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 $r \in \mathbb{R}^{n \times 3}$ on which the convolution are performed and the features $\phi_i^{(\ell)} \in \mathbb{R}^{c'}$, $i = 1, \ldots, n$, coming from the previous layer $\ell$. Note that these features are simply the point coordinates $r$ at the input of $g$ and the estimated flow $\tilde{f}$ at the input of $h$. For each point $r_i$, the indices $\mathcal{N}(r_i)$ of the $m = 32$ nearest neighbors to $r_i$ in $r$ are then computed to obtain $m$ features at point $r_i$, each one satisfying

$$
\left(\phi_j^{(\ell)^T}, r_j^T - r_i^T\right)^T \in \mathbb{R}^{c'+3},
$$

$j \in \mathcal{N}(r_i)$. These features are passed through a MLP : $\mathbb{R}^{c'+3} \rightarrow \mathbb{R}^{c''}$ 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 $r_i$ is obtained after passing through a final max pooling layer:

$$
\phi_i^{(\ell+1)} = \max_{j \in \mathcal{N}(p_i)} \left\{ \text{MLP} \left( \left(\phi_j^{(\ell)^T}, r_j^T - r_i^T\right)^T \right) \right\} \in \mathbb{R}^{c''},
$$

where the max is computed independently for each of the $c''$ channels. These computations are repeated for each point $r_i$ 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 FT3D$_s$ and KITTI$_s$ are prepared$^3$ as in [1]. No occluded point remains in the processed point clouds: one can always find a point $q_j$ in $q$ such that $q_j = p_i + f_i$ at full sampling rate $N$. However, in practice, most of the points

---

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

<table>
<thead>
<tr>
<th>Layer $\ell$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4^{(*)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP size</td>
<td>32 - 32</td>
<td>64 - 64</td>
<td>128 - 128</td>
<td>3</td>
</tr>
</tbody>
</table>

$p_i$ do not have a direct matching in $q$ 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 $t + 1$. FT3D$_o$ contains 19,640 training examples, from which we keep 2,000 aside for validation, and 3,824 test examples. KITTI$_o$ 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 KITTI$_o$ are removed using a threshold on the height. All points whose depth is larger than 35 m are removed in both datasets.

The datasets FT3D$_o$ and KITTI$_o$ are the prepared by [2]. In FT3D$_o$, masks where the flow is non valid, e.g., due to occlusions, are provided 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 KITTI$_o$. FT3D$_o$ contains 19,999 training examples, from which we keep 2,000 aside for validation, and 2,003 test examples. KITTI$_o$ contains 150 test examples. The ground points in 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]:

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

The above metrics are computed as follows. The point clouds $p, q$ 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 $f$ 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>K</th>
<th>EPE</th>
<th>AS</th>
<th>AR</th>
<th>Out.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT3D$_o$</td>
<td>FLOT$_0$</td>
<td>0.3539 (0.0028)</td>
<td>6.98 (0.11)</td>
<td>22.05 (0.28)</td>
<td>88.76 (0.14)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.3412 (0.0042)</td>
<td>7.55 (0.17)</td>
<td>23.50 (0.40)</td>
<td>88.02 (0.22)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.3426 (0.0028)</td>
<td>7.38 (0.04)</td>
<td>23.09 (0.05)</td>
<td>88.21 (0.03)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.3440 (0.0021)</td>
<td>7.32 (0.05)</td>
<td>22.94 (0.16)</td>
<td>88.34 (0.09)</td>
</tr>
</tbody>
</table>

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 $d_{\text{max}}$ meters apart, as these points never contribute to $T$.

References


$^6$ Computed on a Nvidia GeForce RTX 2080 Ti.

