects from different motion patterns. Without it, the network tends to assign all points to a single object as the shortcut to minimize the other two losses. However, we observe in Table 6 that without $\ell_{dynamic}$, the network still works to some extend. This is because the synthetic SAPIEN dataset tends to have a number of point cloud frames with only 2 or 3 objects, thus assigning all points to a single object can still get plausible

Table 6: Ablation studies about loss designs on SAPIEN.

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
w/o $\ell_{dynamic}$	35.4	35.3	54.1	**91.1**	38.5	28.6	52.7
w/o ℓ_{smooth}	21.8	18.5	26.9	19.1	45.4	52.4	63.7
w/o $\ell_{invariant}$	48.9	46.1	61.3	61.9	60.7	57.9	70.3
Full OGC	**55.6**	**50.6**	**65.1**	65.0	**65.2**	**60.9**	**73.4**

scores. This issue is further validated by conducting additional ablation experiments on curated SAPIEN dataset. More details are in Appendix A.5.

In addition, we evaluate the robustness of our object segmentation method with regard to different types of motion estimations, and different hyperparameter and design choices of our smoothness loss ℓ_{smooth}. More results are in Appendix A.5.

(2) Iterative Optimization Algorithm: We also conduct ablative experiments to validate the effectiveness of our proposed Algorithm 1. We set the number of iterative rounds R as $\{1, 2, 3\}$. As shown in Table 7, after 2 rounds, satisfactory segmentation results can be achieved, although we expect better results after more rounds with longer training time. This shows that our iterative optimization algorithm can indeed fully leverage the mutual benefits between object segmentation and motion estimation.

Table 7: Iterative optimization on SAPIEN.

	Object Segmentation						
$\#R$	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
1	45.9	47.7	62.3	60.2	64.5	60.2	72.3
2	55.6	50.6	65.1	65.0	65.2	60.9	73.4
3	56.3	50.7	65.4	65.1	65.8	61.1	73.7

4.6 Pushing the Boundaries of Unsupervised Scene Flow Estimation and Segmentation

In addition to the improvement of object segmentation from our iterative optimization algorithm, the scene flow estimation can be naturally further improved from our estimated object masks as well. Given our well trained model on the KITTI-SF dataset in Section 4.3, we use the estimated object masks to further improve the scene flow estimation. As shown in Table 8, following the exact evaluation settings of FlowStep3D [34], our method, not surprisingly, significantly boosts the scene flow accuracy, surpassing the state-of-the-art unsupervised FlowStep3D [34] and other baselines in all metrics.

Table 8: Scene flow estimation on the KITTI-SF dataset.

	EPE3D↓	AccS↑	AccR↑	Outlier↓
Ego-motion [66]	41.54	22.09	37.21	80.96
PointPWC-Net [74]	25.49	23.79	49.57	68.63
FlowStep3D [34]	10.21	70.80	83.94	24.56
OGC(Ours)	**6.72**	**80.16**	**89.08**	**22.56**

In fact, our object segmentation backbone is also flexible to take the scene flow as input instead of point xyz to segment objects. This is commonly called motion segmentation. We replace the single point clouds by (estimated) scene flow vectors as our network inputs, and train the network from scratch using the same settings on the KITTI-SF dataset. As shown in Table 9, we can see that our network can still achieve superior results regardless of the modality of inputs, demonstrating the generality of our framework.

Table 9: Motion vs Points based segmentation on KITTI-SF.

Input	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
scene flow	47.3	41.2	50.2	**50.9**	49.6	56.0	89.3
point cloud	**54.4**	**42.4**	**52.4**	47.3	**58.8**	**63.7**	**93.6**

5 Conclusion

In this paper, we demonstrate for the first time that 3D objects can be accurately segmented using an unsupervised method from raw point clouds. Unlike the existing approaches which usually rely on a large amount of human annotations of every 3D object for training networks, we instead turn to leverage the diverse motion patterns over sequential point clouds as supervision signals to automatically discover the objectness from single point clouds. A series of loss functions are designed to preserve the object geometry consistency over spatial and temporal scales. Extensive experiments over multiple datasets including the extremely challenging outdoor scenes demonstrate the effectiveness of our method.

Broader Impact: The proposed OGC learns 3D objects from raw point clouds without requiring human annotations for supervision. We showcase the effectiveness for some basic applications including object part instance segmentation, indoor object segmentation and outdoor vehicle identification. We also believe that our method can be general for other domains such as AR/VR.

Acknowledgements: This work was partially supported by Shenzhen Science and Technology Innovation Commission (JCYJ20210324120603011).

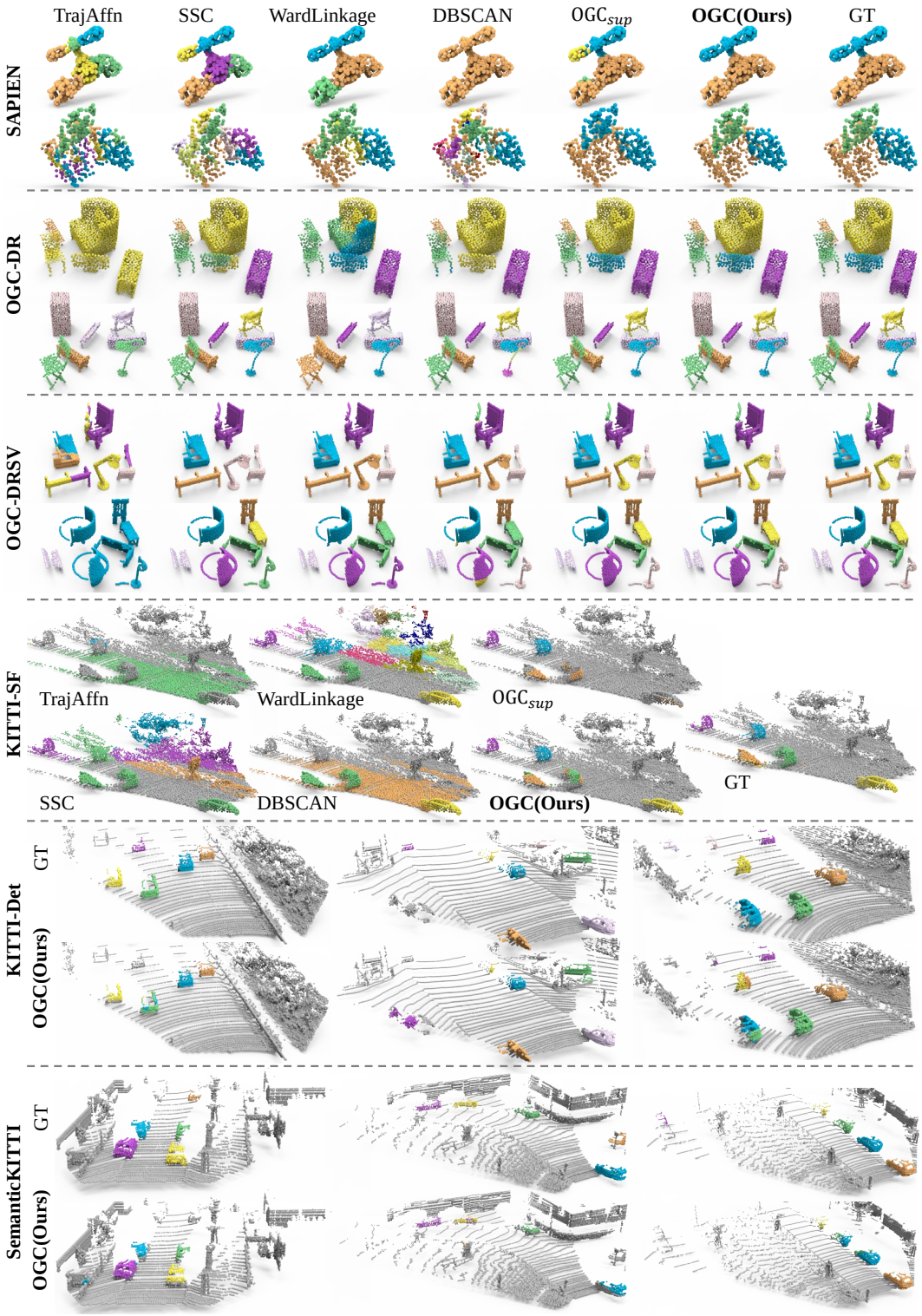

Figure 3: Qualitative results on various datasets. More qualitative results can be found in Appendix A.7 and our video demo: https://youtu.be/dZBjvKWJ4K0

.

References

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. *IJCV*, 126:961–972, 2018. 7

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *TPAMI*, (5):698 – 700, 1987. 5, 17, 19

[3] S. A. Baur, D. J. Emmerichs, F. Moosmann, P. Pinggera, B. Ommer, and A. Geiger. SLIM: Self-Supervised LiDAR Scene Flow and Motion Segmentation. *ICCV*, 2021. 1, 3

[4] A. Behl, D. Paschalidou, S. Donné, and A. Geiger. PointFlowNet: Learning Representations for 3D Scene Flow Estimation from Point Clouds. *CVPR*, 2019. 3

[5] 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. *ICCV*, 2019. 5, 7, 8

[6] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. MONet: Unsupervised Scene Decomposition and Representation. *arXiv:1901.11390*, 2019. 3

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv*, 2015. 6

[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. *CVPR*, 2017. 3

[9] B. Cheng, A. G. Schwing, and A. Kirillov. Per-Pixel Classification is Not All You Need for Semantic Segmentation. *NeurIPS*, 2021. 3

[10] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang. SegFlow: Joint Learning for Video Object Segmentation and Optical Flow. *ICCV*, 2017. 3

[11] J. H. Cho, U. Mall, K. Bala, and B. Hariharan. PiCIE: Unsupervised Semantic Segmentation using Invariance and Equivariance in Clustering. *CVPR*, 2021. 5

[12] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. *CVPR*, 2019. 3

[13] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. *ECCV*, 2016. 6

[14] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. *AAAI*, 2021. 8

[15] D. Ding, F. Hill, A. Santoro, M. Reynolds, and M. Botvinick. Attention over learned object embeddings enables complex visual reasoning. *NeurIPS*, 2021. 3

[16] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. *NIPS*, 2016. 3

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 1996. 6, 7, 19

[18] D. Fortun, P. Bouthemy, and C. Kervrann. Optical flow modeling and computation: A survey. *CVIU*, 134(1-21), 2015. 3

[19] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. *CVPR*, 2012. 5, 7

[20] Z. Gojcic, O. Litany, A. Wieser, L. J. Guibas, and T. Birdal. Weakly Supervised Learning of Rigid 3D Scene Flow. *CVPR*, 2021. 2, 4

[21] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal. Learning multiview 3D point cloud registration. *CVPR*, 2020. 4, 17

[22] B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. *CVPR*, 2018. 3

[23] K. Greff, R. L. Kaufman, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. Multi-object representation learning with iterative variational inference. *ICML*, 2019. 3

[24] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds. *CVPR*, 2019. 3

[25] J. T. Hsieh, B. Liu, D. A. Huang, L. Fei-Fei, and J. C. Niebles. Learning to decompose and disentangle representations for video prediction. *NIPS*, 2018. 3

[26] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. *CVPR*, 2020. 3

[27] J. Huang, H. Wang, T. Birdal, M. Sung, F. Arrigoni, S. M. Hu, and L. Guibas. MultiBodySync: Multi-Body Segmentation and Motion Estimation via 3D Scan Synchronization. *CVPR*, 2021. 3, 6

[28] C. Jiang, D. P. Paudel, D. Fofi, Y. Fougerolle, and C. Demonceaux. Moving Object Detection by 3D Flow Field Analysis. *TITS*, 22(4):1950–1963, 2021. 1

[29] J. Jiang, S. Janghorbani, G. D. Melo, and S. Ahn. SCALOR: Generative World Models with Scalable Object Representations. *ICLR*, 2020. 3

[30] J. H. W. Jr. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58:236–244, 1963. 6, 7, 19

[31] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 4, 17

[32] T. Kipf, G. F. Elsayed, A. Mahendran, A. Stone, S. Sabour, G. Heigold, R. Jonschkowski, A. Dosovitskiy, and K. Greff. Conditional Object-Centric Learning from Video. *ICLR*, 2022. 3

[33] A. Kirillov, K. He, R. B. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. *CVPR*, 2019. 6

[34] Y. Kittenplon, Y. C. Eldar, and D. Raviv. FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation. *CVPR*, 2021. 2, 3, 4, 5, 9, 16, 17

[35] A. R. Kosiorek, H. Kim, I. Posner, and Y. W. Teh. Sequential attend, infer, repeat: Generative modelling of moving objects. *NeurIPS*, 2018. 3

[36] H. W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 5, 17

[37] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. *CVPR*, 2019. 1, 3

[38] B. Li, T. Zhang, and T. Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. *RSS*, 2016. 3

[39] R. Li, G. Lin, and L. Xie. Self-Point-Flow: Self-Supervised Scene Flow Estimation from Point Clouds with Optimal Transport and Random Walk. *CVPR*, 2021. 3

[40] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *ECCV*, 2014. 6

[41] R. Liu, Z. Wu, S. X. Yu, and S. Lin. The Emergence of Objectness : Learning Zero-Shot Segmentation from Videos. *NeurIPS*, 2021. 3

[42] X. Liu, C. R. Qi, and L. J. Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. *CVPR*, 2019. 3, 4

[43] X. Liu, M. Yan, and J. Bohg. MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences. *ICCV*, 2019. 3, 6

[44] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-Centric Learning with Slot Attention. *NeurIPS*, 2020. 3

[45] X. Lu, W. Wang, J. Shen, Y.-W. Tai, D. J. Crandall, and S. C. H. Hoi. Learning Video Object Segmentation From Unlabeled Videos. *CVPR*, 2020. 3

[46] C. Luo, X. Yang, and A. Yuille. Self-Supervised Pillar Motion Learning for Autonomous Driving. *CVPR*, 2021. 3

[47] N. Mayer, E. Ilg, H. Philip, D. Cremers, A. Dosovitskiy, and T. Brox. A Large Dataset to Train Convolutional Networks for Disparity , Optical Flow , and Scene Flow Estimation. *CVPR*, 2016. 6

[48] M. Menze, A. Geiger, and T. Mpi. Object Scene Flow for Autonomous Vehicles. *CVPR*, 2015. 5, 7

[49] M. Minderer, C. Sun, R. Villegas, F. Cole, K. Murphy, and H. Lee. Unsupervised Learning of Object Structure and Dynamics from Videos. *NeurIPS*, 2019. 3

[50] H. Mittal, B. Okorn, and D. Held. Just go with the flow: Self-supervised scene flow estimation. *CVPR*, 2020. 3

[51] U. M. Nunes and Y. Demiris. 3D motion segmentation of articulated rigid bodies based on RGB-D data. *BMVC*, 2018. 5, 6, 7, 19

[52] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *TPAMI*, 36(6):1187–1200, 2014. 6, 7, 19

[53] B. Ouyang and D. Raviv. Occlusion guided scene flow estimation on 3D point clouds. *CVPR*, 2021. 3

[54] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional Occupancy Networks. *ECCV*, 2020. 6, 18

[55] G. Puy, A. Boulch, and R. Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. *ECCV*, 2020. 3

[56] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, 2017. 1

[57] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NIPS*, 2017. 2, 3, 5, 6, 16

[58] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. *CVPR*, 2019. 8

[59] S. Shi, X. Wang, and H. Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *CVPR*, 2019. 8

[60] W. Shi, Ragunathan, and Rajkumar. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. *CVPR*, 2020. 3

[61] D. Sun, E. B. Sudderth, and M. J. Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. *NIPS*, 2010. 3

[62] D. Sun, E. B. Sudderth, and M. J. Black. Layered segmentation and optical flow estimation over time. *CVPR*, 2012. 3

[63] J. Sun, Y. Dai, X. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen. Efficient Spatial-Temporal Information Fusion for LiDAR-Based 3D Moving Object Segmentation. *IROS*, 2022. 2

[64] H. Thomas, B. Agro, M. Gridseth, J. Zhang, and T. D. Barfoot. Self-Supervised Learning of Lidar Segmentation for Autonomous Indoor Navigation. *ICRA*, 2021. 3

[65] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. *ICCV*, 2019. 3

[66] I. Tishchenko, S. Lombardi, M. R. Oswald, and M. Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. *3DV*, 2020. 9

[67] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *NIPS*, 2017. 3, 16

[68] T. Vu, K. Kim, T. M. Luu, X. T. Nguyen, and C. D. Yoo. SoftGroup for 3D Instance Segmentation on Point Clouds. *CVPR*, 2022. 3

[69] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt. A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization. *Psychological Bulletin*, 138(6):1172–1217, 2012. 1

[70] W. Wang, R. Yu, Q. Huang, and U. Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. *CVPR*, 2018. 1, 3

[71] Z. Wang, C. Shen, R. Li, C. Zhang, and G. Lin. RigidFlow: Self-Supervised Scene Flow Learning on Point Clouds by Local Rigidity Prior. *CVPR*, 2022. 2

[72] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou. PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds. *CVPR*, 2021. 3

[73] M. Wertheimer. Untersuchungen zur Lehre yon der Gestalt. *Psychologische Forschung*, (4):301–350, 1923. 1

[74] W. Wu, Z. Y. Wang, Z. Li, W. Liu, and L. Fuxin. PointPWC-Net: Cost Volume on Point Clouds for (Self-)Supervised Scene Flow Estimation. *ECCV*, 2020. 2, 3, 9

[75] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. SAPIEN: A SimulAted Part-based Interactive ENvironment. *CVPR*, 2020. 5, 6, 8

[76] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. *NeurIPS*, 2019. 1, 3

[77] C. Yang, H. Lamdouar, E. Lu, A. Zisserman, and W. Xie. Self-supervised Video Object Segmentation by Motion Grouping. *ICCV*, 2021. 2, 3

[78] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto. Unsupervised moving object detection via contextual information separation. *CVPR*, 2019. 2

[79] Y. Yang and B. Yang. Promising or Elusive? Unsupervised Object Segmentation from Real-world Single Images. *arXiv:2210.02324*, 2022. 3

[80] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas. Deep part induction from articulated object pairs. *SIGGRAPH Asia*, 37(6), 2018. 3, 6

[81] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. J. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 2016. 6

[82] Y. Zeng, Y. Qian, Z. Zhu, J. Hou, H. Yuan, and Y. He. CorrNet3D: Unsupervised End-to-end Learning of Dense Correspondence for 3D Point Clouds. *CVPR*, 2021. 3

[83] M. Zhai, X. Xiang, N. Lv, and X. Kong. Optical Flow and Scene Flow Estimation: A Survey. *Pattern Recognition*, 2021. 3

[84] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *CVPR*, 2018. 3

[85] D. Zoran, R. Kabra, and D. J. Rezende. PARTS: Unsupervised segmentation with slots , attention and independence maximization. *ICCV*, 2021. 3

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix A.6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Error bars are not reported because it would be too computationally expensive
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Network Architecture

We provide a detailed description of our object segmentation network and the auxiliary self-supervised scene flow estimator.

(1) Object Segmentation Network

As shown in Figure 4, the network takes a single point cloud with N points as input. It consists of Set Abstraction (SA) modules from PointNet++ [57] to extract per-point features for the downsampled point cloud with N' points. Feature Propagation (FP) modules are applied subsequently to obtain per-point embeddings for all N points. Given the intermediate features for the N' points and the K learnable queries, the standard Transformer decoder [67] is used to compute the K object embeddings, each of which is expected to represent an object in the input point cloud. An MLP layer is added to reduce the dimension of object embeddings to be the same as point embeddings obtained from the PointNet++ backbone. At last, we obtain each (soft) binary mask $\boldsymbol{O}_k^t$ via a dot product between the kth object embedding and per-point embeddings. For each point, a softmax activation function is applied to normalize its probabilities of being assigned to different objects.

In practice, the downsampling rate and point neighborhood selection in the PointNet++ backbone are adapted to the point densities and sizes of different datasets, as shown in Table 10. The embedding dimension from the Transformer decoder is set as 128 in all datasets.

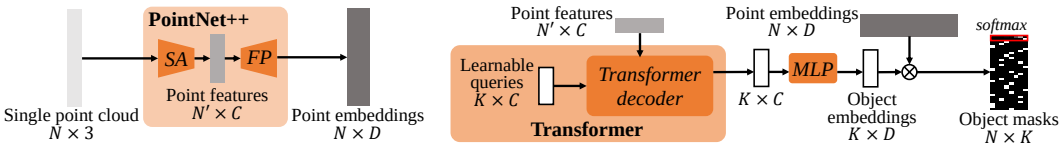

Figure 4: Detailed architecture of our object segmentation network.

Table 10: Configuration of the PointNet++ backbone in our object segmentation network. s denotes the point cloud downsampling/upsampling rate. k controls the K nearest neighbors selected within a ball with radius r. c denotes the first input and the following output channels of MLP layers. In SA modules, the level 1-1 and 1-2 compose a multi-scale grouping (MSG) [57] with outputs concatenated. In FP modules, the multi-level point features from SA are concatenated as inputs.

		SAPIEN / OGC-DR / OGC-DRSV				KITTI-SF / KITTI-Det / SemanticKITTI			
	level	s	k	r	c	s	k	r	c
SA	1-1	1/2	64	0.1(0.05)	{3,64,64,64}	1/4	64	1.0	{3,32,32,32}
	1-2	1/2	64	0.2(0.1)	{3,64,64,128}	1/4	64	2.0	{3,32,32,64}
	2	1/4	64	0.4(0.2)	{192,128,128,256}	1/8	64	4.0	{96,64,64,128}
	3					1/16	64	8.0	{128,128,128,256}
FP	3					1/8			{384,128,128}
	2	1/2			{448,256,128}	1/4			{224,64,64}
	1	1			{131,128,128,64}	1			{67,64,64,64}

(2) Self-Supervised Scene Flow Estimator

We use the existing FlowStep3D as our self-supervised scene flow estimator. This method extracts per-point features via a PointNet++ backbone from two input point cloud frames separately. Then it adopts a recurrent architecture to refine the scene flow predictions iteratively. We refer readers to [34] for more details. On SAPIEN and OGC-DR / OGC-DRSV datasets, with smaller scene sizes and fewer input points, we remove the last SA module with 1/32 downsampling rate and reduce the number of nearest neighbors as half of its original choice in all modules.

A.2 Object-Aware ICP Algorithm

In Algorithm 2, we present our object-aware ICP (Iterative Closest Point) algorithm.

Algorithm 2 Object-aware ICP algorithm. Assume each training sample contains a pair of point clouds and scene flow estimations ($\boldsymbol{P}^t, \boldsymbol{P}^{t+1}, \boldsymbol{M}^t \in \mathbb{R}^{N\times 3}$), and object masks ($\boldsymbol{O}^t, \boldsymbol{O}^{t+1} \in \mathbb{R}^{N\times K}$) obtained from a trained object segmentation network.

Step 1: Match the individual masks in $\boldsymbol{O}^t$ and $\boldsymbol{O}^{t+1}$.

• Use the estimated scene flows to warp the first point cloud: $\boldsymbol{P}^t_w = \boldsymbol{P}^t + \boldsymbol{M}^t$, the warped $\boldsymbol{P}^t_w$ naturally inherits the per-point object masks from $\boldsymbol{P}^t$: $\boldsymbol{O}^t_w = \boldsymbol{O}^t$.

• Compute another object masks $\hat{\boldsymbol{O}}^{t+1}$ for the second point cloud $\boldsymbol{P}^{t+1}$ using the nearest-neighbor interpolation from $(\boldsymbol{P}^t_w, \boldsymbol{O}^t_w)$.

• Leverage the Hungarian algorithm [36] to one-one match the individual masks in $\hat{\boldsymbol{O}}^{t+1}$ and $\boldsymbol{O}^{t+1}$ according to the object pair-wise IoU scores.

• Reorder the masks in $\boldsymbol{O}^{t+1}$ to align with $\hat{\boldsymbol{O}}^{t+1}$, thus aligh with $\boldsymbol{O}^t$.

Step 2: Iteratively refine the rigid scene flow estimations

• Compute the per-point object consistency scores $\boldsymbol{O} \in \mathbb{R}^{N\times N}$ between $\boldsymbol{P}^t$ and $\boldsymbol{P}^{t+1}$: $\boldsymbol{O} = \boldsymbol{O}^t(\boldsymbol{O}^{t+1})^\intercal$.

for number of iterations I **do**

• Compute the per-point soft correspondence scores $\boldsymbol{C} \in \mathbb{R}^{N\times N}$ between $\boldsymbol{P}^t$ and $\boldsymbol{P}^{t+1}$ based on the nearest-neighbor (closest point) distances, where

$$\boldsymbol{C}_{ij} = exp(-\boldsymbol{\delta}_{ij}/\tau), \boldsymbol{\delta}_{ij} = \left\| \boldsymbol{p}^t_i + \boldsymbol{m}^t_i - \boldsymbol{p}^{t+1}_j \right\|_2$$

• Filter the per-point correspondence scores by the object consistency scores: $\boldsymbol{C} = \boldsymbol{C} * \boldsymbol{O}$.

• Update the scene flows $\boldsymbol{M}^t$ from the object-aware soft correspondences, where

$$\boldsymbol{m}^t_i = \frac{\sum_{j=1}^N \boldsymbol{C}_{ij}(\boldsymbol{p}^{t+1}_j - \boldsymbol{p}^t_i)}{\sum_{j=1}^N \boldsymbol{C}_{ij}}$$

• For the kth object, retrieve its (soft) binary mask $\boldsymbol{O}^t_k$, and then feed the tuple $\{\boldsymbol{P}^t, \boldsymbol{P}^t + \boldsymbol{M}^t, \boldsymbol{O}^t_k\}$ into weighted-Kabsch [31; 21] algorithm, estimating its transformation matrix $\boldsymbol{T}_k$.

• Update the scene flows $\boldsymbol{M}^t$ from the estimated transformations, where:

$$\boldsymbol{m}^t_i = \Big(\sum_{k=1}^K o^t_{ik} \cdot (\boldsymbol{T}_k \circ \boldsymbol{p}^t_i) \Big) - \boldsymbol{p}^t_i$$

Return the scene flows $\boldsymbol{M}^t$ from the last iteration.

In the iterative optimization, as the scene flow estimations gradually approach more consistent and accurate values, the number of iterations I in the object-aware ICP can be reduced for efficiency. We set I in the object-aware ICP as {20, 10, 5} in the round {1, 2, 3} of the iterative optimization.

Table 11: Scene flow estimation on KITTI-SF benchmark.

	EPE3D↓	AccS↑	AccR↑	Outlier↓
FlowStep3D [34]	10.21	70.80	83.94	24.56
Weighted Kabsch [31; 21]	9.31	71.01	81.20	28.75
Object-aware ICP	**6.72**	**80.16**	**89.08**	**22.56**

Compared to the weighted-Kabsch [31; 21] algorithm, our object-aware ICP algorithm takes two frames as input to correct the inconsistency in the flows. As shown in Table 11, our algorithm obtains a larger improvement for scene flow estimations. In general, our algorithm is an extension of the classical ICP [2] to 3D scenes with multiple rigid objects. Our algorithm can be naturally implemented in a batch-wise manner, without sacrificing the optimization speed of the network.

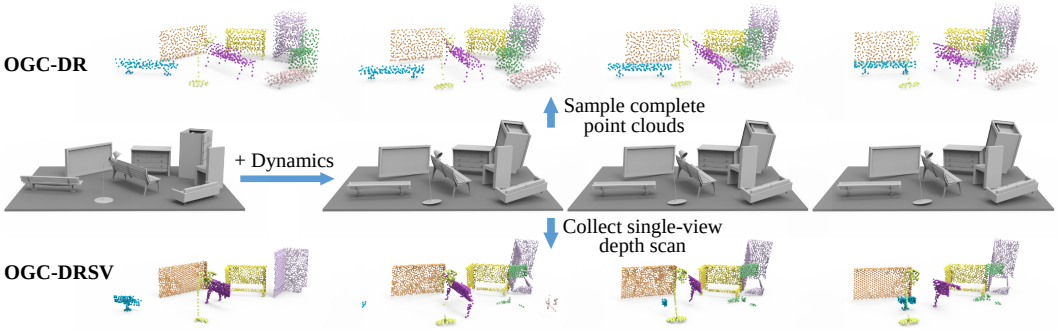

Figure 5: Illustration of the data generation process for our OGC-DR dataset.

A.3 OGC-DR and OGC-DRSV Datasets

Here we provide details about the data generation of our OGC-DR and OGC-DRSV datasets. Following [54], we first generate 5000 static scenes with $4 \sim 8$ objects. For a single scene, the ratio of width and length of the ground plane is uniformly sampled between $0.6 \sim 1.0$. For each object in a scene, its scale is sampled from $0.2 \sim 0.45$. The object rotation angles around the vertical y-axis are randomly sampled from $-180^\circ \sim 180^\circ$. Unlike [54], we do not keep walls and ground planes in the generated raw point clouds since these textureless surfaces create intractable ambiguities for self-supervised scene flow estimation. In fact, it is trivial to detect and remove them via plane fitting in real-world indoor scenes. The walls and ground planes are simply used to place all objects in a more realistic manner.

We then create rigid dynamics for the objects. For each object in a scene, we sample a rigid transformation relative to its pose in the previous frame. In particular, we first uniformly sample an angle from $-10^\circ \sim 10^\circ$ rotated by y/x/z axis with the probability of $\{0.6, 0.2, 0.2\}$ respectively. Afterwards, we uniformly sample a translation only on the x-z plane from the range of $-0.04 \sim 0.04$ for each object, ensuring that the object is always on the ground. Note that, we reject all samples which have objects overlapped or being out of the scene boundary.

The last point cloud sampling step varies for the two datasets. For OGC-DR, we directly sample point clouds from the surfaces of complete mesh models, while for OGC-DRSV, we collect single-view depth scans on mesh models. Note that for both datasets, the point sampling is independently conducted on each frame in a scene. Therefore, there is no exact point correspondences between consecutive frames. This setting is consistent with the scene flow estimation task in general real-world scenes. Figure 5 illustrates the complete data generation process.

A.4 Additional Implementation Details

(1) Data Preparation

SAPIEN: In SAPIEN, each scene (an articulated object) has 4 sequential scans. During training, we leverage consecutive frame pairs (both forward and backward) only, because the self-supervised scene flow estimator can hardly handle rapid motions. Therefore, each object has 6 pairs of point clouds. Given 13682/2356 objects in training/validation splits, we get 82092 training and 14136 validation frame pairs. The 720 objects in testing split contribute 2880 individual frames for evaluation.

OGC-DR/OGC-DRSV: Similar to SAPIEN, each scene in OGC-DR/OGC-DRSV holds 4 sequential frames. The 3750/250 scenes in training/validation splits give 22500 training and 1500 validation frame pairs, and the 1000 scenes in the testing split provide 4000 testing frames.

KITTI-SF: The 100 pairs of point clouds in the training split of KITTI-SF contribute 200 training pairs (both forward and backward), and the other 100 pairs in testing split provide 200 individual frames for evaluation. A tricky problem in KTTI-SF is the scene flow estimation for the ground. The textureless ground poses intractable ambiguities for the self-supervised scene flow estimator. However, we cannot simply remove the ground by applying a height threshold, because the background points above the ground will no longer be spatially connected, thus breaking the assumption behind our geometry smoothness regularizer ℓ_{smooth}. To address this issue, we apply the self-supervised scene

flow estimator to points above the ground only. Meanwhile, we use the classical ICP [2] algorithm onto points above the ground and regard the fitted transformation as motions for ground points (*i.e.*, the camera ego-motion). Although this solution relies on an assumption that static background points dominate the scene, our object-aware ICP algorithm can empirically alleviate potential errors in the iterative optimization.

(2) Hyperparameter Selection

Geometry Smoothness Regularization: We choose $L1$ for the distance function $d()$ given its less sensitivity to "outliers", *i.e.*, adjacent points belonging to different objects. As shown in Table 12, we select two groups of neighboring points from two different scales (denoted by $\{k_1, r_1\}$ and $\{k_2, r_2\}$) and weight them by {3.0, 1.0} in ℓ_{smooth}.

Geometry Invariance Loss: For the distance function $\hat{d}()$ here, $L1$, $L2$ and cross-entropy are all theoretically reasonable. We choose $L2$ for the best performance. The transformation for augmentation comprises a scale factor uniformly sampled from $0.95 \sim 1.05$ and a rotation around the vertical y-axis sampled from $-180 \sim 180°$. On KITTI-SF dataset, we add an x-z translation sampled from $-1 \sim 1$ and a y translation sampled from $-0.1 \sim 0.1$.

Network Training: We adopt the Adam optimizer with a learning rate of 0.001 and train on SAPIEN/OGC-DR/KITTI-SF datasets for 40/40/200 epochs, respectively. The batch size is set as 32/8/4 on each dataset to fill in the whole memory of a single RTX3090 GPU. The three losses $\ell_{dynamic}$, ℓ_{smooth} and $\ell_{invariant}$ are weighted by {10.0, 0.1, 0.1}. On SAPIEN and OGC-DR, since $\ell_{invariant}$ can slow down the convergence, we first train 20 epochs without it and then add it back. We also find that ℓ_{smooth} occasionally overwhelms the initial iterations of training, causing network predictions to collapse and assign all points to a single object. Therefore, we empirically disable ℓ_{smooth} before iterating through the initial 2000/2000/200 samples on SAPIEN/OGC-DR/KITTI-SF datasets.

Table 12: The choices of neighboring points in the geometry smoothness regularization. k controls the K nearest neighbors selected within a ball with radius r.

SAPIEN / OGC-DR				KITTI-SF			
k_1	r_1	k_2	r_2	k_1	r_1	k_2	r_2
8	0.1(0.02)	16	0.2(0.04)	32	1.0	64	2.0

(3) Baseline methods on KITTI-SF

Since the KITTI-SF dataset is too challeging for the classical unsupervised methods, we leverage the prior about ground planes in the dataset to improve baseline methods. First, we detect and temporarily remove the ground plane, letting the baseline algorithms to segment above-ground points only. For TrajAffn and SSC, we can use motion information to merge the ground points with above-ground segments that are likely to be part of the static background. To do this, we employ the Kabsch algorithm to estimate the rigid transformation of the ground. Then the above-ground segments whose motions are well fitted by the ground's transformation will be incorporated. For WardLinkage and DBSCAN, the ground is treated as a separate segment. We conduct ablation studies to validate the use of ground plane prior for these baseline methods on the KITTI-SF dataset, as shown in Table 13 and Figure 6. After using the ground plane prior, SSC and WardLinkage gain remarkable improvements both quantitatively and qualitatively. For TrajAffn and DBSCAN, although the quantitative performance gain is not significant, we find that their qualitative results become more meaningful.

Table 13: Ablation studies about the ground plane prior for baseline methods on KITTI-SF.

	use ground plane prior	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
TrajAffn [52]		30.4	34.7	42.7	40.5	45.3	49.0	83.5
	✓	24.0	30.2	43.2	37.6	50.8	48.1	58.5
SSC [51]		2.9	5.2	7.5	6.2	9.5	19.3	33.2
	✓	12.5	20.4	28.4	22.8	37.6	41.5	48.9
WardLinkage [30]		1.3	2.4	3.8	2.2	14.3	26.8	15.7
	✓	25.0	16.3	22.9	13.7	69.8	60.5	44.9
DBSCAN [17]		14.8	29.9	32.9	46.5	25.4	31.3	84.8
	✓	13.4	22.8	32.6	26.7	42.0	42.6	55.3

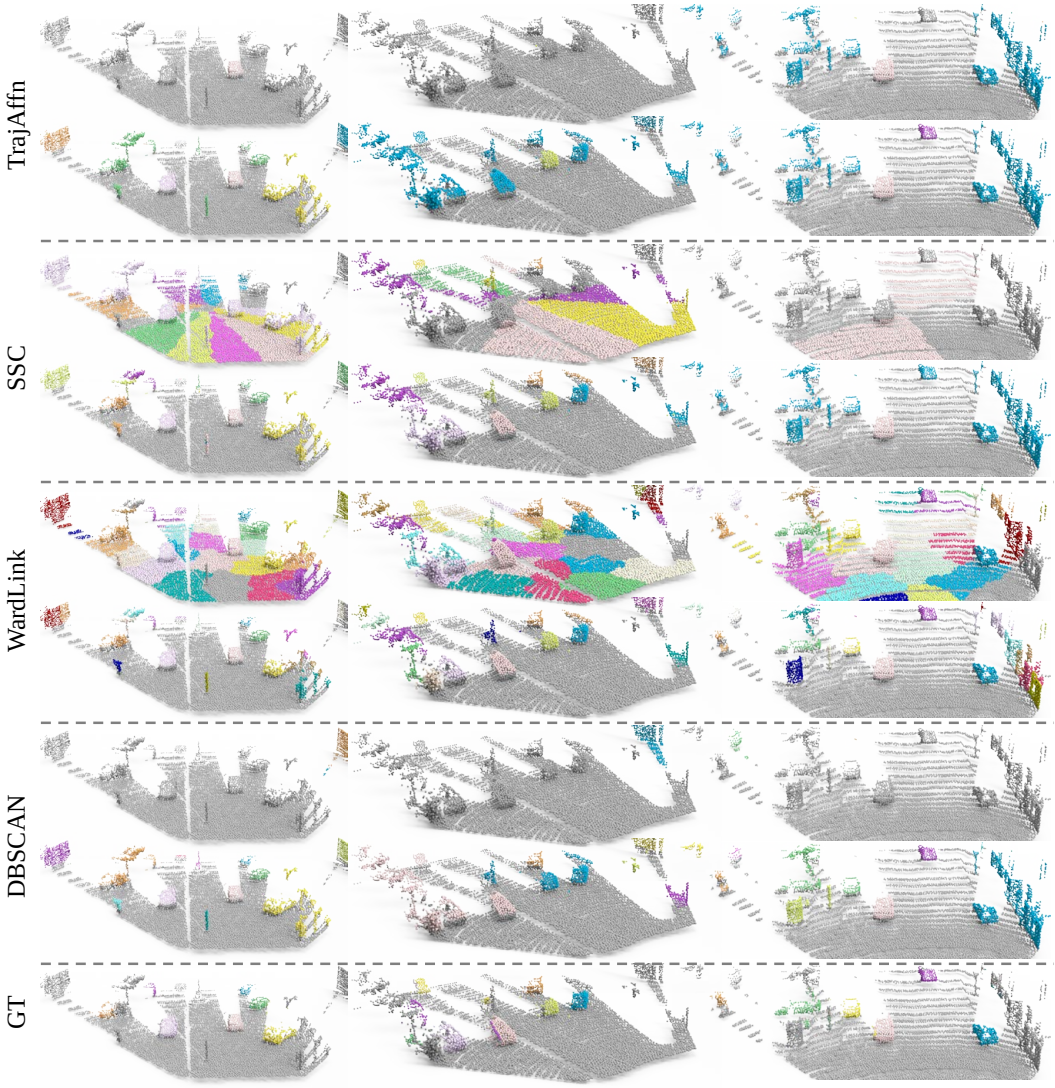

Figure 6: Qualitative results for ablation studies about the ground plane prior on KITTI-SF. For each baseline method, segmentation results without (top row) and with (bottom row) the ground plane prior are shown.

A.5 Additional Ablation Studies

(1) Geometry Consistency Losses

We conduct additional ablation experiments on the curated SAPIEN dataset for a more comprehensive analysis of losses in our framework. Recall that without the dynamic rigid loss $\ell_{dynamic}$, network predictions collapse and assign all points to a single object. The full SAPIEN dataset holds a number of point cloud frames with only 2 or 3 object parts, enabling the ablated model without $\ell_{dynamic}$ to still get plausible scores. However, as shown in Tables 14&15, once we evaluate the ablated models on point clouds with ≥ 3 object parts (Table 14) or ≥ 4 object parts (Table 15), the performance of the ablated model without $\ell_{dynamic}$ drops rapidly. This clearly shows that the $\ell_{dynamic}$ loss is truly critical to tackle complex scenes with more and more objects. Figure 7 shows qualitative examples.

We also conduct additional ablative experiments on the KITTI-SF dataset, as shown in Table 16. Similar to the results on the SAPIEN dataset, we observe the collapse of object segmentation without $\ell_{dynamic}$ and the oversegmentation issue without ℓ_{smooth}. Figure 7 gives an intuitive illustration.

Table 14: Additional ablation results on a curated SAPIEN dataset where only point clouds with ≥ 3 object parts are kept (844 frames in total).

Config.	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
w/o $\ell_{dynamic}$	15.8	21.0	32.7	**69.7**	21.4	18.4	44.1
w/o ℓ_{smooth}	13.0	15.5	23.7	18.9	31.8	42.9	65.5
w/o $\ell_{invariant}$	23.8	28.3	41.3	47.6	36.5	38.2	64.0
Full OGC	**30.8**	**34.0**	**48.2**	52.4	**44.6**	**43.4**	**67.4**

Table 15: Additional ablation results on a curated SAPIEN dataset where only point clouds with ≥ 4 object parts are kept (120 frames in total).

Config.	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
w/o $\ell_{dynamic}$	10.8	12.9	22.4	**65.0**	13.5	11.1	35.0
w/o ℓ_{smooth}	12.8	13.9	22.2	20.3	24.5	**35.3**	**67.3**
w/o $\ell_{invariant}$	15.7	21.6	31.9	46.2	24.3	26.8	60.3
Full OGC	**22.3**	**26.6**	**40.1**	55.0	**31.6**	29.7	59.8

Table 16: Additional ablation studies about loss designs on KITTI-SF.

Config.	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
w/o $\ell_{dynamic}$	24.8	37.1	39.6	**100.0**	24.7	31.3	88.3
w/o ℓ_{smooth}	44.9	31.8	39.6	30.9	55.1	61.2	90.2
w/o $\ell_{invariant}$	47.1	35.0	43.0	35.5	54.4	60.7	92.5
Full OGC	**54.4**	**42.4**	**52.4**	47.3	**58.8**	**63.7**	**93.6**

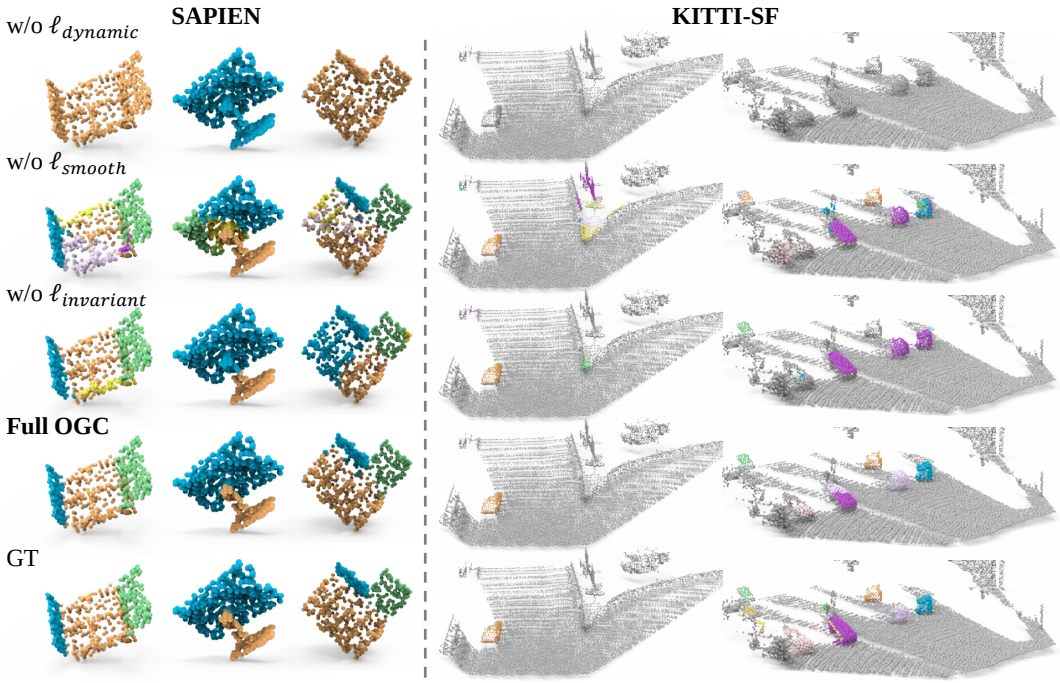

Figure 7: Qualitative results for ablation study on SAPIEN and KITTI-SF.

(2) Use of the Invariance Loss in Iterative Optimization

We conduct ablation experiments on the KITTI-SF dataset to validate our choice of using the invariance loss $l_{invariant}$ only in the final round of iterative optimization. To do this, we iteratively optimize on KITTI-SF for 2 rounds, with two different configurations: (i) We use $l_{invariant}$ in object segmentation optimization of the two rounds. (ii) We use $l_{invariant}$ only in the 2nd round. For each configuration, we run for five times with different random seeds and report the mean results with uncertainty levels.

Table 17: Ablation results about the use of the invariance loss in iterative optimization. $\#R$ denotes the number of iterative optimization rounds.

$\#R$	Split	Config	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
1	Train	(i)	40.7±2.1	27.7±0.8	38.7±1.1	27.0±0.9	68.0±1.6	61.5±0.8	62.9±1.1
		(ii)	41.4±3.0	31.3±0.9	43.5±1.2	30.5±1.1	75.4±1.2	64.8±0.7	60.6±1.0
	Test	(i)	34.1±2.0	24.1±1.6	35.0±2.1	26.0±1.8	53.9±2.2	52.9±1.1	57.2±0.9
		(ii)	29.1±1.9	22.7±1.0	33.7±1.6	24.9±1.4	52.1±1.6	51.2±0.8	54.2±1.1
2	Train	(i)	57.0±8.1	41.3±7.3	52.4±7.2	41.9±9.2	71.7±2.1	69.2±3.6	83.1±11.1
		(ii)	67.0±2.3	50.5±2.3	61.5±2.4	53.1±3.6	73.1±1.4	73.7±0.9	95.5±0.3
	Test	(i)	41.9±8.2	33.0±6.4	43.0±6.1	35.8±7.6	54.7±3.0	58.4±4.0	79.4±13.0
		(ii)	51.8±2.2	40.8±2.2	50.6±2.6	45.5±3.5	57.2±1.7	62.1±1.2	93.4±0.3

Analysis: As shown in Table 17, in the 1st round, the model (ii) trained without $l_{invariant}$ sacrifices some generalization performance on testing data (F1: 33.7 vs 35.0) while fitting better on training data (F1: 43.5 vs 38.7). As shown in Table 18, such advantages in segmentation performance on training data lead to more refined scene flows (EPE3D: 2.36 vs 3.12), which will directly influence the optimization in the next round. In constrast, better segmentation on testing data cannot be passed to the next round. In the 2nd round, both models are trained with $l_{invariant}$. The model (ii) stably produces superior segmentation results on both training (F1: 61.5±2.4 vs 52.4±7.2) and testing data (F1: 50.6±2.6 vs 43.0±6.1), owing to higher quality scene flow refinement inherited from the previous round.

Table 18: Refined scene flow estimation after the 1st round on KITTI-SF dataset.

Config	EPE3D↓	AccS↑	AccR↑	Outlier↓
(i)	3.12±0.16	92.8±0.7	94.7±0.5	23.9±0.5
(ii)	2.36±0.12	94.7±0.5	96.4±0.2	22.5±0.1

The findings above are consistent with our theoretical expectations. In the iterative optimization, the previous rounds can only influence the final results by passing refined scene flows (on training split) to the following rounds. The invariance loss $l_{invariant}$ brings better generalization, especially to static objects, while these properties are of little use for scene flow refinement on training data. Therefore, in the previous rounds of iterative optimization, we can exclude $l_{invariant}$ and let the model focus on moving objects in training samples and produce more refined scene flows. In the last round, $l_{invariant}$ can be included to boost the generalization ability of the final model.

(3) Robustness to Scene Flow Distortions

We investigate the robustness of our method to scene flow distortions on the OGC-DR and KITTI-SF datasets. To do this, we conduct experiments on three types of distorted scene flows: (i) We add different degrees of zero-mean Gaussian noise into the ground truth scene flows, and these noisy scene flows are used to supervised our object segmentation network. (ii) We use insufficiently trained scene flow estimators to produce low-quality scene flow estimations for supervision. (iii) We use the initial scene flow estimations which has not been refined by our iterative optimization. On our OGC-DR dataset, we evaluate all these ablations. On the KITTI-SF dataset, we only evaluate (i) and (iii), as we use a FlowStep3D model publicly released by the authors to estimate scene flows for KITTI-SF. The intermediate training models are not available to evaluate (ii).

Analysis on OGC-DR: As shown in Table 19, our OGC is robust to Gaussian noises in scene flows. The model maintains 85.2 AP even when the AccR of scene flows degrades to 6.9 only. In contrast, the flow distortions from insufficiently trained estimators incur a notable drop in the segmentation performance. The AP drops to 84.7 even when the scene flow AccR is still 32.2. From this, we hypothesize that our OGC can be robust to noisy flows with large variance thanks to the rigid loss integrated with weighted-Kabsch algorithm, but sensitive to large biases in estimated scene flows.

Analysis on KITTI-SF: As shown in Table 20, our method has strong robustness to Gaussian noise, same as on OGC-DR dataset. The model achieves 59.5 AP even when the scene flow is corrupted by

Table 19: Ablation results about the robustness to scene flow distortions on OGC-DR. Bold text denotes **the configuration of full OGC.** #R denotes the number of iterative optimization rounds. We report the object segmentation performance on the testing set and scene flow quality on training set (the scene flow quality of the testing set is irrelevant to our object segmentation).

			Object Segmentation							Scene Flow	
	Flow Source	#R	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑	EPE3D↓	AccR↑
Ablation (i)	GT + Gaussian (std=1.0)	1	91.3	83.3	88.1	84.4	92.1	89.2	97.6	1.60	73.9
	GT + Gaussian (std=2.0)	1	86.4	79.7	86.2	85.0	87.4	83.8	96.5	3.19	19.9
	GT + Gaussian (std=3.0)	1	85.2	77.2	84.5	82.9	86.2	82.0	95.8	4.79	6.9
Ablation (ii)	FlowStep3D (epoch=20)	1	90.1	81.0	86.4	81.6	91.9	88.2	96.8	1.14	90.2
	FlowStep3D (epoch=10)	1	89.1	79.8	86.1	82.1	90.6	86.2	96.3	1.45	86.0
	FlowStep3D (epoch=1)	1	84.7	71.6	80.5	74.3	87.8	80.9	93.5	3.84	32.2
Ablation (iii)	FlowStep3D (epoch=50)	1	91.3	83.7	88.4	84.5	92.7	89.7	97.6	0.98	90.0
	FlowStep3D (epoch=50)	**2**	92.3	85.1	89.4	85.6	93.6	90.8	97.8	0.76	92.2

Table 20: Ablation results about the robustness to scene flow distortions on KITTI-SF. Bold text denotes **the configuration of full OGC.**

			Object Segmentation							Scene Flow	
	Flow Source	#R	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑	EPE3D↓	AccR↑
Ablation (i)	GT + Gaussian (std=10.0)	1	61.1	49.5	59.5	54.9	65.0	68.8	94.5	15.96	35.8
	GT + Gaussian (std=20.0)	1	59.5	48.5	58.5	54.4	63.4	67.3	94.3	31.92	7.5
Ablation (iii)	FlowStep3D (epoch=120)	1	36.0	24.6	35.4	26.4	53.8	53.7	57.8	12.21	72.8
	FlowStep3D (epoch=120)	**2**	54.4	42.4	52.4	47.3	58.8	63.7	93.6	2.29	96.3

Gaussian noise to only 7.5 AccR. In contrast, the model without iterative optimization only gives 36.0 AP when the scene flow AccR is 72.8. Figure 8 shows qualitative results. In the middle column, although scene flows have an overall high quality, the inconsistency in scene flows between two parts of the same object leads to over-segmentation. We believe the Weighted Kabsch algorithm inside our dynamic rigid loss is the key. This algorithm inherently smooths the Gaussian-like noise in scene flows but cannot handle the biased errors. Fortunately, our object-aware ICP is designed to correct such inconsistency in the flows, thus improving segmentation performance in iterative optimization.

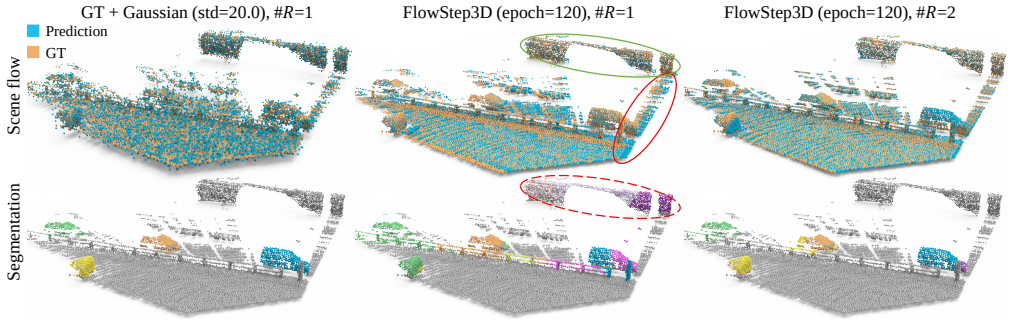

Figure 8: Qualitative results for ablation study about the robustness to scene flow distortions on KITTI-SF. Scene flows are visualized via the point cloud warped by the flows. In the middle column, the scene flow estimations are accurate for the above-ground background points (solid green ellipse) but have biased errors for the ground plane points (which can be clearly seen inside the solid red ellipse). This deviation of scene flow estimations between the above-ground background and the ground plane leads to over-segmentation of the background (dashed red ellipse).

(4) Choice of Hyperparameters for Smoothness Regularization

We evaluate the influence of smoothness regularization hyperparameters on OGC-DR, as shown in Table 21. When we strengthen the regularization by enforcing smoothness in a larger local neighborhood (*i.e.*, ablation H_3), the Precision score improves with less over-segmentation, while the Recall score is sacrificed. Figure 9 shows qualitative results. In general, as expected, such hyperparameters control the trade-off between over- and under-segmentation issues.

(5) Weighted Smoothness Regularization via Motion Similarity

We investigate a variant of the smoothness regularization which is weighted by the inter-point motion similarity. This motion-similarity-weighted smoothness regularization is mathematically defined as,

$$\ell'_{smooth} = \frac{1}{N}\sum_{n=1}^{N}\Big(\frac{1}{H}\sum_{h=1}^{H} d(\boldsymbol{o}_n, \boldsymbol{o}_{n_h}) \cdot \frac{exp(-\|\boldsymbol{m}_n - \boldsymbol{m}_{n_h}\|_2/\tau)}{E}\Big) \tag{4}$$

Table 21: Different choices of smoothness regularization hyperparamters on the OGC-DR dataset. The hyperparameter k controls the K nearest neighbors selected within a ball with radius r.

	$(k_1, r_1), (k_2, r_2)$	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
H_1	(4, 0.01), (8, 0.02)	92.2	83.4	88.1	83.1	93.7	90.5	97.6
H_2 **(Full OGC)**	(8, 0.02), (16, 0.04)	92.3	85.1	89.4	85.6	93.6	90.8	97.8
H_3	(32, 0.08), (64, 0.16)	84.6	81.0	87.4	89.5	85.4	82.3	96.8

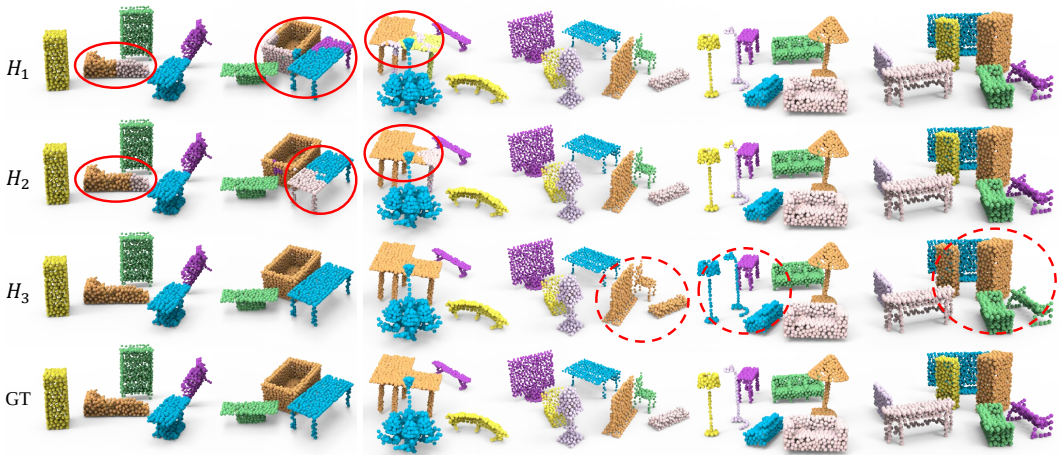

Figure 9: Qualitative results for the influence of smoothness regularization hyperparamters on OGC-DR. H_3 reduces over-segmentation issues in H_1/H_2 (solid red ellipses in column $1 \sim 3$), but fails to separate different objects sometimes (dashed red ellipses in column $4 \sim 6$).

where $\boldsymbol{m}_n \in \mathbb{R}^3$ represents the motion vector of center point $\boldsymbol{p}_n$, and $\boldsymbol{m}_{n_h} \in \mathbb{R}^3$ represents the motion vector of its hth neighbouring point. $\tau = 0.01$ is a temperature factor. E is a normalization term, *i.e.*, $E = \sum_{h=1}^{H} exp(-\|\boldsymbol{m}_n - \boldsymbol{m}_{n_h}\|_2/\tau)$. This variant selectively enforces the object mask smoothness among points with close locations and similar motions. Intuitively, it may avoid blurry predictions on object boundaries.

Table 22: Quantitative results of motion-similarity-weighted smoothness regularization on KITI-SF. Bold text denotes **the configuration of full OGC.**

Flow Source	#R	Regularizer	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
FlowStep3D (epoch=120)	**2**	l_{smooth}	54.4	42.4	52.4	47.3	58.8	63.7	93.6
FlowStep3D (epoch=120)	2	l'_{smooth}	49.8	40.7	50.1	46.0	55.0	61.1	93.5
GT + Gaussian (std=10.0)	1	l_{smooth}	61.1	49.5	59.5	54.9	65.0	68.8	94.5
GT + Gaussian (std=10.0)	1	l'_{smooth}	60.0	48.1	58.5	54.0	63.9	66.9	94.5
GT + Gaussian (std=20.0)	1	l_{smooth}	59.5	48.5	58.5	54.4	63.4	67.3	94.3
GT + Gaussian (std=20.0)	1	l'_{smooth}	57.9	46.5	56.3	51.1	62.6	67.2	94.4

Analysis: As shown in Table 22, l'_{smooth} brings no benefits to our OGC method under various scene flow situations. We believe the weighting via motion similarity makes l'_{smooth} more sensitive to noises in scene flow estimations, thus being inferior to our l_{smooth}.

A.6 Limitations of OGC

Our method can neither segment non-rigid objects nor discover unseen object types due to the lack of supervision signals. Besides, the trained OGC model may not provide generalizable intermediate representations to boost the performance of supervised model, which is discussed in details below.

We investigate whether our unsupervised method OGC can be used as a pre-training technique before fully-supervised fine-tuning with a small amount of labeled data, like other popular self-supervised representation learning methods. To do this, we firstly keep a subset (10%) of labelled point clouds

Table 23: Quantitative results of OGC as a pre-training step on KITTI-Det.

training strategy	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	RI↑
train OGC$_{sup}$ on 10% labelled KITTI-Det	71.5	58.3	68.4	61.8	76.7	79.1	96.0
(train OGC on unlabelled KITTI-SF + finetune on 10% labelled KITTI-Det)	66.4	53.6	63.3	56.0	72.7	76.2	95.1
train OGC on unlabeled KITTI-SF	41.0	30.9	37.7	31.4	47.0	59.9	85.0

from KITTI-Det training set, and then use our OGC model unsupervisedly trained on KITTI-SF to be fine-tuned on the labelled subset. For comparison, we additionally train a new model with full supervision on the same labelled subset from scratch.

Analysis: From Table 23, we can see that: 1) Not surprisingly, using the pre-trained OGC model followed by fine-tuning brings a significant improvement over our unsupervised OGC model, *i.e.*, the 1st row vs the 3rd row. 2) However, the (pre-train + fine-tune) strategy fails to outperform the fully-supervised model from scratch, *i.e.*, the 1st row vs the 2nd row. Fundamentally, this is because our unsupervised OGC is not dedicated to learn general intermediate representations for multiple downstream tasks. Instead, our OGC is task-driven and it aims to directly segment objects from raw point clouds. The learned latent representations in unsupervised training are likely to be different from the latent representations learned in fully-supervised training. In this regard, a naïve combination of (pre-train + fine-tune) may confuse the network and give inferior results. Nevertheless, how to effectively leverage the unsupervised model along with full supervision is an interesting direction and we leave it for future exploration.

A.7 Additional Qualitative Results

We provide additional qualitative results in Figure 11 for experiments in Sections 4.1 and 4.2 on SAPIEN and OGC-DR datasets, and in Figure 12 for experiments in Section 4.3 and 4.4 on the KITTI datasets.

For better visualization, we additionally project the segmented point clouds onto the corresponding RGB images in KITTI-SF and KITTI-Det datasets. As shown in Figure 10, our method OGC can successfully segment static cars parking alongside the road, thanks to our geometry invariance loss $l_{invariant}$ which enables our network to generalize the segmentation strategy to similar yet static objects through a set of scene transformations.

Figure 10: Qualitative results for static object segmentation on KITTI. Images in the 1st row are from KITTI-SF dataset, the rest are from KITTI-Det dataset. Static cars in yellow ellipses are parking alongside the road and can be successfully segmented by our method.

Figure 11: Additional qualitative results on SAPIEN, OGC-DR, and OGC-DRSV.

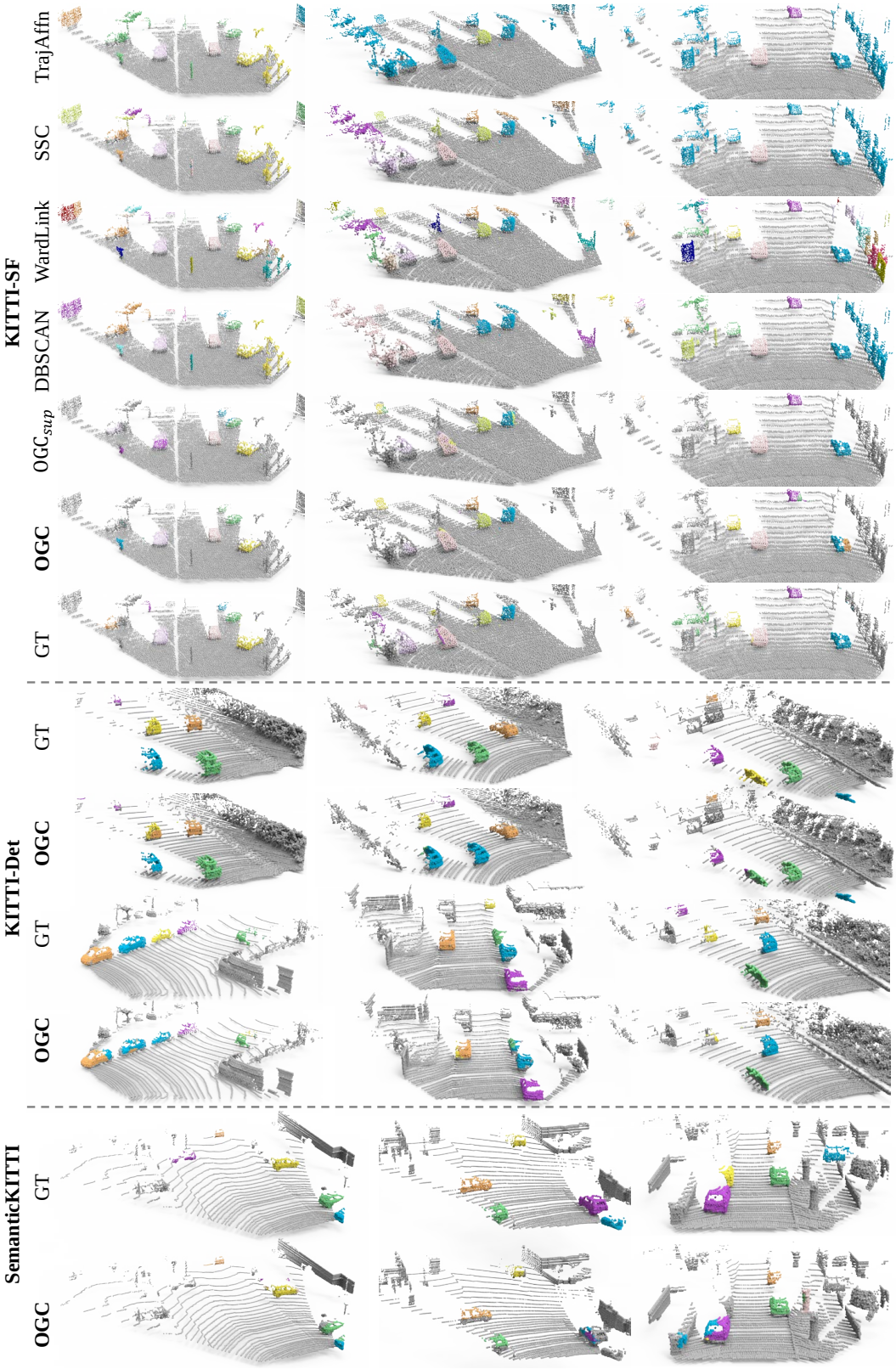

Figure 12: Additional qualitative results on KITTI-SF, KITTI-Det and SemanticKITTI.