Supplementary Material for "PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds"

Yi Wei[1,2,3]*, Ziyi Wang[1,2,3]*, Yongming Rao[1,2,3]*, Jiwen Lu[1,2,3]†, Jie Zhou[1,2,3,4]
[1]Department of Automation, Tsinghua University, China
[2]State Key Lab of Intelligent Technologies and Systems, China
[3]Beijing National Research Center for Information Science and Technology, China
[4]Tsinghua Shenzhen International Graduate School, Tsinghua University, China
{y-wei19, wziyi20}@mails.tsinghua.edu.cn; raoyongming95@gmail.com; {lujiwen, jzhou}@tsinghua.edu.cn

In this supplementary material, we first provide the detailed network architecture of PV-RAFT. Then in Section B, we show additional experimental results to demonstrate the necessity of point-voxel correlation fields.

A. Network Architecture

The architecture of our network can be divided into four parts: (1) Feature Extractor, (2) Correlation Module (3) Iterative Update Module (4) Refinement Module. In this section, we will introduce the implementation details of each structure.

A.1. Feature Extractor

Backbone Feature Extractor We first construct a graph $\mathcal{G}$ of input point cloud P, that contains neighborhood information of each point. Then we follow FLOT which is based on PointNet++ to design the feature extractor.

The feature extractor consists of three SetConvs to lift feature dimension: $3 \rightarrow 32 \rightarrow 64 \rightarrow 128$. In each SetConv, we first locate neighbor region $\mathcal{N}$ of P and use $F = concat(F_{\mathcal{N}} - F_P, F_{\mathcal{N}})$ as input features, where $concat$ stands for concatenation operation. Then features F are fed into the pipeline: $FC \rightarrow pool \rightarrow FC \rightarrow FC$. Each FC block consists of a 2D convolutional layer, a group normalization layer and a leaky ReLU layer with the negative slope as 0.1. If we denote the input and output dimension of the SetConv as d_i, d_o, then the dimension change for FC blocks is: $d_i \rightarrow d_{mid} = (d_i + d_o)/2 \rightarrow d_o \rightarrow d_o$. However, if $d_i = 3$, then d_{mid} is set to $d_o/2$. The $pool$ block performs the max-pooling operation.

Context Feature Extractor The context feature extractor aims to encode context features of P_1. It has exactly the same structure as the backbone feature extractor, but without weight sharing.

*Equal Contribution
†Corresponding author

A.2. Correlation Module

Point Branch The extracted KNN features $F_p(P)$ are first concatenated with position features $C(\mathcal{N}_P) - C(P)$, then it is fed into a block that consists of one point-wise convolutional layer, one group normalization layer, one p-ReLU layer, one max-pooling layer and one point-wise convolutional layer. The feature dimension is updated from 4 to 64.

Voxel Branch The extracted voxel features $F_v(P)$ are fed into a block that consists of one point-wise convolutional layer, one group-norm layer, one p-ReLU layer and one point-wise convolutional layer. The feature dimension is updated as: $a^3 * l \rightarrow 128 \rightarrow 64$, where $a = 3$ is the resolution hyper-parameter and $l = 3$ is the pyramid level.

A.3. Iterative Update Module

The update block consists of three parts: Motion Encoder, GRU Module and Flow Head.

Motion Encoder The inputs of motion encoder are flow f and correlation features $\mathbf{C}$. These two inputs are first fed into a non-share convolutional layer and a ReLU layer separately to get f' and $\mathbf{C}'$. Then they are concatenated and fed into another convolutional layer and a ReLU layer to get f''. Finally we concat f and f'' to get motion features f_m.

GRU Module The inputs of GRU module are context features and motion features. The update process has already been introduced in our main paper.

Flow Head The input of the flow head is the final hidden state h_t of GRU module. h_t is first fed into a 2D convolutional layer to get h_t'. On the other hand, h_t is fed into a SetConv layer, introduced in backbone feature extractor, to get h_t''. Then we concatenate h_t' and h_t'' and pass through a 2D convolutional layer to adjust the feature dimension to 3. The output is used to update flow prediction.

Table 1: The necessity of point-voxel correlation fields. We conducted experiments on FlyingThings3D dataset without refinement. KNN pyramid means we concatenated correlation features with different K.

Modality	Hyperparameters	EPE(m)↓	Acc Strict↑	Acc Relax↑	Outliers↓
KNN	$K = 32$	0.0741	0.6111	0.8868	0.4549
	$K = 64$	0.2307	0.1172	0.3882	0.8547
	$K = 128$	0.6076	0.0046	0.0333	0.9979
KNN pyramid	$K = 16, 32, 64$	0.1616	0.2357	0.6062	0.7318
	$K = 32, 64, 128$	0.4841	0.0158	0.0885	0.9882
voxel pyramid	$r = 0.0625, l = 3$	0.1408	0.5126	0.8057	0.5340
	$r = 0.125, l = 3$	0.0902	0.5345	0.8533	0.5085
	$r = 0.25, l = 3$	0.0712	0.6146	0.8983	0.4492
	$r = 0.0625, l = 5$	0.0672	0.6325	0.9131	0.4023
point-voxel	$K = 32, r = 0.25, l = 3$	**0.0534**	**0.7348**	**0.9418**	**0.3645**

A.4. Refinement Module

The input of the refinement module is the predicted flow f^*. The refinement module consists of three SetConv modules and one Fully Connected Layer. The SetConv module has been introduced in feature extractor part and the dimension is changed as: $3 \rightarrow 32 \rightarrow 64 \rightarrow 128$. The output feature f_r^* of fully connected layer is of dimension 3. We implement a residual mechanism to get the final prediction that combines f^* and f_r^*.

B. Additional Experiments

As mentioned in Section 4.3, we tried intuitive solutions to model all-pairs correlations. We conducted experiments on FlyingThings3D dataset without refinement. Specifically, for the point branch, we leveraged more nearest neighbors to encode large receptive fields. When only using the voxel branch, we reduce the side length r of lattices to capture fine-grained relations. Moreover, we adopted the KNN search with different K simultaneously to construct a KNN pyramid , which aims to aggregate the feature with different receptive fields. However, as shown in Table 1, all these tries failed to achieve promising results. We argue that this may because of the irregularity of point clouds. On the one hand, for the region with high point density, a large number of neighbors still lead to a small receptive field. On the other hand, although we reduce side length, the voxel branch cannot extract point-wise correlation features. Integrating these two types of correlations, the proposed point-voxel correlation fields help PV-RAFT to capture both local and long-range dependencies.