# RMS-FlowNet++: Efficient and Robust Multi-Scale Scene Flow Estimation for Large-Scale Point Clouds

Ramy Battrawy[1*], René Schuster[1] and Didier Stricker[1]

[1]Augmented Vision, German Research Center for Artificial Intelligence (DFKI), Trippstadter Str. 122, Kaiserslautern, 67663, Rhineland-Palatinate, Germany.

*Corresponding author(s). E-mail(s): ramy.battrawy@dfki.de;
Contributing authors: rene.schuster@dfki.de; didier.stricker@dfki.de;

**Abstract**

The proposed RMS-FlowNet++ is a novel end-to-end learning-based architecture for accurate and efficient scene flow estimation that can operate on high-density point clouds. For hierarchical scene flow estimation, existing methods rely on expensive Farthest-Point-Sampling (FPS) to sample the scenes, must find large correspondence sets across the consecutive frames and/or must search for correspondences at a full input resolution. While this can improve the accuracy, it reduces the overall efficiency of these methods and limits their ability to handle large numbers of points due to memory requirements. In contrast to these methods, our architecture is based on an efficient design for hierarchical prediction of multi-scale scene flow. To this end, we develop a special flow embedding block that has two advantages over the current methods: First, a smaller correspondence set is used, and second, the use of Random-Sampling (RS) is possible. In addition, our architecture does not need to search for correspondences at a full input resolution. Exhibiting high accuracy, our RMS-FlowNet++ provides a faster prediction than state-of-the-art methods, avoids high memory requirements and enables efficient scene flow on dense point clouds of more than 250K points at once. Our comprehensive experiments verify the accuracy of RMS-FlowNet++ on the established FlyingThings3D data set with different point cloud densities and validate our design choices. Furthermore, we demonstrate that our model has a competitive ability to generalize to the real-world scenes of the KITTI data set without fine-tuning.

**Keywords:** Point Cloud, Scene Flow, Random-Sampling, Farthest-Point-Sampling, Flow Embedding, Patch-to-Dilated-Patch

## 1 Introduction

Robust perception of the dynamic environment is a fundamental task for many real-world applications such as autonomous driving, robot navigation, augmented reality, and human-robot interaction systems. The goal of scene flow is to estimate 3D displacement vectors between two consecutive scenes, representing all observed points in the scene as a dense or semi-dense 3D motion field. Therefore, scene flow can serve as an upstream step in high-level challenging computer vision tasks such as object tracking, odometry, action recognition, etc. With prior knowledge of the camera's intrinsic parameters, the 3D scene flow can be projected onto the image plane to obtain its 2D counterpart in pixel coordinates, which is called optical flow.

Many approaches propose pixel-wise scene flow estimation using stereo image sequences by combining geometry reconstruction with optical flow estimation to obtain dense scene flow [1–8]. Despite significant advances in such approaches, the overall accuracy of the resulting scene flow is highly dependent on the image quality, which can be poor under adverse lighting conditions. Compared to stereo systems, LiDAR sensors can accurately capture 3D geometry in the form of 3D point clouds and are less sensitive to lighting conditions. Therefore, there is an increasing emphasis on estimating scene flow directly from 3D point clouds.

Handling point clouds and finding correspondences in 3D space is more challenging due to the irregularity, sparsity of points, and varying point density of the scene. To tackle these challenges, several techniques develop deep neural architectures to estimate scene flow from point clouds. Some of these methods project the point cloud onto a permutohedral lattice [9] and then use bilateral convolutions [10]. Others organize 3D point clouds into voxels [11, 12] and use sparse convolutions [13] to facilitate scene flow prediction. However, these regular representations can introduce discretization artifacts and information loss that negatively affect the accuracy of the network.

With the advent of point-based networks on 3D point clouds [14, 15], many works estimate scene flow directly from *raw* point clouds using the multi-layer perceptron (MLP) as in [16–24] without the need for regular or intermediate representations. All of these techniques build a flow embedding module at coarse resolutions and then either use hierarchical refinement modules along with upsampling [16, 18, 20, 22–24] or use gated recurrent units (GRUs) [25] with iterative flow updating for the refinement process [17, 19, 21]. Despite their ability to capture both near and far matches, GRU-based methods [17, 19, 21] are less efficient in terms of runtime due to iterative flow updates along with expensive flow embedding layers.

Following hierarchical schemes, Wang *et al.* [22] propose a double attentive flow embedding along with the explicit learning of the residual scene flow. Their extension in [23] further improves the results by jointly learning of backward constraints in the all-to-all flow embedding

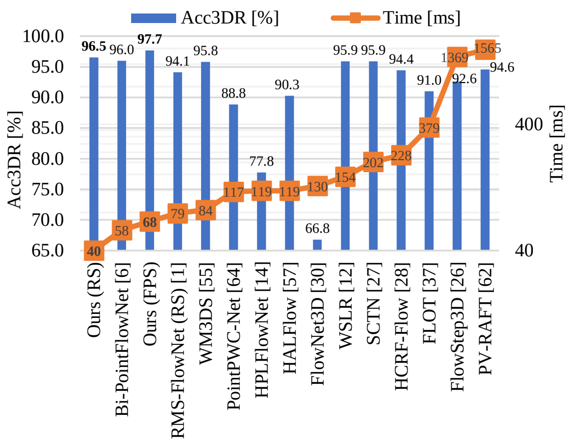


**Fig. 1**: Our RMS-FlowNet++ shows an accurate scene flow (Acc3DR) with a low runtime. The accuracy is tested on KITTI$_s$ [26] with 8192 points as input and the runtime is analyzed for all methods equally on a Geforce GTX 1080 Ti.

layer to capture distant matches. However, both methods [22, 23] reduce the scene flow resolution to a quarter of the input points, and they show that obtaining high accuracy of the scene flow at the full input resolution requires a further refinement module. This can be computationally expensive, while using a simple interpolation method can degrade the overall accuracy. In addition, the use of an all-to-all flow embedding layer in [23] increases the size of the correlation volume, which in turn increases the computational load of further operations. More recently, Bi-PointFlowNet [24] has proposed to learn bidirectional correlations from coarse-to-fine, searching for correspondences in both directions, and it uses a flow embedding layer at full input resolution.

All of the above point-based methods use Farthest-Point-Sampling (FPS) and rely on a K-Nearest-Neighbor (KNN) search with a large set of correspondences during the flow embedding, which both increases the computational and memory requirements and limits the ability to handle large point clouds.

To tackle these challenges, we present our RMS-FlowNet++ – a hierarchical point-based learning approach that requires smaller correspondence sets compared to the state-of-the-art methods and outperforms them when using FPS on the KITTI [26] data set. In addition, our model

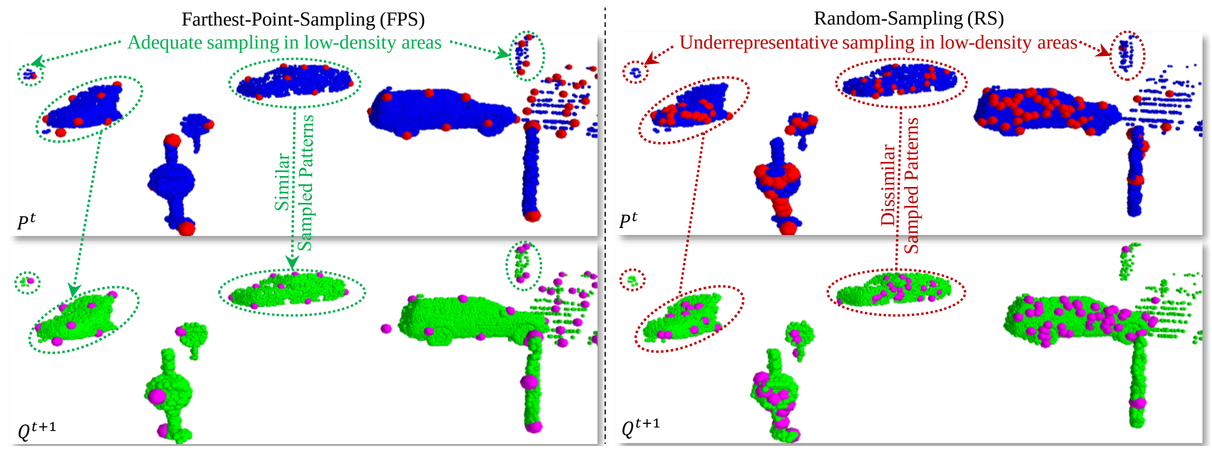

Farthest-Point-Sampling (FPS)
Adequate sampling in low-density areas
Similar Sampled Patterns
$P^t$
$Q^{t+1}$
Random-Sampling (RS)
Underrepresentative sampling in low-density areas
Dissimilar Sampled Patterns
$P^t$
$Q^{t+1}$

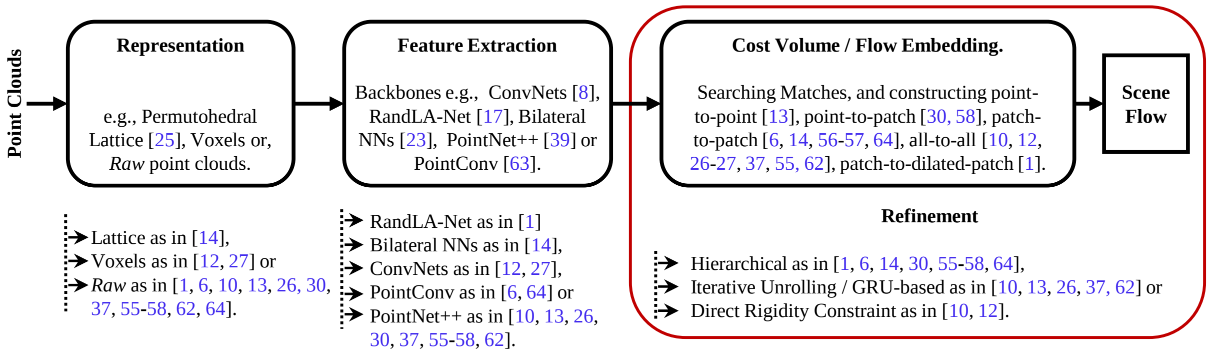
**Fig. 2**: The challenges of Random-Sampling (RS) (right) compared to Farthest-Point-Sampling (FPS) (left): Both techniques sample two consecutive scenes $P^t$ (blue) and $Q^{t+1}$ (green) into red and pink samples, respectively. Areas of low density are often not sufficiently covered by Random-Sampling (RS), resulting in dissimilar patterns. The patterns of the corresponding objects are much more similar when FPS is used, making it easier to match the points.

allows the use of RS and is therefore more efficient, has a smaller memory footprint, and shows comparable results at a lower runtime compared to the other state-of-the-art methods (*cf.* Fig. 1).

The advantage of RS combined with the smaller correspondence set results in our model being the only one that can robustly estimate scene flow on a very large set of points, as shown in Section 4.5. However, using RS for scene flow estimation is challenging for two main reasons: 1.) RS will reflect the spatial distribution of the input point cloud, which is problematic if it is far from uniform, which is a disadvantage compared to FPS. 2.) Corresponding (rigid) areas between consecutive point clouds will be sampled differently by RS, while FPS will yield more similar patterns. Both issues are illustrated in Fig. 2.

To overcome these problems, we propose a novel *Patch-to-Dilated-Patch* flow embedding consisting of four layers with lateral connections (*cf.* Fig. 5) to incorporate a larger receptive field during matching without increasing the physical set of correspondences. Overall, our fully supervised architecture consists of a hierarchical feature extraction, an optimized flow embedding, and scene flow prediction at multiple scales. The preliminary version of our network design has been published in RMS-FlowNet [27], but we are improving the overall design, which will lead to a very accurate result with higher efficiency. Our contribution can be summarized as follows:

- We propose RMS-FlowNet++ – an end-to-end scene flow estimation network that operates on dense point clouds with high accuracy.
- Our network consists of a hierarchical scene flow estimation with a novel flow embedding module (called *Patch-to-Dilated-Patch*) which is suitable for the combination with Random-Sampling.
- Compared to our previous work in RMS-FlowNet [27], we significantly reduce the size of the correspondence set, and omit some layers in the feature extraction module to increase the overall efficiency. Furthermore, we show that a feature-based search can increase the overall accuracy without sacrificing the efficiency.
- We explore the advantages of RS over FPS on high-density point clouds and its ability to generalize during the inference.
- We provide an intensive benchmark showing our strong results in terms of accuracy, generalization, and runtime compared to previous methods.
- Finally, we investigate the robustness of our network to occlusions and evaluate it for points acquired at longer distances ($> 35$ m).



Point Clouds
Representation
e.g., Permutohedral Lattice [25], Voxels or, *Raw* point clouds.
Lattice as in [14],
Voxels as in [12, 27] or
*Raw* as in [1, 6, 10, 13, 26, 30, 37, 55-58, 62, 64].
Feature Extraction
Backbones e.g., ConvNets [8], RandLA-Net [17], Bilateral NNs [23], PointNet++ [39] or PointConv [63].
RandLA-Net as in [1]
Bilateral NNs as in [14],
ConvNets as in [12, 27],
PointConv as in [6, 64] or
PointNet++ as in [10, 13, 26, 30, 37, 55-58, 62].
Cost Volume / Flow Embedding.
Searching Matches, and constructing point-to-point [13], point-to-patch [30, 58], patch-to-patch [6, 14, 56-57, 64], all-to-all [10, 12, 26-27, 37, 55, 62], patch-to-dilated-patch [1].
Scene Flow
Refinement
Hierarchical as in [1, 6, 14, 30, 55-58, 64],
Iterative Unrolling / GRU-based as in [10, 13, 26, 37, 62] or
Direct Rigidity Constraint as in [10, 12].

**Fig. 3**: We describe the generic pipeline of recent scene flow estimation methods. Like our previous work [27], our RMS-FlowNet++ estimates scene flow directly from *raw* point clouds and extracts features based on RandLA-Net [28]. Compared to recent scene flow methods, our novel *Patch-to-Dilated-Patch* allows the use of RS along with hierarchical or coarse-to-fine refinement.

## 2 Related Work

3D scene flow was first introduced by [29], developed using image-based (*e.g.*, RGB-D) setups [30–35], and then further developed in advanced deep learning networks [36, 37]. Since RGB-D sensors can only perceive depth at short distances, there have been many works that estimate scene flow from stereo images by jointly estimating disparity and optical flow [4, 5, 38, 39]. However, two-view geometry has inherent limitations in self-driving cars, such as inaccuracies in disparity estimation in distant regions. It can also suffer from poor lighting conditions, such as in dark tunnels. Our work focuses on learning scene flow directly from point clouds, without relying on RGB images.

**Scene Flow from Point Clouds:** With the recent advent of LiDAR sensors, which provide highly accurate 3D geometry of the environment for autonomous driving and robot navigation, it becomes increasingly important to estimate scene flow directly from point clouds in 3D world space. In this context, there is some work [40, 41] that formulates the task of scene flow estimation from point clouds as an energy optimization problem without taking advantage of deep learning. Advances in deep learning on 3D point clouds [14, 15, 42, 43] make neural networks more attractive and accurate for 3D scene flow estimation than the traditional methods [9, 11, 16–21, 24, 44–48]. These recent methods mostly follow a general scene flow estimation pipeline as shown in Fig. 3, but differ in how they represent point clouds, extract features, design the cost volume, or apply the refinement strategy. For example, with the breakthrough architecture of PointNet++ [15], many works estimate scene flow directly from *raw* point clouds in an end-to-end fashion [16, 17, 19–23, 47, 49, 50]. Based on PointNet++, FlowNet3D [16] is the first work to introduce a novel flow embedding layer. However, its accuracy is limited because there is only a single flow embedding layer and the correlation in local regions relies on the nearest spatial neighbor search, which may fail for long-range motion (*i.e.*, distant matches). In an attempt to overcome the limitations of FlowNet3D, many approaches introduce hierarchical scene flow estimation, iterative unrolling methods, or work under rigidity assumptions.

*Hierarchical Scene Flow:* HPLFlowNet [9] introduces multi-scale correlation layers by projecting the points into a permutohedral lattice as in SplatNet [43] and applying bilateral convolutional layers (BCL) [10, 51]. Despite of the efficiency of HPLFlowNet [9] on high-density point clouds, but the accuracy of the network is prone to unavoidable errors due to the splatting and slicing process. PointPWC-Net [18] avoids the grid representation in [9] and improves the scene flow accuracy on *raw* point clouds based on PointConv [14] by regressing multi-scale flows from coarse-to-fine. Following the hierarchical point-based designs, HALFlow [20] uses the point feature learning of PointNet++ [15], but proposes a hierarchical attention learning flow embedding with double attentions leading to better results than PointPWC-Net [18]. Further improvements are proposed in [22, 23] to develop the flow embedding of [20] through explicit prediction of residual flow [22] and backward reliability validation [23]. All previous methods take advantage of FPS for downsampling to provide accurate scene flow estimation, but at the cost of efficiency, especially for dense points. Compared to these methods, our network solves the challenge of using RS to work with high-density points.

*Iterative Unrolling for Scene Flow:* Besides hierarchical flow embedding schemes, a new trend started in FLOT [47], inspired by [52–54], to iteratively refine the scene flow by unrolling a fixed number of iterations to globally optimize an optimal transport map [55]. PV-RAFT [17], FlowStep3D [19], and RCP [21] extend unrolling techniques from optimization problems to learning-based models by using gated recurrent units (GRUs) [25] and capturing both local and global correlations. We find that iterative unrolling with a fixed number of iterations and repeated use of flow re-embedding works well at low input resolution, but is inefficient compared to hierarchical designs.

*Rigidity Assumption for Scene Flow:* Axiomatic concepts of explicit rigidity assumptions with ego-motion estimation are explored in [11, 50]. A plug-in refinement module is proposed by HCRF-Flow [48], which uses high-order conditional random fields (CRFs) to refine the scene flow by applying the rigidity condition at the region level. Our RMS-FlowNet++ is free of any rigidity constraint, so it can work with non-rigid bodies, such as pedestrians.

**Flow Embedding on Point Clouds:** The irregular data structure of point clouds makes it difficult to build cost volumes as with image-based solutions [54, 56, 57]. Therefore, previous works such as [9, 16, 18, 19] design complicated flow embedding layers to aggregate the matching costs from consecutive point clouds.

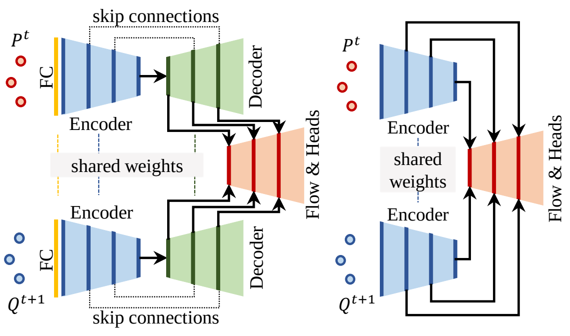
*Patch-to-Point Correlation:* FlowNet3D [16] introduces the flow embedding in a patch-to-point manner, which means that the set of neighboring correspondences in the target point cloud set are grouped into the source one based on the Euclidean space. Then, the correlations are learned using multi-layer perceptron (MLP) followed by max-pooling to aggregate the features of the correspondence set.

*Patch-to-Patch Correlation:* To incorporate a large field of correlations leading to better accuracy, HPLFlowNet [9] proposes a multi-scale patch-to-patch design that takes advantage of the regular representation using the permutohedral lattice [10, 51]. Apart from regular representations, PointPWC-Net [18] uses a patch-to-point flow embedding layer to aggregate the features of the correspondence set in the adjacent frames based on the point-wise continuous convolution in PointConv [14]. A point-to-patch embedding is then applied to aggregate the set of neighbor correspondences in the source itself. Instead of using the backbone of PointConv [14], HALFlow [20] uses a two-stage of attention mechanism to softly weight the neighboring correspondence features and allocate more attention to the regions with correct correspondences. With two-stage attentions, hierarchical and explicit learning of the residual scene flow is proposed by [22] to reduce the inconsistencies between the correlations and to handle fast-moving objects. Bi-PointFlowNet [24] uses the patch-to-patch mechanism in a bidirectional manner across all multi-scale layers, which requires intensive computation of forward-backward KNNs and additional refinement at full input resolution. Compared to [24], our network finds reliable correlations under bidirectional constraints using the cosine similarity matrix at the coarse scale, and then operates unidirectionally at the upper scales, requiring a small number of correlations defined by the KNN search. It also makes our solution more efficient without sacrificing accuracy by eliminating the need to refine at full input resolution.

*All-to-All Correlations:* There are several approaches that compute a global cosine similarity based on latent features and then learn soft correlations by iteratively refining an optimal transport problem using the non-parametric Sinkhorn algorithm as in FLOT [47]. WSLR [11] uses Sinkhorn, but refines the scene flow at the object-level based on the rigidity assumption and in combination with ego-motion estimation. Apart from object-level refinement, FlowStep3D [19] computes an initial global correlation matrix, and then uses gated recurrent units (GRUs) for local region refinement to iteratively align point clouds. Another GRU-based method is proposed by PV-RAFT [17], but its flow embedding design combines point-based and voxel-based features to preserve fine-grained information while encoding large correspondence sets at the same time. WM3D [23] designs all-to-all correlations, supported by reliability validation, but used only at low scale resolution, where each point in the source uses all points in the target for correlation, and thus each point in the target can therefore obtain the correlation with all points in the source. Then, two-stage attentive flow embedding as in [20, 22] is used to aggregate reliable correspondences among this large set of correspondences. All previous methods significantly increase the number of correlation candidates, which makes the refinement operations computationally intensive, especially when the input point clouds contain a large number of points. For this reason, some approaches, such as [20, 22, 23], limit the scene flow estimation to the quarter resolution of the input points to avoid further computation. Compared to [20, 22, 23], we design four stages in our flow embedding, ending with two stages of attention for final correlations refinements in a large receptive field. It also allows us to get the full resolution of the scene flow exactly the same as the resolution of the input points.

**RMS-FlowNet [27]:** Our preliminary network has proposed a trade-off between efficiency and accuracy by replacing the FPS sampling technique with the computationally cheap RS technique. In RMS-FlowNet, we have proposed a novel flow embedding design, called as *Patch-to-Dilated-Patch*, with three embedding steps to solve the challenges of using RS. In addition, we significantly optimize the correspondence search based on KNN by using the Nanoflann framework [58][1], which further increases the efficiency. Compared to RMS-FlowNet [27], we change the architectural design in our RMS-FlowNet++ to speed up the feature extraction module by eliminating the upsampling part (*i.e.*, decoder) (*cf.* Fig. 4) and the dense layers at the full input resolution in the encoder part. And in terms of accuracy, we add another search step in the flow embedding based on the feature space to improve the overall accuracy, and we also improve the way of pairwise correspondence search at the coarse scale under a bidirectional constraint. With these improvements, our

[1] https://github.com/jlblancoc/nanoflann



(a) RMS-FlowNet [27]. (b) RMS-FlowNet++.

**Fig. 4**: Our network design consists of feature extraction, flow embedding, warping layers, and scene flow heads, similar to our previous work RMS-FlowNet [27]. Compared to the feature extraction module in RMS-FlowNet, which consists of fully connected layers (FC) at full input resolution, encoder and decoder modules ($a$), we omit (FC) and the decoder in our RMS-FlowNet++ ($b$).

RMS-FlowNet++ becomes much more accurate while still showing high efficiency.

## 3 Network Design

Our RMS-FlowNet++ estimates scene flow as translational vectors from consecutive frames of point clouds (*e.g.*, from LiDAR or RGB-D sensors), with no assumptions about object rigidity or direct estimation of sensor motion within the environment (*i.e.*, no direct estimation of ego-motion).

Given Cartesian 3D point cloud frames $P^t = \{p_i^t \in \mathbb{R}^3\}_{i=1}^N$ and $Q^{t+1} = \{q_j^{t+1} \in \mathbb{R}^3\}_{j=1}^M$ at timestamps $t$ and $t+1$, respectively, our goal is to estimate point-wise 3D flow vectors $S^t = \{s_i^t \in \mathbb{R}^3\}_{i=1}^N$ for each point within the reference frame $P^t$ (*i.e.*, $s_i^t$ is the motion vector for $p_i^t$). The sizes $(N, M)$ of the two frames do not have to be identical, and the two frames should not have exact correspondences between their points. Our network is designed to estimate scene flow at multi-scale levels through hierarchical feature extraction using a novel design of flow embedding, called *Patch-to-Dilated-Patch*, with warping layers and scene flow estimation heads.

The components of each module are described in detail in the following sections.

## 3.1 Feature Extraction Module

The feature extraction module consists of two pyramid networks with shared parameters for the hierarchical extraction of two feature sets from $P^t$ and $Q^{t+1}$. Unlike our previous work in RMS-FlowNet [27], the design of this module includes only the encoder parts, while no decoder and no transposed convolutions are required to upsample the extracted features to the full resolution, as shown in Fig. 4.

The encoder part computes a hierarchy of features at four scales $\{l_k\}_{k=0}^{3}$ from fine-to-coarse resolution, where $l_0$ is the full resolution of ($P^t$ and $Q^{t+1}$) and the resolutions of the downsampled scales are fixed to $\{\{l_k\}_{k=1}^{3} \mid l_1 = 2048, l_2 = 512, l_3 = 128\}$ during training, but are kept adaptive at higher point densities (*cf.* Section 4.5). Each scale is essentially composed of two layers, where Local-Feature-Aggregation (LFA) is applied to aggregate the features at the $l_k$ scale, followed by Downsampling (DS) to aggregate the features from the $l_k$ level to $l_{k+1}$, resulting in a decrease in resolution. Inspired by RandLA-Net [28], which focuses only on semantic segmentation, we use the feature aggregation layer of LFA, which consists of three neural units: 1) local spatial encoding to encode the geometric and relative position features, 2) attentive pooling to aggregate the set of neighbor features, and 3) a dilated residual block.

To apply LFA, we search for the number of nearest neighbors ($K_p$) at all scales using KNNs search in Euclidean space and aggregate the features with two attentive pooling layers designed as in [28], where the attentive pooling unit is based on the mechanism of self-attention [59, 60]. DS samples the points to the defined resolution in layer $l_{k+1}$ and aggregates the nearest neighbors ($K_p$) from the higher resolution $l_k$ by using max-pooling. During training and evaluation with RS, $K_p$ is set to 20 in all layers and changed to 16 during evaluation with FPS.

The feature extraction module outputs two feature sets over all scales $\{F_k^t \in \mathbb{R}^{c_k}\}_{k=0}^{3}$ and $\{F_k^{t+1} \in \mathbb{R}^{c_k}\}_{k=0}^{3}$ for $\{P_k^t \in \mathbb{R}^3\}_{k=0}^{3}$ and $\{Q_k^{t+1} \in \mathbb{R}^3\}_{k=0}^{3}$, respectively. Here, $c_k$ is the feature dimension, which is fixed as $\{\{c\}_{k=0}^{3} \mid c_0 = 32, c_1 = 128, c_2 = 256, c_3 = 512\}$. The feature extraction module of our RMS-FlowNet++ is shown in Fig. 4b compared to our preliminary design in Fig. 4a. The design of LFA and DS allows the use of RS but still requires well-designed flow embedding to ensure robust scene flow.

## 3.2 Flow Embedding

A flow embedding module across consecutive frames is the key component for correlating the adjacent frames of point clouds, where finding reliable correlations is extremely important for encoding 3D motion. In this context, previous state-of-the-art methods combine: 1) grouping of correspondences from $Q^{t+1}$, 2) robust aggregation of correspondence features into $P^t$, 3) refinement of flow embedding.

All point-wise learning-based methods take an advantage of FPS (as explained in Fig. 2) to sample the consecutive frames and rely on finding large correspondence sets ($\geqslant 32$ matches) in the sampled $Q^{t+1}$ based on KNN as in [18, 20, 24] or much more in all-to-all correlations as in [23]. While FPS can generate similar patterns across consecutive scenes to facilitate obtaining strong match pairs, finding large correspondence sets can increase the likelihood of correlating distant matches (*i.e.*, for large displacements) [22, 23]. In addition, some work aims to add refinement at the fine resolution of the input points [18, 24]. Taken together, this can increase accuracy, but it reduces the overall efficiency of these methods and limits their ability to handle high point densities. In addition, considering large correspondence sets can greatly increase the possibility of aggregating unreliable correlations, leading to inaccurate estimates.

To address these issues, we develop a special flow embedding module that has two advantages over current point-based methods: First, a smaller correspondence set is used without the need for flow embedding at full resolution, and second, the use of RS is possible. As a result, we speed up our model and make it amenable to RS, as shown by our results in Section 4, which allows higher point densities with low memory requirements (*cf.* Fig. 7). We must recall that, the use of RS is more challenging than FPS (*cf.* Fig. 2) for two reasons. First, regions with low local point density are underrepresented when using RS. Second, the sampling patterns for corresponding regions are less correlated across frames.

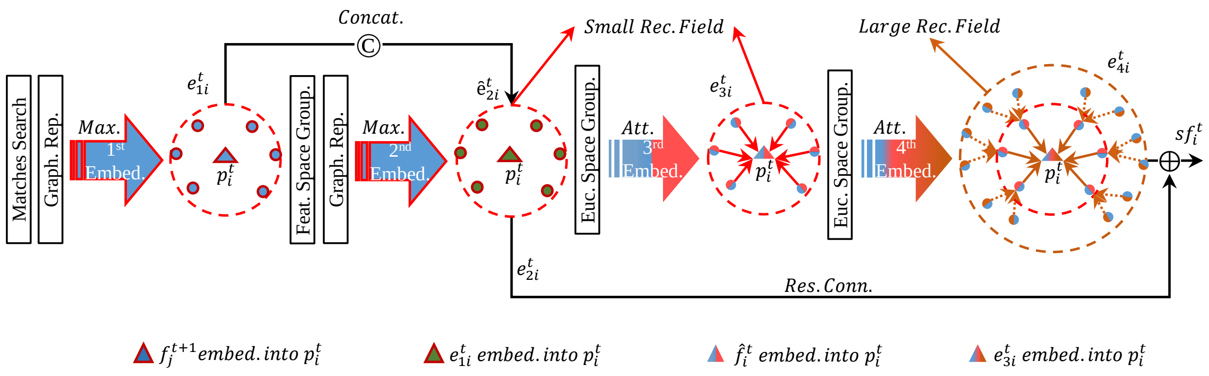


**Fig. 5**: Our novel Flow-Embedding (FE) module consists of four main steps and yields the scene flow feature $sf^t_i$: Two maximum embedding layers based on both Euclidean and feature space followed by two attentive embedding layers. Lateral connections are also used: A Concatenation (*Concat.*) between the first two embeddings and a residual connection (*Res. Conn.*).

Our novel and efficient flow embedding, called *Patch-to-Dilated-Patch*, aggregates large correspondence sets without increasing the physical number of the nearest neighbors. This is basically the same design as in our previous work RMS-FlowNet [27], but we apply some changes that are outlined in Table 4. In this context, we search for correspondences not only in Euclidean space, but also in feature space, and we add another embedding step.

**Matches Search**: Grouping strong correspondences is the first step in any flow embedding. Many state-of-the-art methods search for the set of matches based on Euclidean space, but apply soft weights in different ways. Since the grouping of correlations based on Euclidean space may not be sufficient to capture distant matches, we use the feature space to find reliable matches at the coarse scale (last down-sampled layer) $l_3$.

*Point-to-Point Bidirectional Map:* For the above reasoning, we compute a simple cosine similarity matrix based on the feature space to find a pair of matches under bidirectional constraint that applies a point-to-point (*i.e.*, one-to-one) correlation map. Based on the above reasoning, $q^{t+1}_j$ in $Q^{t+1}$ is a true match to $p^t_i$ in $P^t$ if the highest similarity score is guaranteed in a bidirectional way, otherwise the search for matches is done in Euclidean space. Finding robust matches at the coarse scale leads to a high quality initial estimate of the scene flow at scale $l_3$. This also approximates the distant matches at the upper scales using the warping layer, so that $p^t_i$ is close to its match in $Q^{t+1}$.

*Patch-to-Point Search:* For the upper scales $\{l\}^2_{k=1}$, it is not worth computing the cosine similarity, since it is difficult to get distinctive features in a one-to-one manner at high point densities, and it is worth searching for the number of closest matches $p^t_i$ within $Q^{t+1}$ based on the Euclidean space, denoted by $\mathcal{N}_Q(p^t_i)$.

**Graph Representation**: After finding the likelihood correspondences, we construct the correlations in a graph form $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the vertices and edges, respectively. Then, we apply multi-layer perceptron (MLP):

$$h^{\mathcal{E}}_{\Theta}(v_i) = MLP(\{[v_i, v_j - v_i] \mid (i,j) \in \mathcal{E}\}) \qquad (1)$$

where $([.,.]$ denotes the concatenation, $v_i$ is a central vertex feature, and $v_j - v_i$ denotes the edge features. This representation is compared in Table 4 with the original form in our preliminary work RMS-FlowNet [27], which omits the $v_i$ part in Eq. (1) and keeps only the edge features.

**Flow Embedding Steps**: Having found the number of possible matches (*i.e.*, $\mathcal{N}_Q(p^t_i)$) within the representation in the form described above, we apply the flow embedding aggregation steps at each scale $l_k$ for every ith element within the down-sampled scales $\{l\}^3_{k=1}$, except the full-resolution scale $l_0$, as follows:

- 1st Embedding (*Patch-to-Point*): We first apply max-pooling to the output of Eq. (1) and obtain

$e^t_{1i}$ as shown in the following equation:

$$e^t_{1i} = \underset{f^{t+1}_j \in \mathcal{N}_Q(p^t_i)}{MAX} (h^{\mathcal{N}_Q(p^t_i)}_\Theta (f^t_i)) \tag{2}$$

- 2nd Embedding (*Patch-to-Point*): Compared to RMS-FlowNet, we add another embedding step to group correspondences that are semantically similar by applying KNN in feature space, inspired by the backbone of DGCNN [61]. For this purpose, we group the number of nearest neighbors $\mathcal{N}_E(e^t_{1i})$ for each output element of Eq. (2) based on the feature space and apply the graph form, where $\mathcal{N}_E$ denotes the neighboring features of $e^t_{1i}$. Next, we apply max-pooling to obtain $\hat{e}^t_{2i}$, which is then channel-wise concatenated with $e^t_{1i}$, followed by a multi-layer perceptron (MLP) to obtain $e^t_{2i}$:

$$\hat{e}^t_{2i} = \underset{e^t_{1j} \in \mathcal{N}_E(e^t_{1i})}{MAX} (h^{\mathcal{N}_E(e^t_{1i})}_\Theta (e^t_{1i})), \tag{3}$$

$$e^t_{2i} = MLP([e^t_{1i}, \hat{e}^t_{2i}]) \tag{4}$$

- 3rd Embedding (*Point-to-Patch*): Using channel-wise concatenation, we combine the feature $f^t_i$ of $p^t_i$ with the output of Eq. (4) on the coarse scale $l_3$, and further combine the upsampled scene flow feature $sf^t_i$ (computed by Eq. (10)) and the upsampled scene flow $s^t_i$ (computed by the scene flow head in Section 3.3) on the upper scales as follows:

$$\hat{f}^t_i = [f^t_i, e^t_{2i}, sf^t_i, s^t_i] \tag{5}$$

Then, we group the nearest features $\hat{f}^t_i$ based on the Euclidean search $\mathcal{N}_P(p^t_i)$ ($p^t_i$ is the 3D spatial location of $\hat{f}^t_i$), then compute the attention weights $w^t_{1i}$ and sum the weighted features to obtain $e^t_{3i}$:

$$w^t_{1i} = g(\hat{f}^t_i, \boldsymbol{W}), \tag{6}$$

$$e^t_{3i} = \sum_{n=1}^{K_p} (\hat{f}^t_n \cdot w^t_{1n}) \tag{7}$$

where $g()$ consists of a shared MLP with trainable weights $\boldsymbol{W}$ followed by *softmax*. With this attention mechanism, high attention is paid to the well-correlated features, while the less correlated features are suppressed. The attention-based mechanism is generally inspired by [59, 60].

- 4th Embedding (*Point-to-Dilated-Patch*): It repeats the previous step on the output result $e^t_{3i}$ with new attention weights $w^t_{2i}$ for the nearest features based on Euclidean space. This embedding layer results in an increased receptive field embedding $e^t_{4i}$:

$$w^t_{2i} = g(e^t_{3i}, \boldsymbol{W}), \tag{8}$$

$$e^t_{4i} = \sum_{n=1}^{K_p} (e^t_{3n} \cdot w^t_{2n}) \tag{9}$$

where $g()$ consists of a shared MLP with trainable weights $\boldsymbol{W}$ followed by *softmax*. Technically, we aggregate features from a larger range by repeating the aggregation mechanism without physically increasing of the number of the nearest neighbors, inspired by [28].

Finally, to improve the quality of our flow embedding, we add a residual connection (Res. Conn.), which is an element-wise summation of $e^t_{2i}$ and $e^t_{4i}$, resulting in the scene flow feature $sf^t_i$ (*cf.* Fig. 5):

$$sf^t_i = e^t_{2i} + e^t_{4i} \tag{10}$$

Note that for all of the above flow embedding steps, we need to group a certain number of features ($K_p$) and aggregate their features either by max-pooling or by attention. $K_p$ is set to 20 in all layers with RS and changed to 16 during the evaluation with FPS. We found that training with RS generalizes well with FPS without any fine-tuning (*cf.* Table 7). In addition, we must emphasize that the third and forth embedding steps do not require a new KNN search because we reuse the predefined neighbors of the feature extraction module. Together, these four steps lead to our novel *Patch-to-Dilated-Patch* embedding, which is described in Fig. 5. In this way, we are able to obtain a larger receptive field with a small number of nearest neighbors, which is computationally more efficient.

Experiments with the 1st, 3rd, and 4th flow embedding steps were performed in our previous work [27], and we explore our feature-based search in the additional 2nd embedding step in Table 4.

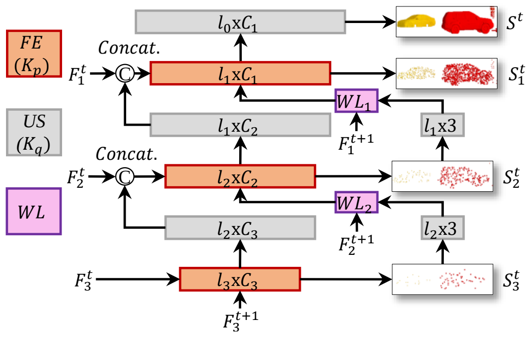


**Fig. 6**: Multi-scale scene flow prediction with three Flow-Embedding (FE) modules (each consists of four steps), two Warping-Layers (WLs), four scene flow estimators and Upsampling (US) layers.

### 3.3 Multi-Scale Scene Flow Estimation

Our RMS-FlowNet++ predicts scene flow at multiple scales, inspired by PointPWC-Net [18], but we consider significant changes in conjunction with RS to make our prediction more efficient. Our scene flow prediction over all scales consists of two Warping-Layers (WLs), three Flow-Embeddings (FEs), four scene flow estimators, and Upsampling (US) modules, as shown in Fig. 6. Compared to the design of PointPWC-Net [18] and Bi-PointFlowNet [24], we save one element from each category and do not use FE at full input resolution. As a result, we speed up our model without sacrificing accuracy, as shown in Fig. 1. The multi-scale estimation starts at the coarse resolution by predicting $S_3^t$ with a scene flow estimation module after an initial FE. The scene flow estimation head takes the resulting scene flow features in Eq. (10) and applies three layers of MLPs with 64, 32, and 3 output channels, respectively. Then, the estimated scene flow and the upcoming features from FE are upsampled to the next higher scale using one nearest neighbor based on KNN search (*i.e.*, $K_q = 1$). We use the same strategy to upsample the scene flow from the $l_1$ scale to the full input resolution $l_0$ without any additional FE.

Our Warping-Layer uses the upsampled scene flow $S_k^t$ at scale level $l_k$ to warp $P_k^t$ toward $Q_k^{t+1}$ to obtain $\widetilde{P}_k^{t+1}$. This forward warping process does not require any further KNN search because the predicted scene flow is associated with $P_k^t$. This is more efficient compared to PointPWC-Net [18] or Bi-PointFlowNet [24], which must first associate the scene flow with $Q_k^{t+1}$ using KNN search in order to warp $Q_k^{t+1}$ to $P_k^t$ in the backward direction.

### 3.4 Loss Function

The model is fully supervised at multiple scales, similar to PointPWC-Net [18]. If $S_k^t$ is the predicted scene flow and the ground truth is $S_{GT,k}^t$ at scale $l_k$, then the loss can be written as follows:

$$\mathcal{L}(\theta) = \sum_{k=0}^{3} \alpha_k \sum_{i=1}^{l_k} \| s_{ki}^t - s_{GT,ki}^t \|_2, \tag{11}$$

where $\|.\|_2$ denotes the $L_2$-norm and the weights per level are $\{\{\alpha_k\}_{k=0}^3 \mid \alpha_0 = 0.02, \alpha_1 = 0.04, \alpha_2 = 0.08, \alpha_3 = 0.16\}$.

## 4 Experiments

We conduct several experiments to evaluate the results of our RMS-FlowNet++ for scene flow estimation. First, we demonstrate its accuracy and efficiency compared to the state-of-the-art methods. Second, we verify our design choice with several analyses.

### 4.1 Evaluation Metrics

For a fair comparison, we use the same evaluation metrics as in [9, 16–24, 47]. Let $S^t$ denotes the predicted scene flow, and $S_{GT}^t$ denotes the ground truth scene flow. The evaluation metrics are averaged over all points and computed as follows:

- *EPE3D [m]*: The 3D end-point error computed in meters as $\|S^t - S_{GT}^t\|_2$.
- *Acc3DS [%]*: The strict 3D accuracy which is the ratio of points whose EPE3D $< 0.05$ $m$ **or** relative error $< 5\%$.
- *Acc3DR [%]*: The relaxed 3D accuracy which is the ratio of points whose EPE3D $< 0.1$ $m$ **or** relative error $< 10\%$.

If a metric is subscripted with "$_{\text{noc}}$", only the non-occluded points are evaluated, otherwise all input points are considered.

## 4.2 Data Sets and Preprocessing

As with state-of-the-art methods, we use the original version $FT3D_o$ and the subset version $FT3D_s$ of the established large-scale synthetic data set FlyingThings3D (FT3D) [62]. The subset version $FT3D_s$ differs from the original $FT3D_o$ by excluding some frames from the original and adding more labels. This data set provides the ground truth for scene flow represented as disparity changes over consecutive frames, disparity maps for consecutive frames and optical flow components, so that the 3D translation vector of the ground truth for scene flow can be computed. In addition, the $FT3D_s$ subset provides occlusion maps on the basis of disparity, future and past motions.

In contrast to FT3D, the KITTI data set [26] is a small data set with optical flow labels that consists of real outdoor scenes for autonomous driving applications and provides sparse disparity maps generated by a LiDAR sensor. The given second disparity map at timestamp $t+1$ has been aligned with the first frame at timestamp $t$, allowing the computation of 3D translation vectors for scene flow.

**Point Clouds Generation:** Since the existing scenes of data sets and labels do not provide a direct representation of point clouds (*i.e.*, 3D Cartesian locations), the state-of-the-art methods [9, 16–24, 47] basically generate 3D point cloud scenes for their models using the given calibration parameters in the data sets. The generated point clouds are randomly subsampled to be evaluated at a certain resolution (*e.g.*, 8192 points) and shuffled to dissolve correlations between consecutive point clouds. For this, we use the preprocessing strategies of the pioneering work in FlowNet3D [16] and HPLFlowNet [9], the latter of which yields non-occluded and exact correlations between the scenes. We also use other preprocessing strategies to ensure that there are no exact correlations between consecutive point clouds. To do this, we use the given consecutive disparity maps in $FT3D_s$, and for the KITTI data set, we use the de-warped disparity maps of the second frame at timestamp $t+1$ generated by [63, 64].

All of the above preprocessing mechanisms differ in how the second point cloud $Q^{t+1}$ is generated, which either results in exact correlations or not, and whether occluded points are considered or not. This is summarized in Table 1 and described in more detail below.

**Preprocessing in HPLFlowNet [9][2]:** This preprocessing considers the complete set of FlyingThings3D subset ($FT3D_s$), which consists of 19640 labeled scenes available in the training set and all 3824 frames available in the test split for evaluation. Unlike the FlowNet3D preprocessing [16], this preprocessing removes all the occluded points using occlusion maps provided in $FT3D_s$ and the second point cloud frame $Q^{t+1}$ is generated from disparity change and optical flow labels. For the KITTI data set, the 142 labeled scenes of the training split available in the *raw* KITTI data are preprocessed. The second frame of the point cloud $Q^{t+1}$ is generated from the second disparity map by warping in 3D space, but without occlusion handling. The data generated from KITTI by this preprocessing is referred to as $KITTI_s$. The preprocessing in HPLFlowNet results in an exact correlation between the consecutive point clouds and occluded points are not taken into account.

**Preprocessing in FlowNet3D [16][3]:** Here, the original version of FlyingThings3D ($FT3D_o$) is used, with 20,000 images from the training set and 2,000 images from the test set randomly selected for training and evaluation, respectively. During preprocessing, many occluded points are included in the data and an occlusion mask is computed for $P^t$, since there are no predefined occlusion maps in $FT3D_o$. The frames of the point clouds $P^t$ and $Q^{t+1}$ are generated directly from the consecutive disparity maps and there are no exact correlations between the consecutive scenes. For the KITTI data set, this preprocessing considers 150 frames with occlusions, but does not compute an occlusion mask. The second frame of the point cloud $Q^{t+1}$ is generated using the second disparity map by a warping process in 3D space with occlusion handling. The data generated from KITTI by this preprocessing is referred to as $KITTI_o$.

**Preprocessing with Occlusion Masks:** In contrast to the preprocessing in HPLFlowNet [9], we generate both point clouds ($P^t$ and $Q^{t+1}$) directly from the consecutive disparity maps of the $FT3D_s$, resulting in very low correlations and

[2]https://github.com/laoreja/HPLFlowNet.
[3]https://github.com/xingyul/flownet3d.

**Table 1**: The preprocessing mechanisms of the data sets differ in how the second point cloud $Q^{t+1}$ is generated and whether occluded points are considered or not.

| Data Set | Processing | Correlated | None Occ. | Partial Occ. | Large Occ. |
|---|---|---|---|---|---|
| **FT3D** [62] | **FT3D$_s$** [9] | ✓ | ✓ | ✗ | ✗ |
| | **FT3D$_o$** [16] | ✗ | ✗ | ✗ | ✓ |
| | **FT3D$_{so}$ (ours)** | ✗ | ✗ | ✓ | ✓ |
| **KITTI** [26] | **KITTI$_s$** [9] | ✓ | ✓ | ✗ | ✗ |
| | **KITTI$_o$** [16] | ✗ | ✗ | ✓ | ✗ |
| | **KITTI$_d$** [64] | ✗ | ✗ | ✓ | ✓ |

existing occlusions in the scenes. By using the occlusion maps for disparity change and optical flow of consecutive scenes in forward and backward direction provided by FT3D$_s$, we omit most of the occlusions in consecutive frames, leaving very few occluded points in all frames. These remaining occlusions are due to imperfections in the occlusion masks, and are referred to as *partial occlusions*. We also generate the same data without filtering any of the occlusions. This version is referred to as *large occlusions*. The preprocessed data from FT3D$_s$ with partial or large occlusions are referred to as FT3D$_{so}$.

To generate decorrelated points in KITTI, where the given disparity maps of $t+1$ are aligned to the reference view at timestamp $t$, we use the preprocessing mechanism proposed in [63, 64]. In this preprocessing, the ground truth of the optical flow is used to generate $Q^{t+1}$ through a pixel-by-pixel de-warping process for each disparity map of $t+1$ aligned with the reference view, which largely dissolves the correlations between the point cloud scenes. We consider the 142 labeled scenes of the training split available in the *raw* KITTI data for preprocessing. The de-warped disparity maps can be downloaded directly from the source code of DeepLiDARFlow [64][4], and are used to compute the point clouds for both frames (*i.e.*, $P^t$ and $Q^{t+1}$). Given the occlusion maps in KITTI, we can either omit or include occluded points to create *partial* or *large occlusions*. The data generated from KITTI by this preprocessing is referred to as KITTI$_d$.

[4]https://github.com/dfki-av/DeepLiDARFlow.

## 4.3 Implementation, Training and Augmentation

As in related work, we train our model twice; once with non-occluded data from FT3D$_s$, considering all frames in the train split of FT3D$_s$ [62], and a second time with FT3D$_o$, containing 20,000 frames with largely occluded points. During training, the preprocessed data is randomly subsampled to 8192 points, where the order of the points is random and the correlation between consecutive frames is dissolved by random selection. Following related work, we remove points with depths greater than 35 meters, which retains the majority of moving objects contained.

We use the Adam optimizer with default parameters and train the final version of our model with RS for 1260 epochs. The final model generalizes well with both RS and FPS sampling methods, and no further training with FPS is required (*cf.* Table 7). However, to speed up some experiments, we also train with FPS for 420 epochs, which converges faster than the training with RS. When we report the results of the model trained with FPS, we highlight (*) next to FPS to distinguish it from the final model trained with RS.

We apply an exponentially decaying learning rate that is initialized at 0.001 and then decreases at a decaying rate of 0.8 every 20 epochs when training with FPS and every 60 epochs when training with RS.

We add two types of augmentation: First, we add geometric augmentation, *i.e.*, points are randomly rotated by a small angle around the X, Y, and Z axes, and a random translational offset is added to increase the ability of our model to generalize to KITTI [26] without fine-tuning. Second, when training with non-occluded data from FT3D$_s$, each high-resolution frame is randomly sampled to 8192 points in each epoch differently. However, we do not consider this type of augmentation with the FT3D$_o$ because the processed data set using the established preprocessing strategy in FlowNet3D [16] contains only 8192 points. The augmentation increases the ability of our model to generalize to the KITTI data set without fine-tuning (*cf.* Table 8).

**Table 2**: When training with non-occluded data on $FT3D_s$, we evaluate 8192 points in non-occluded scenes from $FT3D_s$ and $KITTI_s$, and the results are written in white cells. The light and dark gray cells contain the results with partial and large occlusions from $FT3D_{so}$ and $KITTI_d$. In all cases, we evaluate all input points, *i.e.* 8192. The best result per data set and column is highlighted in bold and the second best is underlined. Each method marked with (†) is designed to estimate the scene flow at a quarter resolution of the input points (*i.e.*, 2048 points out of 8192).

| Data Set | Model | without occlusions | | | with partial occlusions | | | with large occlusions | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **EPE3D** [m] ↓ | **Acc3DS** [%] ↑ | **Acc3DR** [%] ↑ | **EPE3D** [m] ↓ | **Acc3DS** [%] ↑ | **Acc3DR** [%] ↑ | **EPE3D** [m] ↓ | **Acc3DS** [%] ↑ | **Acc3DR** [%] ↑ |
| **$FT3D_s/FT3D_{so}$** [62] | FlowNet3D [16] | 0.114 | 41.25 | 77.06 | 0.197 | 21.46 | 43.17 | 0.293 | 8.330 | 31.35 |
| | HPLFlowNet [9] | 0.080 | 61.60 | 85.57 | 0.131 | 44.22 | 72.33 | 0.344 | 19.02 | 43.27 |
| | PointPWC-Net [18] | 0.059 | 73.79 | 92.76 | 0.113 | 65.30 | 83.58 | 0.292 | 35.72 | 55.94 |
| | FLOT [47] | 0.052 | 73.20 | 92.70 | 0.130 | 37.40 | 69.91 | 0.227 | 30.26 | 59.21 |
| | WSLR [11] | 0.052 | 74.60 | 93.60 | 0.140 | 55.60 | 79.10 | 0.461 | 28.00 | 49.70 |
| | HALFlow† [20] | 0.049 | 78.50 | 94.68 | 0.125 | 46.11 | 74.96 | 0.313 | 22.67 | 47.41 |
| | HCRF-Flow [48] | 0.049 | 83.37 | 95.07 | - | - | - | - | - | - |
| | PV-RAFT [17] | 0.046 | 81.69 | 95.74 | 0.131 | 61.19 | 81.27 | 0.441 | 29.85 | 50.53 |
| | FlowStep3D [19] | 0.046 | 81.62 | 96.14 | 0.111 | 62.67 | 82.75 | 0.361 | 30.73 | 51.62 |
| | RCP [21] | 0.040 | 85.67 | 96.35 | - | - | - | - | - | - |
| | SCTN [12] | 0.038 | 84.70 | 96.80 | - | - | - | - | - | - |
| | ResidualFlow† [22] | 0.031 | 91.39 | 97.68 | - | - | - | - | - | - |
| | WM3D† [23] | **0.028** | **92.90** | **98.17** | **0.078** | **77.66** | **89.58** | **0.267** | 46.49 | **65.81** |
| | Bi-PointFlowNet [24] | **0.028** | 91.80 | 97.80 | 0.085 | 75.55 | 87.68 | 0.284 | 46.04 | 62.04 |
| | RMS-FlowNet (RS) [27] | 0.051 | 80.00 | 95.60 | 0.104 | 64.20 | 84.20 | 0.305 | 35.50 | 57.20 |
| | RMS-FlowNet (FPS) [27] | 0.052 | 79.10 | 95.60 | 0.111 | 61.30 | 83.00 | 0.336 | 33.10 | 55.20 |
| | **RMS-FlowNet++ (RS)** | 0.033 | 91.00 | 97.51 | 0.087 | 76.24 | 88.41 | 0.301 | 45.12 | 63.43 |
| | **RMS-FlowNet++ (FPS)** | 0.029 | 92.41 | 98.10 | 0.081 | 77.54 | 89.17 | 0.285 | **47.04** | 64.94 |
| **$KITTI_s/KITTI_d$** [26] | FlowNet3D [16] | 0.177 | 37.38 | 66.77 | 0.206 | 15.16 | 44.05 | 0.330 | 13.17 | 36.32 |
| | HPLFlowNet [9] | 0.117 | 47.83 | 77.76 | 0.161 | 40.62 | 68.37 | 0.338 | 31.93 | 52.95 |
| | PointPWC-Net [18] | 0.069 | 72.81 | 88.84 | 0.096 | 74.36 | 85.91 | 0.276 | 59.82 | 69.81 |
| | FLOT [47] | 0.055 | 75.93 | 91.00 | 0.135 | 57.54 | 76.08 | 0.318 | 46.71 | 63.69 |
| | WSLR [11] | 0.042 | 84.90 | 95.90 | 0.107 | 72.70 | 84.50 | 0.344 | 60.90 | 70.90 |
| | HALFlow [20] | 0.062 | 76.49 | 90.26 | 0.133 | 56.88 | 77.26 | 0.250 | 52.25 | 67.10 |
| | HCRF-Flow [48] | 0.053 | 86.31 | 94.44 | - | - | - | - | - | - |
| | PV-RAFT [17] | 0.051 | 83.24 | 94.55 | 0.095 | 73.63 | 87.60 | 0.213 | 59.67 | 73.73 |
| | FlowStep3D [19] | 0.056 | 79.94 | 92.58 | 0.103 | 69.84 | 84.48 | 0.296 | 58.11 | 70.31 |
| | RCP [21] | 0.048 | 84.91 | 94.48 | - | - | - | - | - | - |
| | SCTN [12] | 0.037 | 87.30 | 95.90 | - | - | - | - | - | - |
| | ResidualFlow† [22] | 0.035 | 89.32 | 96.20 | - | - | - | - | - | - |
| | WM3D† [23] | 0.031 | 90.47 | 95.80 | 0.048 | 83.92 | 92.34 | 0.215 | 65.09 | 73.44 |
| | Bi-PointFlowNet [24] | 0.030 | 92.00 | 96.00 | 0.059 | 86.08 | 91.45 | 0.233 | 68.68 | 74.52 |
| | RMS-FlowNet (RS) [27] | 0.052 | 83.30 | 94.10 | 0.094 | 69.50 | 85.40 | 0.259 | 58.40 | 71.50 |
| | RMS-FlowNet (FPS) [27] | 0.047 | 88.20 | 95.80 | 0.093 | 75.60 | 86.90 | 0.256 | 63.90 | 73.20 |
| | **RMS-FlowNet++ (RS)** | 0.035 | 90.23 | 96.53 | 0.062 | 83.35 | 92.03 | 0.236 | 66.95 | 75.13 |
| | **RMS-FlowNet++ (FPS)** | **0.027** | **93.98** | **97.67** | **0.046** | **88.94** | **94.50** | **0.214** | **71.42** | **77.72** |

## 4.4 Comparison to State-of-the-Art

To demonstrate the accuracy and generalization of our model, we compare it with state-of-the-art methods in Table 2. The white cells denote the evaluation on the non-occluded $FT3D_s$ as usually done in related work. The results within the light and dark gray cells denote the evaluation with partially and extensively occluded scenes of $FT3D_{so}$. Our RMS-FlowNet++ allows the use of RS and shows very comparable results to the use of FPS, but with lower runtime (*cf.* Fig. 1), especially for higher resolution points (*cf.* Fig. 7).

**Evaluation on $FT3D_s$:** We test our RMS-FlowNet++ on non-occluded data from $FT3D_s$, as shown in the white cells in Table 2. Processing the entire points with an all-to-all correlation (*i.e.*, global correlation) using an optimal transport solver in FLOT [47] shows significantly lower accuracy than the hierarchical mechanism of our RMS-FlowNet++ using RS. This confirms our decision to design our model in a hierarchical way. The RS version of our RMS-FlowNet++ significantly outperforms the regular representative methods as in [9, 11, 12]. This supports our

**Table 3**: Occluded points are taken into account during training and inference. $FT3D_o$ and $KITTI_o$ are generated using FlowNet3D [16] preprocessing.

| Data Set | Model | EPE3D [m] ↓ | $EPE3D_{noc}$ [m] ↓ | $Acc3DS_{noc}$ [%] ↑ | $Acc3DR_{noc}$ [%] ↑ |
|---|---|---|---|---|---|
| $FT3D_o$ [62] | FlowNet3D [16] | 0.212 | 0.158 | 22.86 | 58.21 |
| | HPLFlowNet [9] | 0.201 | 0.169 | 26.29 | 57.45 |
| | PointPWC-Net [18] | 0.195 | 0.155 | 41.60 | 69.90 |
| | FLOT [47] | 0.250 | 0.153 | 39.65 | 66.08 |
| | FESTA [49] | - | 0.125 | 39.52 | 71.24 |
| | OGSFNet [65] | 0.163 | 0.122 | 55.18 | 77.67 |
| | OGSelSFNet [66] | 0.138 | 0.103 | 63.76 | 82.40 |
| | WM3D [23] | - | **0.063** | **79.10** | **90.90** |
| | Bi-PointFlowNet [24] | **0.102** | 0.073 | **79.10** | 89.60 |
| | **RMS-FlowNet++ (RS)** | 0.126 | 0.074 | 74.18 | 88.53 |
| | **RMS-FlowNet++ (FPS)** | 0.113 | 0.066 | 77.16 | 90.04 |
| $KITTI_o$ [26] | FlowNet3D [16] | 0.175 | - | 9.850 | 41.98 |
| | HPLFlowNet [9] | 0.343 | - | 10.30 | 38.60 |
| | PointPWC-Net [18] | 0.118 | - | 40.30 | 75.70 |
| | FLOT [47] | 0.110 | - | 41.90 | 72.10 |
| | FESTA [49] | 0.094 | - | 44.58 | 83.35 |
| | OGSFNet [65] | 0.075 | - | 70.70 | 87.25 |
| | OGSelSFNet [66] | 0.060 | - | 77.55 | 90.69 |
| | WM3D [23] | 0.073 | - | 81.90 | 89.90 |
| | Bi-PointFlowNet [24] | 0.065 | - | 76.90 | 90.60 |
| | **RMS-FlowNet++ (RS)** | 0.067 | - | 80.63 | 90.58 |
| | **RMS-FlowNet++ (FPS)** | **0.051** | - | **89.00** | **94.78** |

decision to handle the raw points without intermediate representations for scene flow estimation. Furthermore, our RMS-FlowNet++ with RS outperforms GRU-based methods [17, 19, 21] with a lower runtime (*cf.* Fig. 1).

Compared to hierarchical designs that basically use FPS, our RMS-FlowNet++ with RS outperforms [16, 18, 20, 48] on all metrics and is highly competitive with very recent methods [22–24] at lower runtime as shown in Fig. 1. However, the FPS version of our RMS-FlowNet++ outperforms the methods of [22, 24] and shows very comparable results to [23] with slight differences.

Moreover, our improvements in RMS-FlowNet++ are significant in both RS and FPS sampling versions, even for a small number of correspondences (*i.e.*, $K_p$ is set to 20 with RS and 16 with FPS), compared to our preliminary work RMS-FlowNet [27], which uses a correspondence set of 33 points.

**Generalization to $KITTI_s$:** We test the generalization ability to the KITTI data set [26] without fine-tuning, as shown in Table 2, where the white cells denote the scores on $KITTI_s$. Our RMS-FlowNet++ shows a stronger generalization ability with both sampling techniques, RS and FPS, than all state-of-the-art methods. This is best indicated by the much smaller gap in scores between the synthetic $FT3D_s$ and the realistic $KITTI_s$ results. With both sampling techniques, our RMS-FlowNet++ outperforms all the methods of [9, 16–22, 47]. Compared to the competing methods in [23, 24], RS with our RMS-FlowNet++ shows comparable results, but with lower runtime (*cf.* Fig. 1), and FPS outperforms these methods for all metrics.

**Robustness to Occlusions:** Training with non-occluded points using $FT3D_s$ shows that our RMS-FlowNet++ is able to estimate a reasonable accuracy of scene flow on the test split data when evaluated on $FT3D_{so}$, as shown in the light and dark gray cells in Table 2. When evaluated on the $FT3D_{so}$, the scores of all input points are reported, taking into account the partial or large number of occluded points. In the evaluation of the $FT3D_{so}$ test split with occlusions, the FPS and RS versions of our RMS-FlowNet++ take second and third place, respectively, behind the method in [23].

On $KITTI_d$, *i.e.* including occluded points in the evaluation, the FPS version of our RMS-FlowNet++ shows the best results of all methods in all metrics, and the faster RS version of our method shows comparable results to competing methods [23, 24] when evaluated.

Table 3 shows the results on $FT3D_o$ and $KITTI_o$, where occluded points are additionally considered during training. For a fair comparison, we follow the other method's evaluation scheme and include the occlusions during inference and evaluate over all input points in the EPE3D metric. In the $EPE3D_{noc}$, $Acc3DS_{noc}$, and $Acc3DR_{noc}$ metrics, we ignore the occluded points when computing the scores, but still include them as input. For both data sets, $FT3D_o$ and $KITTI_o$, our RMS-FlowNet++ with RS ranks right behind the competing methods [23, 24], but with FPS we rank second for $FT3D_o$ and first for $KITTI_o$.

## 4.5 Varying Point Densities

We evaluate the two versions of our RMS-FlowNet++, *i.e.* with RS and FPS, against our earlier work [27] and the competing methods [23, 24] in terms of accuracy (Acc3DR) and runtime at different input densities. The results of this comparison are shown in Fig. 7. We consider a wide range of densities $N = \{4096 * 2^i\}_{i=0}^{7}$ of $FT3D_s$, and finally all available non-occluded points are evaluated, which corresponds to ~225K points on average. For a fair comparison, all methods are trained with a fixed resolution of 8192 points only, and we do not consider fine-tuning or further training with different point densities. We measure the inference time for all methods

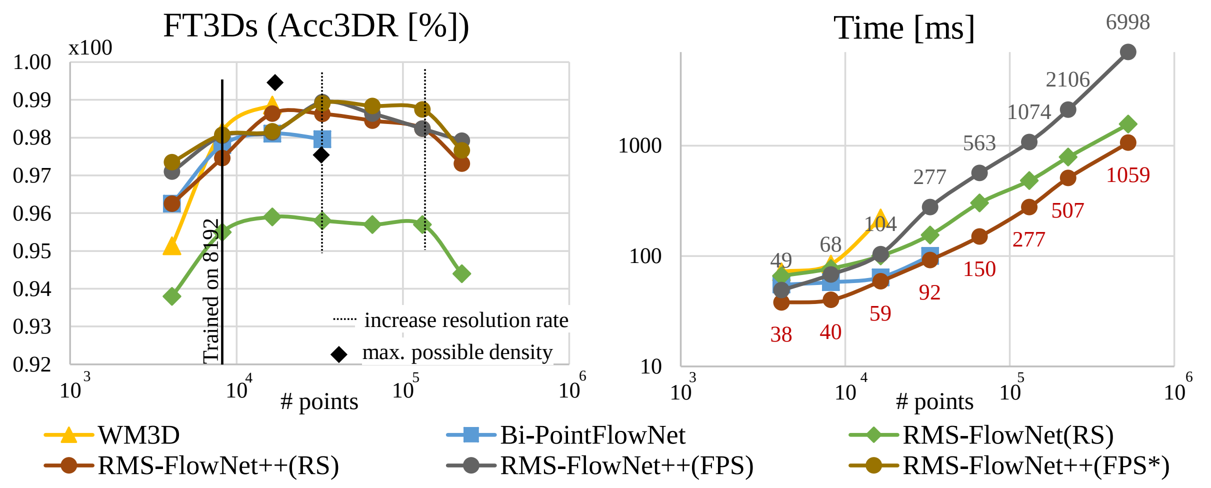


**Fig. 7**: Analysis of accuracy and runtime on FT3D$_s$ for different numbers of input points compared to state-of-the-art methods.

equally on a Geforce GTX 1080 Ti, including our RMS-FlowNet++ and RMS-FlowNet [27] with RS.

For both RS and FPS versions of our method, to keep the accuracy stable for densities > 32K, we increase the resolution rate of the downsampling layers (*cf.* Section 3.1) to $\{\{l\}_{k=1}^{3} \mid l_1 = 4096, l_2 = 1024, l_3 = 256\}$ without any further training or fine-tuning. Furthermore, for the RS of our RMS-FlowNet++ for densities > 131K, we increase the resolution rate of the downsampling layers to $\{l\}_{k=1}^{3} \mid l_1 = 8192, l_2 = 2048, l_3 = 512\}$ without further training or fine-tuning. Increasing the resolution of the downsampling layers is not possible with FPS, as this would exceed the memory limit of the Geforce GTX 1080 Ti. Nevertheless, the accuracy of our method remains stable over a wide range of densities for both RS and FPS versions (*cf.* Fig. 7). To maintain the accuracy of WM3D [23] and to evaluate more than 8192 input points, we had to increase the resolution rate of the downsampling layers based on the resolutions suggested in [23]. Yet, for the competing methods WM3D [23] and Bi-PointFlowNet [24], the maximum possible densities are limited to 16384 and 32768, respectively, since they exceed the memory limit of the Geforce GTX 1080 Ti at higher densities. For the other state-of-the-art-methods FLOT [47], PV-RAFT [17], PointPWC-Net [18], and HPLFlowNet [9], the maximum possible densities are limited to 8192, 8192, 32768 and 65536, respectively, for the same reason (not shown in Fig. 7).

In contrast, our RMS-FlowNet++ allows very high densities with high accuracy without exceeding the memory limit of the Geforce GTX 1080 Ti. Although FPS is computationally expensive, the reduced number of nearest neighbors ($K_p = 16$) allows the operation with ∼225K points. Using RS with the increased number of nearest neighbors ($K_p = 20$) allows our RMS-FlowNet++ to operate 5 to 6 times faster than with FPS, especially at densities $> 65K$. Consequently, the design of RMS-FlowNet++ allows a much higher maximum density compared to other methods in terms of memory requirement and time consumption. However, the runtime of our RMS-FlowNet++ still increases super-linearly with increasing input density > 225K due to the KNN search. In addition, the initial drop in accuracy at ∼225K points in Fig. 7 indicates that it may be necessary to further increase the resolution of the downsampling layers when using even higher densities.

We visually present some results on KITTI$_s$ with dense points (∼50K points) and three examples of non-occluded points of the FT3D$_s$ (∼300K points) in Fig. 9. To obtain a denser scene in the KITTI$_s$, we include distant points down to < 210 meters. Our RMS-FlowNet++ shows a high accuracy even with this very dense data.

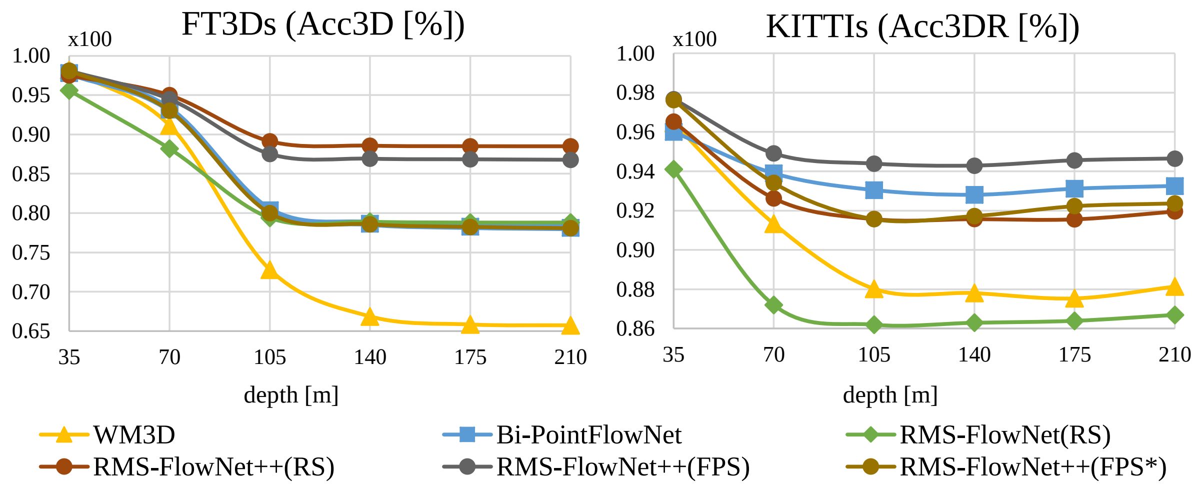


**Fig. 8**: Analysis of accuracy for different depth limits on $\text{FT3D}_\text{s}$ and $\text{KITTI}_\text{s}$ compared to state-of-the-art methods.

## 4.6 Varying Depth Ranges

We emphasize that all state-of-the-art methods only consider objects in the near range ($<$ 35 meters) during training and evaluation. We consider the same range during training, but in this work, for the first time, we evaluate the accuracy of scene flow for more distant objects using the $\text{FT3D}_\text{s}$ and $\text{KITTI}_\text{s}$ data sets (*cf.* Fig. 8).

On $\text{FT3D}_\text{s}$, the accuracy of our RMS-FlowNet++ with RS and FPS is better than the competing methods for every depth limit. The accuracy of WM3D [23] decreases significantly, and the accuracy of Bi-PointFlowNet [24] is ~8% lower than ours. Surprisingly, RS generalizes slightly better than FPS to increasing depth limits. When RMS-FlowNet++ is trained and evaluated with FPS (marked with *), the results are on par with our prior work RMS-FlowNet [27] and the competing method Bi-PointFlowNet [24]. When evaluated on $\text{KITTI}_\text{s}$, our RMS-FlowNet++ shows significantly better results than our prior work RMS-FlowNet [27]. Furthermore, both sampling strategies perform significantly better than WM3D [23]. However, when trained with RS and evaluated with FPS, the scene flow accuracy exceeds that of Bi-PointFlowNet [24].

It follows that training with RS can better generalize to a wider range of points than FPS, leading to better scene flow accuracy with FPS than training with FPS itself. In other words, FPS causes the downsampled points to cover approximately the same spatial locations, which reduces the variation during training.

Qualitatively, we visualize two scenes of $\text{KITTI}_\text{s}$ with the corresponding error maps in Fig. 10. We qualitatively compare our RMS-FlowNet++ with both sampling techniques to the most competitive method [24]. Without changing the training strategy, we compare the predicted scene flow with the narrowest depth range ($<$ 35m) and with the widest range ($<$ 210m) against Bi-PointFlowNet. Testing within the trained depth range ($<$ 35m), we see that Bi-PointFlowNet has higher errors on flat surfaces than our approach, which produces the best results with FPS. Testing outside the trained depth range ($>$ 35m) shows that the accuracy decreases when distant points ($<$ 210m) are included. In [24], even nearby objects (*e.g.* moving cars) are negatively affected when distant points ($>$ 35m) are included in the scene. However, our RMS-FlowNet++ performs robustly in this case.

## 4.7 Ablation Study

To speed up our experiments, we verify our design decisions, the additional components in the FE by training with FPS, which converges faster than RS. We also compare RS with FPS and increase the KNNs to test the effect on the results. Finally, we compare the impact of our augmentation on the overall results with RS.

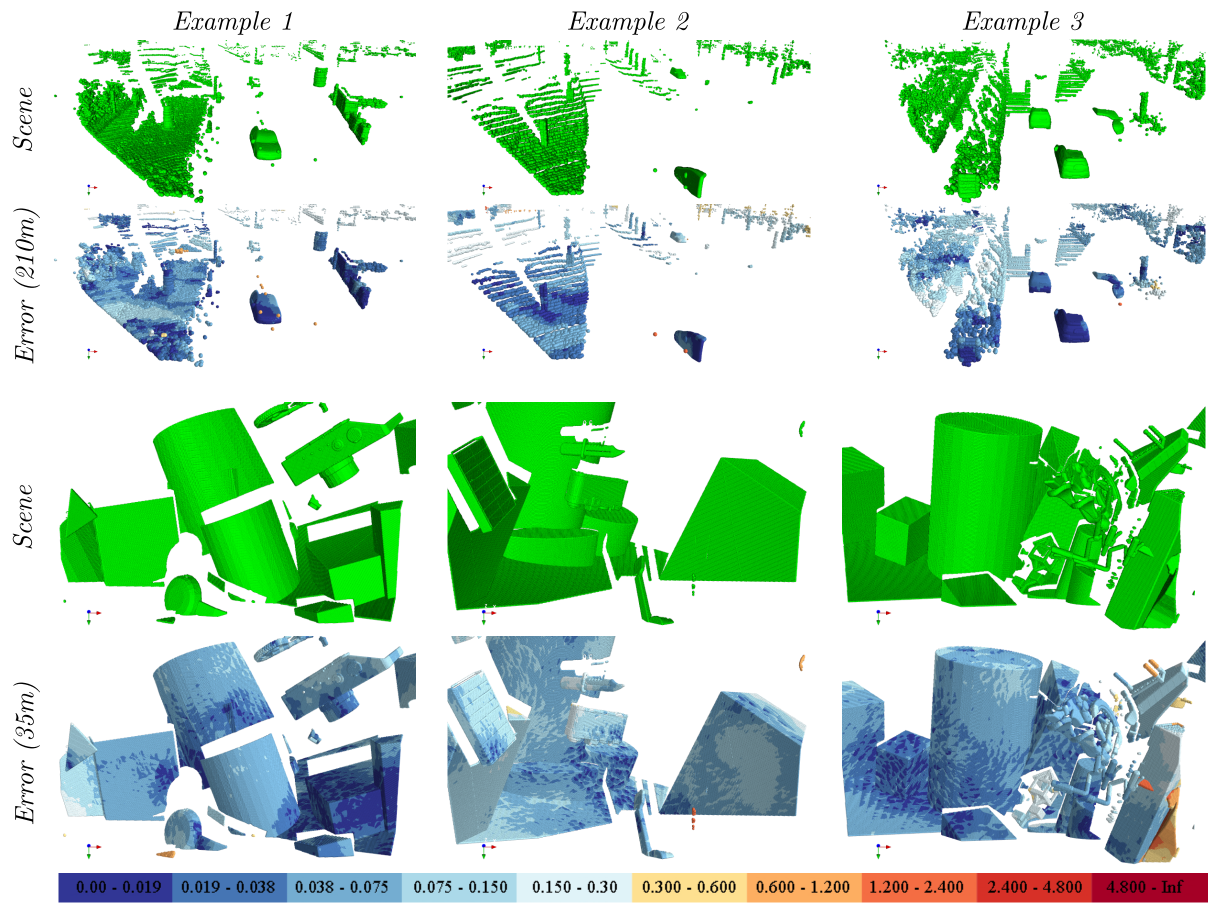


**Fig. 9**: Three examples from the non-occluded versions of FT3D$_s$ and KITTI$_s$ show that our RMS-FlowNet++ allows high point densities with high accuracy using RS. The scene of each example (first and third rows) visualizes $P^t$ as green color. The error map of each scene (second and forth rows) shows the end-point error in meters according to the color map shown in the last row.

**Design Decisions:** First, we reduce the number of correlation points ($K_p$) from 33 in our preliminary work of RMS-FlowNet [27] to 16 in the whole design (*i.e.*, all scales of LFA and FE), then we verify our improvements in RMS-FlowNet++ as shown in Table 4. The accuracy of our preliminary work RMS-FlowNet decreases by reducing the number of correlation points to 16, but it is improved again by using the graph representation of Eq. (1) in our FE, which adds the $f_i^t$ part to the original representation in RMS-FlowNet [27]. Second, we slightly improve the results by omitting the decoder part in the feature extraction which saves more operations and upsampling layers and avoids the use of KNN search. Third, we show the positive effect of adding the 2$^{nd}$ embedding step to the FE of RMS-FlowNet [27], which is based on the feature space. Then, we check the positive effect of our similarity map (*i.e.*, one-to-one map) based on the feature space to find a pair of matching points under a bidirectional constraint as explained in Section 3.2. Finally, we show the final results using the FPS of the model, but trained using the RS sampling technique with correspondence set ($K_p$) during training. Compared to RMS-FlowNet [27], the reduction of $K_p$ makes the method more efficient, while the sum of architectural changes improves the results.

**Aspects of Attention in FE:** We design our Flow-Embedding (FE) with two maximum embedding layers based on both Euclidean and feature space followed by two attentive embedding

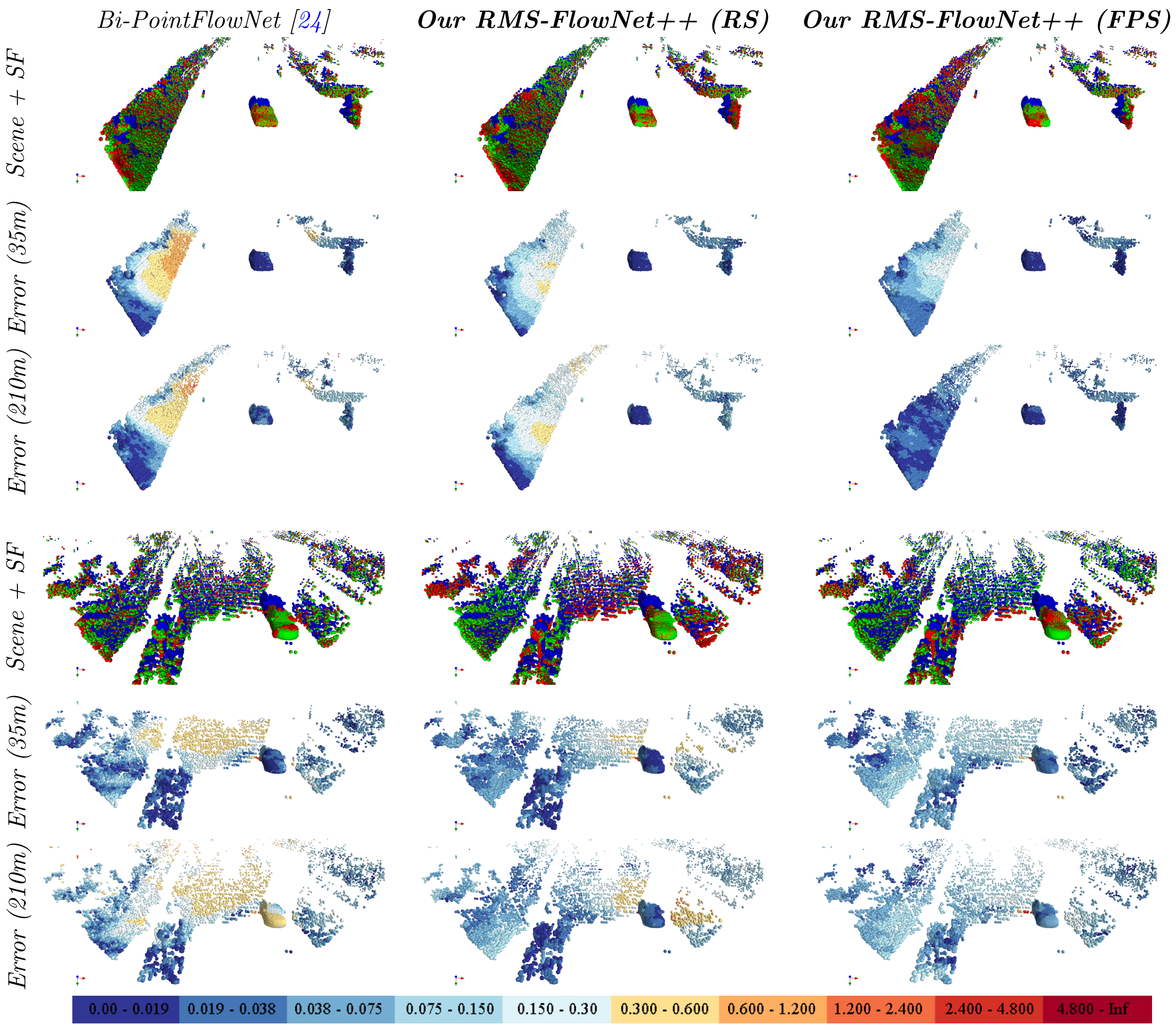


**Fig. 10**: Two examples taken from KITTI$_s$ show the impact of our RMS-FlowNet++ compared to the competing method Bi-PointFlowNet [24]. The scene of each example (first and forth rows) visualizes $P^t$ as blue color and the predicted and ground truth scene flow after adding them to $P^t$ in green and red color, respectively. The error map of each scene (second, third, fifth and sixth rows) shows the end-point error in meters according to the color map shown in the last row. Our RMS-FlowNet++ shows lower errors (dark blue) over a wide area of the observed scene, compared to the competing method.

layers (see Fig. 5). Focusing on our stacked attention layers, we verify three important aspects of our stacked attention design on the FT3D$_s$ data set as follows:

1. Adding the feature of reference frame $f_i^t$ (as in Eq. (5)) to the input of the attention mechanism.
2. Adding the residual connection (Res. Conn.) as in Figure 5 or $e_{2i}^t$ as in Eq. (10).
3. Encoding of the spatial locations to the features and the concatenation to the $\hat{f}_i^t$ in the Eq. (5).

Combining all of the above components yields more accurate results, as verified in Table 5.

**Table 4**: We explore our improvements in RMS-FlowNet++ compared to our preliminary work RMS-FlowNet [27]. To speed up the experiments, we use FPS during training and evaluation. The first line corresponds to the method in [27].

| $K_p$ | Graph Rep. | Decoder | $2^{nd}$ Embedding | Bidirectional Map | FT3D$_s$ [62] EPE3D [m] | FT3D$_s$ [62] Acc3DR [%] | KITTI$_s$[26] EPE3D [m] | KITTI$_s$[26] Acc3DR [%] |
|---|---|---|---|---|---|---|---|---|
| 33 | ✗ | ✓ | ✗ | ✗ | 0.051 | 95.60 | 0.047 | 95.80 |
| 16 | ✗ | ✓ | ✗ | ✗ | 0.054 | 95.15 | 0.067 | 91.72 |
| 16 | ✓ | ✓ | ✗ | ✗ | 0.041 | 96.85 | 0.043 | 94.16 |
| 16 | ✓ | ✗ | ✗ | ✗ | 0.034 | 97.63 | 0.039 | 94.57 |
| 16 | ✓ | ✗ | ✓ | ✗ | 0.033 | 97.77 | 0.036 | 95.52 |
| 16 | ✓ | ✗ | ✓ | ✓ | **0.029** | **98.09** | **0.030** | **97.63** |
| 16 | **Trained with RS with $K_p = 20$** | | | | **0.029** | **98.10** | **0.027** | **97.67** |

**Table 5**: We verify the aspects of our stacked attention in flow embedding on FT3D$_s$ data set. To speed up the experiments, we use FPS for training and evaluation.

| Spatial Encoding | $f_i^t$ | Res. Conn. | EPE3D [m] ↓ | Acc3DR [%] ↑ |
|---|---|---|---|---|
| ✗ | ✗ | ✗ | 0.050 | 95.13 |
| ✓ | ✗ | ✗ | 0.040 | 96.84 |
| ✓ | ✓ | ✗ | 0.038 | 97.20 |
| ✗ | ✓ | ✓ | 0.038 | 96.96 |
| ✓ | ✓ | ✓ | **0.029** | **98.09** |
| **Trained with RS with $K_p = 20$** | | | **0.029** | **98.10** |

**Training with FPS vs. RS:** First, we train with FPS using different numbers of $K_p$ and evaluate with the same numbers used in training to determine the appropriate number of $K_p$ (*i.e.*, the correct correspondence set) that gives the best results, as shown in Table 6. We find that small correspondence sets such as 8 and 12 have lower accuracy than 16 and 20, which both give roughly comparable results, making them appropriate numbers, but at the cost of a higher number of FLOPs. Based on this, we train with RS and these determined numbers of $K_p$ (*i.e.*, 16 and 20). After evaluation, we find that $K_p = 20$ works best with RS, as shown in Table 6. Based on this, we set $K_p$ to 16 and 20 for FPS and RS, respectively. We then perform cross evaluations with both sampling techniques to verify which sampling method generalizes better. The results are shown in Table 7. RS generalizes much better than FPS. It even improves the results when evaluated with FPS, compared to the original training with FPS.

**Impact of Augmentation:** We examine the effect of our data augmentation (*cf.* Section 4.3) individually. The results of these experiments are shown in Table 8. As mentioned before, to speed up the tests, we train with FPS and a correspondence set of 16.

When training without any augmentation, the results are good on FT3D$_s$, but generalize poorly to the real-world data of KITTI$_s$. When we randomize the initial sampling in each epoch (*i.e.*, change the spatial locations for each epoch), the results on the FT3D$_s$ data set drop slightly, but the accuracy and end-point error on KITTI$_s$ are significantly improved. We observe a similar behavior when adding only the geometric augmentation, with an even larger positive impact on KITTI$_s$. Both augmentation strategies together

**Table 6**: We evaluate the number of KNNs that can be used for both FPS and RS sampling techniques. Training and evaluation are always performed with the same sampling technique and number of nearest neighbors. FLOPs do not count for the sampling or for the nearest neighbor search.

| Sampling Technique | $K_p$ | FT3D$_s$ [62] EPE3D [m] | FT3D$_s$ [62] Acc3DR [%] | KITTI$_s$[26] EPE3D [m] | KITTI$_s$[26] Acc3DR [%] | FLOPs [G] |
|---|---|---|---|---|---|---|
| **FPS** | 8 | 0.035 | 97.70 | 0.043 | 95.38 | **17.16** |
| | 12 | 0.031 | 98.04 | 0.031 | 97.54 | 23.55 |
| | **16** | **0.029** | **98.09** | 0.030 | **97.63** | 29.93 |
| | **20** | **0.029** | 98.07 | **0.029** | 97.56 | 36.32 |
| **RS** | 16 | 0.034 | 97.50 | 0.047 | 94.21 | **29.93** |
| | **20** | **0.033** | **97.51** | **0.035** | **96.53** | 36.32 |

**Table 7**: We evaluate the generalization of each sampling technique to the other on FT3D$_s$ and KITTI$_s$. Training with RS can generalize very well when evaluated with FPS.

| Train\Test | FPS FT3D$_s$ [62] Acc3DR [%] | FPS KITTI$_s$ [26] Acc3DR [%] | RS FT3D$_s$ [62] Acc3DR [%] | RS KITTI$_s$ [26] Acc3DR [%] |
|---|---|---|---|---|
| FPS | 98.09 | 97.63 | 85.73 | 84.53 |
| **RS** | **98.10** | **97.67** | **97.51** | **96.53** |

improve the overall results on both data sets and provide the best generalization from synthetic to real scenes.

### 4.8 Limitations

In terms of accuracy, there are three major limitations: 1) Any errors at the coarsest level can accumulate in the higher resolution layers, degrading the overall accuracy. 2) Our one-to-one bidirectional matching at the coarsest resolution can lead to mismatches if the scene contains repetitive patterns (*e.g.*, along road pillars or traffic barriers). 3) Areas of homogeneous geometry (*e.g.*, road surfaces or grass along the road) pose a challenge to our model, especially when RS is used (*cf.* Fig. 10). The error increases significantly when these untextured objects are represented or scanned by high-density points, as shown in Fig. 11. In terms of efficiency, two major limitations remain: 1) To maintain accuracy at higher input densities, it is necessary to increase the resolution rates in the downsampling layers, which increases the runtime and memory requirements (*cf.* Fig. 7). 2) The KNN search dominates the computational complexity as the input density increases.

## 5 Conclusion

In this paper, we propose RMS-FlowNet++ – an efficient and fully supervised network for multi-scale scene flow estimation in high-density point clouds. By using Random-Sampling (RS) during feature extraction, we are able to boost the runtime and memory footprint for an efficient processing of point clouds with an unmatched maximum density. The novel Flow-Embedding module (called *Patch-to-Dilated-Patch*), resolves the prominent challenges in using RS for scene flow estimation. Compared to our preliminary work [27], we reduce the operations in our network and improve the accuracy. We demonstrate the advantages of RS over FPS on high-density point clouds and its ability to generalize to FPS during inference. We provide an intensive benchmark, in which our RMS-FlowNet++ achieves the best results in terms of accuracy, generalization, and runtime compared to the previous state-of-the-art. We also investigate the robustness of our network to occlusions and explore its ability to operate on long-range point clouds (*i.e.*, up to 210 meters).

In the future, we would like to improve our model by fusing the 3D information of point clouds with textural 2D information captured by RGB cameras. We also plan to add ego-motion estimation to our model to avoid inaccuracies in static, homogeneous areas such as the road surface.

**Table 8**: We study the effect of augmentation on FT3D$_s$ and KITTI$_s$. To speed up the experiments, we train and evaluate using FPS.

| Spatial | Geometry | **FT3D$_s$ [62]** | | **KITTI$_s$ [26]** | |
|---|---|---|---|---|---|
| | | **EPE3D** [m] | **Acc3DR** [%] | **EPE3D** [m] | **Acc3DR** [%] |
| ✗ | ✗ | 0.032 | 97.87 | 0.067 | 87.23 |
| ✓ | ✗ | 0.033 | 97.63 | 0.044 | 93.67 |
| ✗ | ✓ | 0.035 | 97.85 | 0.031 | 97.57 |
| ✓ | ✓ | 0.029 | **98.08** | **0.030** | **97.63** |

## Declarations

This version of the article has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/s11263-024-02093-9.

**Funding.** This work was partially funded by the Federal Ministry of Education and Research Germany under the project DECODE (01IW21001), partially in the funding program Photonics Research Germany under the project FUMOS (13N16302) and partially by EU Horizon Europe Framework Program under the grant agreement 101092889 (SHARESPACE).

**Availability of data and materials.** The data sets analyzed in the current study are available at the following links:

- KITTI data set: [https://www.cvlibs.net/datasets/kitti/eval_scene_flow.php]
- FlyingThings3D and FlyingThings3D subset data sets: [https://academictorrents.com/userdetails.php?id=9551], [https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html]

Other repository names of the established preprocessing scripts for the data sets with the persistent web links are available in the manuscript.

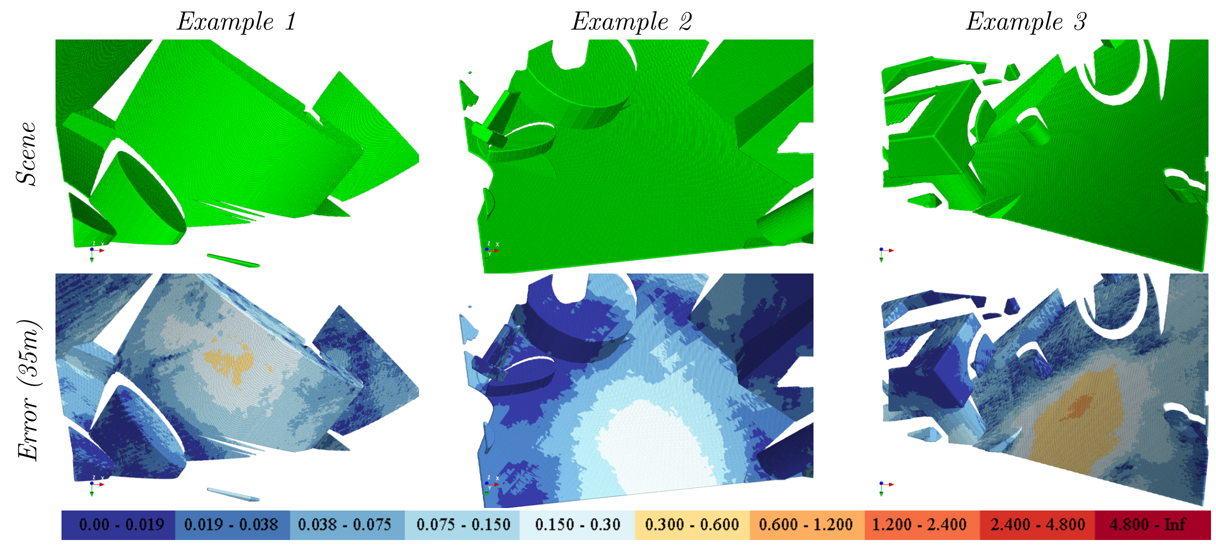


**Fig. 11**: Three examples from the non-occluded versions of FT3D$_s$ show the failure cases of our RMS-FlowNet++ with high point densities using RS. The scene of each example (first row) visualizes $P^t$ as green color. The error map of each scene (second row) shows the end-point error in meters according to the color map shown in the last row.

## References

[1] Franke, U.; Rabe, C.; Badino, H.; Gehrig, S. 6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception. Joint Pattern Recognition Symposium. 2005.

[2] Huguet, F.; Devernay, F. A Variational Method for Scene Flow Estimation from Stereo Sequences. IEEE International Conference on Computer Vision (ICCV). 2007.

[3] Wedel, A.; Rabe, C.; Vaudrey, T.; Brox, T.; Franke, U.; Cremers, D. Efficient Dense Scene Flow from Sparse or Dense Stereo Data. European Conference on Computer Vision (ECCV). 2008.

[4] Ilg, E.; Saikia, T.; Keuper, M.; Brox, T. Occlusions, Motion and Depth Boundaries with a Generic Network for Disparity, Optical Flow or Scene Flow Estimation. European Conference on Computer Vision (ECCV). 2018.

[5] Chen, Y.; Gool, L. V.; Schmid, C.; Sminchisescu, C. Consistency Guided Scene Flow Estimation. European Conference on Computer Vision (ECCV). 2020.

[6] Schuster, R.; Wasenmüller, O.; Unger, C.; Kuschk, G.; Stricker, D. SceneFlowFields++: Multi-frame Matching, Visibility Prediction, and Robust Interpolation for Scene Flow Estimation. *International Journal of Computer Vision (IJCV)* **2020**,

[7] Schuster, R.; Wasenmüller, O.; Kuschk, G.; Bailer, C.; Stricker, D. SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences. IEEE Winter Conference on Applications of Computer Vision (WACV). 2018.

[8] Saxena, R.; Schuster, R.; Wasenmüller, O.; Stricker, D. PWOC-3D: Deep Occlusion-Aware End-to-End Scene Flow Estimation. *IEEE International Conference on Intelligent Vehicles Symposium (IV)* **2019**,

[9] Gu, X.; Wang, Y.; Wu, C.; Lee, Y. J.; Wang, P. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[10] Jampani, V.; Kiefel, M.; Gehler, P. V. Learning Sparse High Dimensional Filters: Image

Filtering, Dense CRFs and Bilateral Neural Networks. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[11] Gojcic, Z.; Litany, O.; Wieser, A.; Guibas, L. J.; Birdal, T. Weakly Supervised Learning of Rigid 3D Scene Flow. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[12] Li, B.; Zheng, C.; Giancola, S.; Ghanem, B. SCTN: Sparse Convolution-Transformer Network for Scene Flow Estimation. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). 2022.

[13] Choy, C.; Gwak, J.; Savarese, S. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[14] Wu, W.; Qi, Z.; Fuxin, L. PointConv: Deep Convolutional Networks on 3D Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[15] Qi, C. R.; Yi, L.; Su, H.; Guibas, L. J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems (NIPS). 2017.

[16] Liu, X.; Qi, C. R.; Guibas, L. J. FlowNet3D: Learning Scene Flow in 3D Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[17] Wei, Y.; Wang, Z.; Rao, Y.; Lu, J.; Zhou, J. PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[18] Wu, W.; Wang, Z. Y.; Li, Z.; Liu, W.; Fuxin, L. PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation. European Conference on Computer Vision (ECCV). 2020.

[19] Kittenplon, Y.; Eldar, Y. C.; Raviv, D. FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[20] Wang, G.; Wu, X.; Liu, Z.; Wang, H. Hierarchical Attention Learning of Scene Flow in 3D Point Clouds. *IEEE Transactions on Image Processing (TIP)* **2021**,

[21] Gu, X.; Tang, C.; Yuan, W.; Dai, Z.; Zhu, S.; Tan, P. RCP: Recurrent Closest Point for Scene Flow Estimation on 3D Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022.

[22] Wang, G.; Hu, Y.; Wu, X.; Wang, H. Residual 3D Scene Flow Learning with Context-Aware Feature Extraction. *IEEE Transactions on Instrumentation and Measurement (TIM)* **2022**,

[23] Wang, G.; Hu, Y.; Liu, Z.; Zhou, Y.; Tomizuka, M.; Zhan, W.; Wang, H. What Matters for 3D Scene Flow Network. European Conference on Computer Vision (ECCV). 2022.

[24] Cheng, W.; Ko, J. H. Bi-PointFlowNet: Bidirectional Learning for Point Cloud Based Scene Flow Estimation. European Conference on Computer Vision (ECCV). 2022.

[25] Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078* **2014**,

[26] Menze, M.; Geiger, A. Object Scene Flow for Autonomous Vehicles. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2015.

[27] Battrawy, R.; Schuster, R.; Mahani, M.-A. N.; Stricker, D. RMS-FlowNet: Efficient and Robust Multi-Scale Scene Flow Estimation for Large-Scale Point Clouds. IEEE International Conference on Robotics and Automation (ICRA). 2022.

[28] Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2020.

[29] Vedula, S.; Baker, S.; Rander, P.; Collins, R.; Kanade, T. Three-Dimensional Scene Flow. IEEE International Conference on Computer Vision (ICCV). 1999.

[30] Hadfield, S.; Bowden, R. Kinecting the dots: Particle Based Scene Flow From Depth Sensors. IEEE International Conference on Computer Vision (ICCV). 2011.

[31] Hornacek, M.; Fitzgibbon, A.; Rother, C. SphereFlow: 6 DoF Scene Flow from RGB-D Pairs. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2014.

[32] Quiroga, J.; Brox, T.; Devernay, F.; Crowley, J. Dense Semi-rigid Scene Flow Estimation from RGBD Images. European Conference on Computer Vision (ECCV). 2014.

[33] Jaimez, M.; Souiai, M.; Gonzalez-Jimenez, J.; Cremers, D. A Primal-Dual Framework for Real-Time Dense RGB-D Scene Flow. IEEE International Conference on Robotics and Automation (ICRA). 2015.

[34] Jaimez, M.; Souiai, M.; Stückler, J.; Gonzalez-Jimenez, J.; Cremers, D. Motion Cooperation: Smooth Piece-Wise Rigid Scene Flow from RGB-D Images. International Conference on 3D Vision (3DV). 2015.

[35] Sun, D.; Sudderth, E. B.; Pfister, H. Layered RGBD Scene Flow Estimation. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2015.

[36] Shao, L.; Shah, P.; Dwaracherla, V.; Bohg, J. Motion-based Object Segmentation based on Dense RGB-D Scene Flow. *IEEE Robotics and Automation Letters (RA-L)* **2018**,

[37] Teed, Z.; Deng, J. RAFT-3D: Scene Flow using Rigid-Motion Embeddings. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[38] Jiang, H.; Sun, D.; Jampani, V.; Lv, Z.; Learned-Miller, E.; Kautz, J. SENSE: a Shared Encoder Network for Scene-flow Estimation. IEEE/CVF International Conference on Computer Vision (ICCV). 2019.

[39] Ma, W.-C.; Wang, S.; Hu, R.; Xiong, Y.; Urtasun, R. Deep Rigid Instance Scene Flow. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[40] Dewan, A.; Caselitz, T.; Tipaldi, G. D.; Burgard, W. Rigid Scene Flow for 3D LiDAR Scans. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2016.

[41] Ushani, A. K.; Wolcott, R. W.; Walls, J. M.; Eustice, R. M. A Learning Approach for Real-Time Temporal Scene Flow Estimation from LIDAR Data. IEEE International Conference on Robotics and Automation (ICRA). 2017.

[42] Qi, C. R.; Su, H.; Mo, K.; Guibas, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

[43] Su, H.; Jampani, V.; Sun, D.; Maji, S.; Kalogerakis, E.; Yang, M.-H.; Kautz, J. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[44] Ushani, A. K.; Eustice, R. M. Feature Learning for Scene Flow Estimation from LIDAR. Conference on Robot Learning (CoRL). 2018.

[45] Wang, S.; Suo, S.; Ma, W.-C.; Pokrovsky, A.; Urtasun, R. Deep Parametric Continuous Convolutional Neural Networks. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[46] Behl, A.; Paschalidou, D.; Donné, S.; Geiger, A. PointFlowNet: Learning Representations for Rigid Motion Estimation from

Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[47] Puy, G.; Boulch, A.; Marlet, R. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. European Conference on Computer Vision (ECCV). 2020.

[48] Li, R.; Lin, G.; He, T.; Liu, F.; Shen, C. HCRF-Flow: Scene Flow from Point Clouds with Continuous High-order CRFs and Position-aware Flow Embedding. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[49] Wang, H.; Pang, J.; Lodhi, M. A.; Tian, Y.; Tian, D. FESTA: Flow Estimation via Spatial-Temporal Attention for Scene Point Clouds. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

[50] Dong, G.; Zhang, Y.; Li, H.; Sun, X.; Xiong, Z. Exploiting Rigidity Constraints for LiDAR Scene Flow Estimation. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022.

[51] Kiefel, M.; Jampani, V.; Gehler, P. V. Permutohedral Lattice CNNs. *arXiv preprint arXiv:1412.6618* **2014**,

[52] Li, Y.; Tofighi, M.; Monga, V.; Eldar, Y. C. AN ALGORITHM UNROLLING APPROACH TO DEEP IMAGE DEBLURRING. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2019.

[53] Monga, V.; Li, Y.; Eldar, Y. C. Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. *IEEE Signal Processing Magazine* **2021**,

[54] Teed, Z.; Deng, J. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. European Conference on Computer Vision (ECCV). 2020.

[55] Titouan, V.; Courty, N.; Tavenard, R.; Flamary, R. Optimal Transport for structured data with application on graphs. International Conference on Machine Learning. 2019.

[56] Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; Brox, T. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

[57] Sun, D.; Yang, X.; Liu, M.-Y.; Kautz, J. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[58] Blanco, J. L.; Rai, P. K. nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees. https://github.com/jlblancoc/nanoflann, 2014.

[59] Yang, B.; Wang, S.; Markham, A.; Trigoni, N. Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. *International Journal of Computer Vision (IJCV)* **2020**,

[60] Zhang, W.; Xiao, C. PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[61] Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; Solomon, J. M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (ToG)* **2019**,

[62] Mayer, N.; Ilg, E.; Hausser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; Brox, T. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[63] Battrawy, R.; Schuster, R.; Wasenmüller, O.; Rao, Q.; Stricker, D. LiDAR-Flow: Dense Scene Flow Estimation from Sparse LiDAR and Stereo Images. IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS). 2019.

[64] Rishav, R.; Battrawy, R.; Schuster, R.; Wasenmüller, O.; Stricker, D. DeepLiDARFlow: A Deep Learning Architecture For Scene Flow Estimation Using Monocular Camera and Sparse LiDAR. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2020.

[65] Ouyang, B.; Raviv, D. Occlusion Guided Scene Flow Estimation on 3D Point Clouds. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 2021.

[66] Ouyang, B.; Raviv, D. Occlusion Guided Self-supervised Scene Flow Estimation on 3D Point Clouds. International Conference on 3D Vision (3DV). 2021.