Supplementary Material: HCRF-Flow: Scene Flow from Point Clouds with Continuous High-order CRFs and Position-aware Flow Embedding

A. Details about the inference of Con-HCRFs

In this section, we show how to derive the mean field inference algorithm of our proposed Con-HCRFs.

As introduced in Sec. 3.1, given a point cloud $\boldsymbol{P}$ with n points, we use $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n]$ to represent a matrix of 3D displacements corresponding to all n points in $\boldsymbol{P}$, where each $\boldsymbol{y}_i \in \mathbb{R}^3$. We model the conditional probability distribution with the following density function

$$\Pr(\boldsymbol{Y}|\boldsymbol{P}) = \frac{1}{\mathrm{Z}(\boldsymbol{P})}\exp(-E(\boldsymbol{Y}|\boldsymbol{P})), \tag{A}$$

where E is the energy function and Z is the partition function defined as: $Z(\boldsymbol{P}) = \int_Y \exp(-E(\boldsymbol{Y}|\boldsymbol{P}))\mathrm{d}\boldsymbol{Y}$.

To refine the predicted scene flow in both point level and region level, the energy function of our continuous high-order CRFs is designed as:

$$E(\boldsymbol{Y}|\boldsymbol{P}) = \sum_i \psi^U(\boldsymbol{y}_i, \boldsymbol{P}) + \sum_{i,j\in\mathcal{N}(i)} \psi^B(\boldsymbol{y}_i, \boldsymbol{y}_j, \boldsymbol{P}) + \sum_{\mathcal{V}\in\boldsymbol{V}_{set}} \sum_{i\in\mathcal{V}} \psi^{SV}(\boldsymbol{y}_i, \boldsymbol{Y}_{\mathcal{V}-i}, \boldsymbol{P}), \tag{B}$$

According to the definition of each potential term in Sec. 3.2, combing Eq. (8), Eq. (9), Eq. (10) with Eq. B, the energy function can be written as:

$$E(\boldsymbol{Y}) = \sum_i \|\boldsymbol{y}_i - \boldsymbol{z}_i\|_2^2 + \sum_{i,j\in\mathcal{N}(i)} \sum_{c=1}^{C} \alpha_c K_{ij}^{(c)} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 + \sum_{\mathcal{V}\in\boldsymbol{V}_{set}} \sum_{i\in\mathcal{V}} \beta \|\boldsymbol{y}_i - \boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i})\|_2^2. \tag{C}$$

Here, $\|\cdot\|_2$ denotes the L^2 norm of a vector, $\boldsymbol{z}_i$ is the initial 3D displacement at point i produced by PAFE module. $\mathcal{N}(i)$ is the set of neighboring points of center point i, and $K_{ij}^{(c)}$ is a weight to specify the relation between the points i and j. $\boldsymbol{V}_{set}$ is the set of rigid regions in the whole point cloud and $\boldsymbol{Y}_{\mathcal{V}-i}$ is a matrix composed by the scene flow of points belonging to the region $\mathcal{V}$ with the point i excluded. And the point set without the point i is denoted as $\mathcal{V} - i$. $\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i})$ is the rigid displacement for the point i, whose computation process is given in Eq. (11), Eq. (14), and Eq. (15). α_c and β are trainable model parameters.

As discussed in Sec. 3.3, following [7], we adopt mean field theory [2] to approximate the distribution $\Pr(\boldsymbol{Y})$ into a product of independent marginals, i.e., $Q(\boldsymbol{Y}) = \prod_{i=1}^{n} Q_i(\boldsymbol{y}_i)$. According to [1], by minimizing the KL-divergence between $\Pr$ and Q, the solution for Q can be written as:

$$\log(Q_i(\boldsymbol{y}_i)) = \mathrm{E}_{j\neq i}[\Pr(\boldsymbol{Y})] + const, \tag{D}$$

where $\mathrm{E}_{j\neq i}$ represents an expectation under Q distributions over all variable $\boldsymbol{y}_j$ for $j \neq i$. Combing Eq. C and Eq. D, we

have:

$$
\begin{aligned}
\log(Q_i(\boldsymbol{y}_i)) &= \|\boldsymbol{y}_i - \boldsymbol{z}_i\|_2^2 + 2\sum_{j\in\mathcal{N}(i)}\sum_{c=1}^{C}\alpha_c K_{ij}^{(c)}\mathrm{E}_{j\neq i}[\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2] + \beta\mathrm{E}_{j\neq i}[\|\boldsymbol{y}_i - \boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i})\|_2^2] + const, \\
&= (\boldsymbol{y}_i - \boldsymbol{z}_i)^{\mathsf{T}}(\boldsymbol{y}_i - \boldsymbol{z}_i) + 2\sum_{j\in\mathcal{N}(i)}\sum_{c=1}^{C}\alpha_c K_{ij}^{(c)}\mathrm{E}_{j\neq i}[(\boldsymbol{y}_i - \boldsymbol{y}_j)^{\mathsf{T}}(\boldsymbol{y}_i - \boldsymbol{y}_j)] \\
&\quad + \beta\mathrm{E}_{j\neq i}[(\boldsymbol{y}_i - \boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i}))^{\mathsf{T}}(\boldsymbol{y}_i - \boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i}))] + const, \\
&= (\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{y}_i - 2\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{z}_i) + 2\sum_{j\in\mathcal{N}(i)}\sum_{c=1}^{C}\alpha_c K_{ij}^{(c)}(\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{y}_i - 2\boldsymbol{y}_i^{\mathsf{T}}\mathrm{E}_{j\neq i}[\boldsymbol{y}_j]]) \\
&\quad + \beta(\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{y}_i - 2\boldsymbol{y}_i^{\mathsf{T}}\mathrm{E}_{j\neq i}[\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i})]) + const,
\end{aligned}
\tag{E}
$$

where $\mathrm{E}_{j\neq i}[\boldsymbol{y}_j]$ is the expectation over variable $\boldsymbol{y}_j$; and $\mathrm{E}_{j\neq i}[\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i})]$ is the expectation of the rigid displacement of the point i. Here, we approximate $\mathrm{E}_{j\neq i}[\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{Y}_{\mathcal{V}-i})]$ with $\boldsymbol{g}(\boldsymbol{p}_i, \mathrm{E}_{j\neq i}[\boldsymbol{Y}_{\mathcal{V}-i}])$, i.e., we use the expectations over variables $\boldsymbol{Y}_{\mathcal{V}-i}$ as input to compute the expectation of the rigid displacement of the point i by Eq. (11), Eq. (14), and Eq. (15). In this way, the approximated $\log(Q_i(\boldsymbol{y}_i))$ can be written as:

$$
\begin{aligned}
\log(Q_i(\boldsymbol{y}_i)) = (\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{y}_i - 2\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{z}_i) + 2\sum_{j\in\mathcal{N}(i)}\sum_{c=1}^{C}\alpha_c K_{ij}^{(c)}(\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{y}_i - 2\boldsymbol{y}_i^{\mathsf{T}}\mathrm{E}_{j\neq i}[\boldsymbol{y}_j]]) \\
+ \beta(\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{y}_i - 2\boldsymbol{y}_i^{\mathsf{T}}\boldsymbol{g}(\boldsymbol{p}_i, \mathrm{E}_{j\neq i}[\boldsymbol{Y}_{\mathcal{V}-i}])) + const.
\end{aligned}
\tag{F}
$$

As shown in Eq. F, each $\log(Q_i(\boldsymbol{y}_i))$ is a quadratic form with respect to $\boldsymbol{y}_i$. Following [7], we represent it as a multivariate normal distribution, and the mean field update for mean $\dot{\mu}_i$ and normalization parameter σ_i can be written as:

$$
\sigma_i = \frac{1}{2(1 + 2\sum_{c=1}^{C}\sum_{j\in\mathcal{N}(i)}\alpha_c K_{ij}^{(c)} + \beta)},
\tag{G}
$$

$$
\dot{\boldsymbol{\mu}}_i = 2\sigma_i(\boldsymbol{z}_i + 2\sum_{c=1}^{C}\sum_{j\in\mathcal{N}(i)}\alpha_c K_{ij}^{(c)}\boldsymbol{\mu}_j + \beta\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{M}_{\mathcal{V}-i})).
\tag{H}
$$

Here, $\boldsymbol{M}_{\mathcal{V}-i}$ is a set of mean $\boldsymbol{\mu}_j$ for all $j \in \mathcal{V} - i$; and σ_i is the diagonal element of covariance $\boldsymbol{\Sigma}_i$. The covariance $\boldsymbol{\Sigma}_i$ is a diagonal matrix and the diagonal elements are the same.

We observe that there usually exist hundreds of points in a supervoxel, which makes the rigid parameters computed on all points in the supervoxel excluding the point i vary close to the rigid parameters computed on all points in the supervoxel, i.e., $[\boldsymbol{R}^*(\boldsymbol{M}_{\mathcal{V}-i}), \boldsymbol{t}^*(\boldsymbol{M}_{\mathcal{V}-i})]$ is vary close to $[\boldsymbol{R}^*(\boldsymbol{M}_{\mathcal{V}}), \boldsymbol{t}^*(\boldsymbol{M}_{\mathcal{V}})]$. Thus, in practice, we approximate $\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{M}_{\mathcal{V}-i})$ in Eq. H with $\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{M}_{\mathcal{V}})$, and the approximated mean $\boldsymbol{\mu}_i$ is:

$$
\boldsymbol{\mu}_i = 2\sigma_i(\boldsymbol{z}_i + 2\sum_{c=1}^{C}\sum_{j\in\mathcal{N}(i)}\alpha_c K_{ij}^{(c)}\boldsymbol{\mu}_j + \beta\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{M}_{\mathcal{V}})).
\tag{I}
$$

After this approximation, we only need to calculate a set of rigid motion parameters for each supervoxel rather than for each point, which greatly reduces the time complexity.

B. Implementation details

In this section, we introduce the Implementation details of our HCRF-Flow. In the Con-HCRFs module, we utilize the algorithm proposed in [3] for supervoxel segmentation. In [3], the task of supervoxel segmentation is formalized as a subset selection problem and local information, such as surface normal and point position, is utilized to solve it. Some supervoxel segmentation results are show in Fig. 1 and Fig. 2.

In the Con-HCRFs module, the desired point number of supervoxel, a hyperparameter to control the wanted supervoxel size, is set to 140. In the pairwise term of the Con-HCRFs, we adopt two kinds of Gaussian kernel, i.e. C is 2. For a pair of

neighboring points i and j, we define the point coordinates as $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, and define the surface normals as $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$. For the first Gaussian kernel, we use the point coordinate as feature to evaluate the similarity between points i and j:

$$K_{ij}^{(1)} = \exp(-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2}{2\theta_p^2}). \tag{J}$$

For the second Gaussian kernel, we adopt both point coordinate and surface normal as feature to evaluate the similarity between points i and j:

$$K_{ij}^{(2)} = \exp(-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2}{2\theta_p^2} - \frac{\|\boldsymbol{n}_i - \boldsymbol{n}_j\|_2^2}{2\theta_n^2}). \tag{K}$$

In our experiment, we set θ_p to 0.32 and θ_n to 0.7. When adopting the Con-HCRFs as a post-processing module, we set the model parameter α_1 to 0.5, α_2 to 0.5, and β to 5. When jointly optimizing the Con-HCRFs with our PAFE, the above model parameters in Con-HCRFs are trainable.

During training, we first train our PAFE with a multi-scale loss function used in [8]. The learning rate starts from 0.001 and is reduced by half at every 80 epochs. After 400 epochs, we add the Con-HCRFs to PAFE for fine-tuning. In fine-tuning, the learning rate starts from 0.0001 and is reduced by half at every 80 epochs. The number of mean-field iterations in Con-HCRFs is set to 1 for training and 3 for testing.

C. More Visualization

C.1. FlyingThings3D

We provide additional qualitative results on the FlyingThings3D [4] dataset in Fig. 1. The supervoxel segmentation results are computed by [3].

C.2. KITTI

We provide additional qualitative results on the KITTI Scene Flow 2015 [6, 5] dataset in Fig. 2.

References

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 1

[2] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. 1

[3] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS journal of photogrammetry and remote sensing*, 143:39–47, 2018. 2, 3

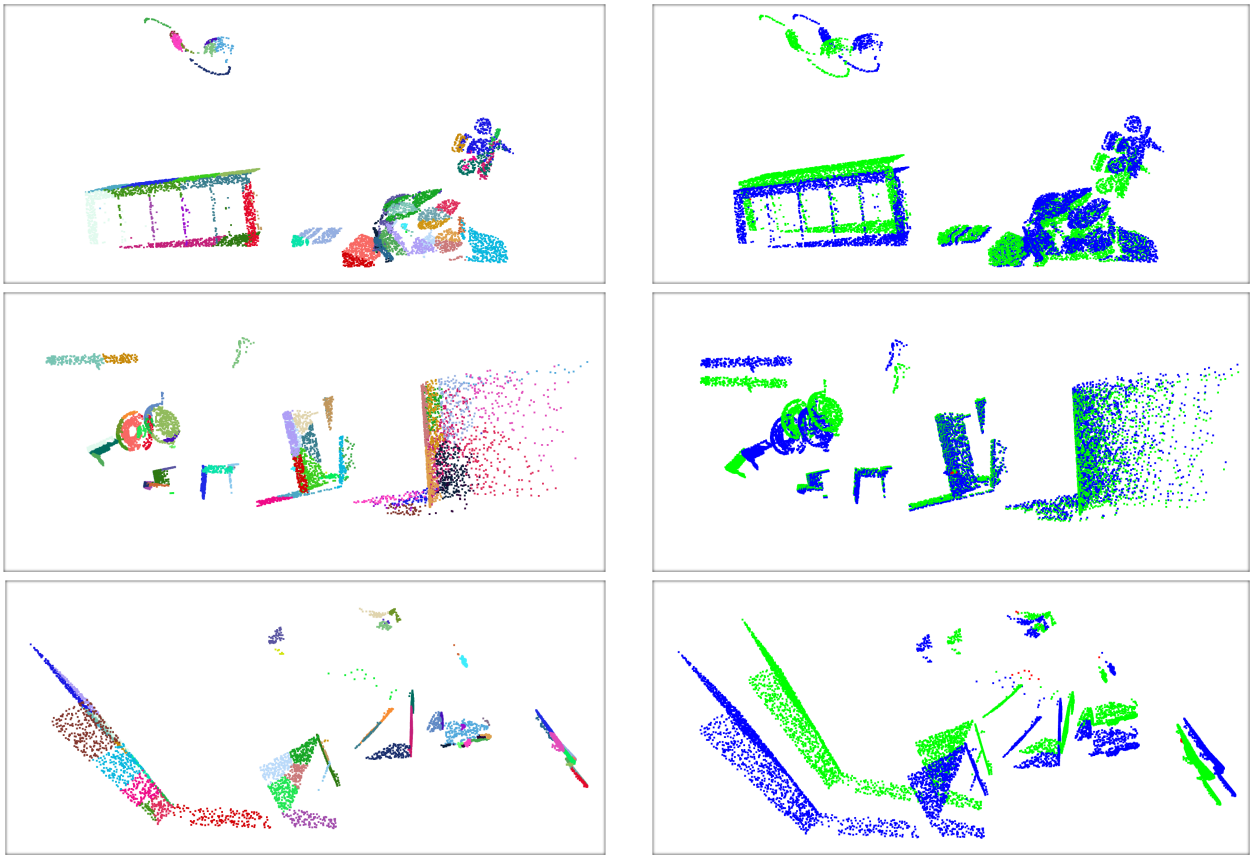
[4] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 3

[5] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2, 2015. 3

[6] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 3

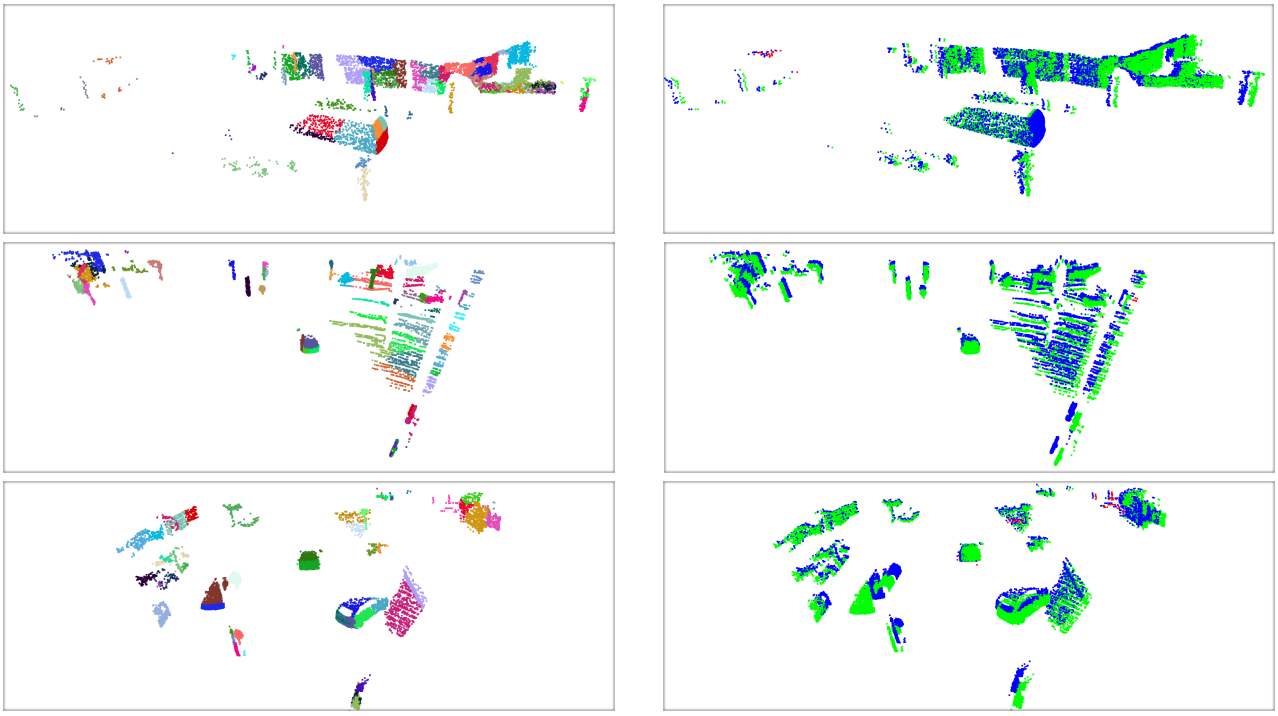
[7] Kosta Ristovski, Vladan Radosavljevic, Slobodan Vucetic, and Zoran Obradovic. Continuous conditional random fields for efficient regression in large fully connected graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013. 1, 2

[8] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision*, pages 88–107. Springer, 2020. 3



(a) Supervoxel segmentation results (b) Scene flow estimation results

Figure 1. Qualitative results on FlyingThings3D. (a) Supervoxel segmentation results. Different colors represent different supervoxel regions. (b) Scene flow estimation results. Blue points are point cloud at frame t. Green points are the warped results at frame $t+1$ for the points, whose predicted displacements are measured as correct by Acc3DR. For the incorrect predictions, we use the ground-truth scene flow to replace them. And the ground truth warped results are shown as red points.



(a) Supervoxel segmentation results (b) Scene flow estimation results

Figure 2. Qualitative results on KITTI. (a) Supervoxel segmentation results. Different colors represent different supervoxel regions. (b) Scene flow estimation results. Blue points are point cloud at frame t. Green points are the warped results at frame $t + 1$ for the points, whose predicted displacements are measured as correct by Acc3DR. For the incorrect predictions, we use the ground-truth scene flow to replace them. And the ground truth warped results are shown as red points.