Supplementary material of FLOT: Scene Flow on Point Clouds guided by Optimal Transport

Gilles Puy[1], Alexandre Boulch[1], and Renaud Marlet[1,2]

[1] valeo.ai, Paris, France
{gilles.puy,alexandre.boulch,renaud.marlet}@valeo.com
[2] ENPC, Paris, France

1 Networks architecture

The convolutions used in g and h are based on PointNet++ [3] in our implementation. Each convolution layer takes as inputs the point cloud $\boldsymbol{r} \in \mathbb{R}^{n\times 3}$ on which the convolution are performed and the features $\boldsymbol{\phi}_i^{(\ell)} \in \mathbb{R}^{c'}$, $i = 1, \ldots, n$, coming from the previous layer ℓ. Note that these features are simply the point coordinates $\boldsymbol{r}$ at the input of g and the estimated flow $\tilde{\boldsymbol{f}}$ at the input of h. For each point $\boldsymbol{r}_i$, the indices $\mathcal{N}(\boldsymbol{r}_i)$ of the $m = 32$ nearest neighbors to $\boldsymbol{r}_i$ in $\boldsymbol{r}$ are then computed to obtain m features at point $\boldsymbol{r}_i$, each one satisfying

$$\left(\boldsymbol{\phi}_j^{(\ell)\intercal},\ \boldsymbol{r}_j^\intercal - \boldsymbol{r}_i^\intercal\right)^\intercal \in \mathbb{R}^{c'+3}, \tag{1}$$

$j \in \mathcal{N}(\boldsymbol{r}_i)$. These features are passed through a MLP : $\mathbb{R}^{c'+3} \to \mathbb{R}^{c''}$ consisting of a series of fully connected layer, instance normalisation layer with affine correction [4], and leaky ReLu with a negative slope of 0.1, repeated 3 times in the same order. Finally, the new feature at point $\boldsymbol{r}_i$ is obtained after passing through a final max pooling layer:

$$\boldsymbol{\phi}_i^{(\ell+1)} = \max_{j\in\mathcal{N}(\boldsymbol{p_i})} \left\{ \mathrm{MLP}\left[(\boldsymbol{\phi}_j^{(\ell)\intercal},\ \boldsymbol{r}_j^\intercal - \boldsymbol{r}_i^\intercal)^\intercal \right] \right\} \in \mathbb{R}^{c''}, \tag{2}$$

where the max is computed independently for each of the c'' channels. These computations are repeated for each point $\boldsymbol{r}_i$ of the point cloud using the same MLP. The networks g and h share the same architecture, which is given in Table 1. Note nevertheless that the weights are not shared between g and h.

2 Datasets

The datasets $\mathrm{FT3D_s}$ and $\mathrm{KITTI_s}$ are prepared[3] as in [1]. No occluded point remains in the processed point clouds: one can always find a point $\boldsymbol{q}_j$ in $\boldsymbol{q}$ such that $\boldsymbol{q}_j = \boldsymbol{p}_i + \boldsymbol{f}_i$ at full sampling rate N. However, in practice, most of the points

[3] Code available at https://github.com/laoreja/HPLFlowNet.

Table 1. Architecture of g and h where layer $4^{(*)}$ is linear and used in h only.

Layer ℓ	1	2	3	$4^{(*)}$
MLP size	32 - 32 - 32	64 - 64 - 64	128 - 128 - 128	3

$\boldsymbol{p}_i$ do not have a direct matching in $\boldsymbol{q}$ as both point clouds are randomly and independently sub-sampled to keep only $n \ll N$ points. This simulates different sampling of the scene. Nevertheless, no object appears or disappears because of occlusions between t and $t+1$. $\mathrm{FT3D_s}$ contains $19,640$ training examples, from which we keep $2,000$ aside for validation, and $3,824$ test examples. $\mathrm{KITTI_s}$ contains 200 examples for which 142 are used for test, as in [1]. We do not use the remaining KITTI examples. The ground points in $\mathrm{KITTI_s}$ are removed using a threshold on the height. All points whose depth is larger than 35 m are removed in both datasets.

The datasets $\mathrm{FT3D_o}$ and $\mathrm{KITTI_o}$ are the prepared[4] by [2]. In $\mathrm{FT3D_o}$, masks where the flow is non valid, *e.g.*, due to occlusions, are provided in used in the training loss, like in [2]. These masks are also used to compute the scores only on valid points at test time for all methods. However, the points where the corresponding flow is non-valid are present at the input of all networks. No mask is provided for $\mathrm{KITTI_o}$. $\mathrm{FT3D_o}$ contains $19,999$ training examples, from which we keep $2,000$ aside for validation, and $2,003$ test examples.[5] $\mathrm{KITTI_o}$ contains 150 test examples. The ground points in $\mathrm{KITTI_o}$ are removed by [2]. All points whose depth is larger than 35 m are removed in both datasets.

3 Performance metrics

We use the following four metrics adopted in [1], [2], [5]:

- $\mathrm{EPE}_i = \|(\boldsymbol{f}_{\mathbf{est}})_i - \boldsymbol{f}_i\|_2$: end point error, averaged over all i;
- AS: percentage of points such that $\mathrm{EPE}_i < 0.05$ or $\mathrm{EPE}_i / \|\boldsymbol{f}_i\|_2 < 0.05$;
- AR: percentage of points such that $\mathrm{EPE}_i < 0.1$ or $\mathrm{EPE}_i / \|\boldsymbol{f}_i\|_2 < 0.1$
- Out.: percentage of points such that $\mathrm{EPE}_i > 0.3$ or $\mathrm{EPE}_i / \|\boldsymbol{f}_i\|_2 > 0.1$.

The above metrics are computed as follows. The point clouds $\boldsymbol{p}, \boldsymbol{q}$ are obtained by selecting n random points out of the N provided points in the datasets. The flow is estimated and compared to the ground truth flow $\boldsymbol{f}$ on these n selected points. The scores are averaged over the whole validation/test set.

[4] Datasets available at https://github.com/xingyul/flownet3d.

[5] We removed 8 examples with all points marked as occluded (7 in the training set and 4 in the test set). One example which contains a non valid value in the training dataset is also removed.

Table 2. Performance of FLOT measured at the output of the OT module, *i.e.*, before refinement by h, on $FT3D_o$. We report the average scores and their standard deviations between parentheses.

Dataset	K	EPE	AS	AR	Out.
$FT3D_o$	$FLOT_0$	0.3539 (0.0028)	6.98 (0.11)	22.05 (0.28)	88.76 (0.14)
	1	0.3412 (0.0042)	7.55 (0.17)	23.50 (0.40)	88.02 (0.22)
	3	0.3426 (0.0028)	7.38 (0.04)	23.09 (0.05)	88.21 (0.03)
	5	0.3440 (0.0021)	7.32 (0.05)	22.94 (0.16)	88.34 (0.09)

4 Additional experimental results

4.1 Study of FLOT

We report in Table 2 the performance of FLOT obtained at the output of the OT module on $FT3D_o$. The corresponding performance with refinement are available in the core of the paper. As on $FT3D_s$, we remark that the refinement permits to improve the EPE by around 2, confirming its utility in presence of occlusions.

4.2 Computation time in the OT module

At $n = 2048$, the computation time[6] in the OT module is 1.4, 2.0 and 2.2 ms for $FLOT_0$, FLOT $K = 1$, FLOT $K = 3$, respectively. At $n = 8192$, the computation time in the OT module is 13.1, 16.0, 17.9 ms for $FLOT_0$, FLOT $K = 1$, FLOT $K = 3$, respectively. This represents at most 8% of the total computation time which is itself at most of 27.8 ms at $n = 2048$ and 346 ms at $n = 8192$. Most of the time, at least 67% at $n = 2048$ and 86% at $n = 8192$, is spent in the feature extractor g. This shows that the OT module is responsible for just a small fraction of the total computation time.

Note that the time spent in the OT module is independent of the type of convolution used. Replacing our implementation of PointNet++ with a faster one or choosing a faster convolution will directly improve the computation time spend in g and h. Our implementation of the OT module can also be made faster by avoiding to compute densely the cost matrix C by restricting the computation to points that are less than $d_{\max}$ meters apart, as these points never contribute to T.

References

1. 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. In: Conference on Computer Vision and Pattern Recognition. pp. 3249–3258. IEEE (2019) 1, 2

[6] Computed on a Nvidia GeForce RTX 2080 Ti.

2. Liu, X., Qi, C.R., Guibas, L.J.: FlowNet3D: Learning Scene Flow in 3D Point Clouds. In: Conference on Computer Vision and Pattern Recognition. pp. 529–537. IEEE (2019) 2
3. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. pp. 5099–5108. Curran Associates, Inc. (2017) 1
4. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance Normalization: The Missing Ingredient for Fast Stylization. arXiv:1607.08022 (2016) 1
5. Wu, W., Wang, Z., Li, Z., Liu, W., Fuxin, L.: PointPWC-Net: A Coarse-to-Fine Network for Supervised and Self-Supervised Scene Flow Estimation on 3D Point Clouds. arXiv:1911.12408v1 (2019) 2