Supplementary Material

In this supplementary material, we first provide the definitions for the symbols used in the supplementary material in Section 1. Then, we walk through the essential background material for optical flow estimation in Section 2 since it is one of the feature representations adopted in PWVO. Next, in Section 3, we provide the link to the source codes of PWVO, and elaborate on the formulations of the loss functions for depth and flow estimation, as well as the hyperparameters adopted by PWVO. In Section 4, we explain the configurations of the data generation workflow. Finally, in Section 5, we present additional ablation analyses and more qualitative results to show the effectiveness of PWVO.

1. List of Notations

In this section, we provide the list of notations used throughout the supplementary material. The symbols and their descriptions are summarized and explained in Table 1.

2. Background Material

2.1. Optical Flow Estimation

Optical flow estimation is an application domain for evaluating displacements of pixels between consecutive image frames, and is one of the most crucial areas that have attracted the attention of computer vision researchers for many years [1, 13]. Due to the rapid advances of deep neural networks (DNNs), there have been a number of DNN-based optical flow estimation methods developed in the past few years. FlowNet [1] was the first convolutional neural network (CNN) based approach that estimates optical flow maps in an end-to-end fashion. FlowNet2 [2] adopted a stacked architecture for iteratively refining optical flow maps. In addition, the authors in [3, 4, 8, 11] further introduced coarse-to-fine pyramid architectures to apply iterative refinement. RAFT [10] proposed to build multi-scale four dimensional correlation volumes, and utilize them to iteratively update residual flow fields through a gated recurrent unit (GRU). The authors in [12, 13] employed transformer based architectures [14] for estimating optical flow maps. Among them, the method proposed in [12] was the first one that utilizes transformers for flow estimation. On the other hand, the method proposed in [13] built their transformer architecture on top of RAFT [10], and achieved state-of-the-art (SOTA) performance. In order to train effective models capable of correctly estimating optical flow maps, these methods typically rely on synthetic datasets [15–18] to offer ground truth annotations to train the models in supervised manners.

3. Implementation Details

The results presented in this paper are fully reproducible, and the source codes can be available at: https://anonymous.4open.science/r/PWVO-5483.

3.1. Loss Function

In this section, we explain the formulations of the two loss functions \( \hat{L}_i^D \) and \( \hat{L}_i^F \) in detail. In addition to \( \hat{L}_i^D \) and \( \hat{L}_i^F \), PWVO is further optimized by \( \hat{L}_i^{F_x} \) and \( \hat{L}_i^{F_y} \), through the use of three different uncertainty maps \( \hat{U}_i^D \), \( \hat{U}_i^{F_x} \), and \( \hat{U}_i^{F_y} \). The loss terms can be formulated as follows:

\[
\hat{L}_i^D = \frac{1}{N} \sum_{p=1}^{N} \sum_{j=1}^{3} \left[ \frac{E^D(D_{p+1}^D, \tilde{D}_{p+1}^D)}{U_i^D(p) \cdot \lambda^D} + \log(\hat{U}_i^D(p) \cdot \lambda^D) \right],
\]

(1)

\[
\hat{L}_i^{F_x} = \frac{1}{N} \sum_{p=1}^{N} \sum_{j=1}^{5} M_i^p \cdot \left[ \frac{E^F(F_{x_i}^{ego}(p), \tilde{F}_{x_i}^{ego}(p))}{\hat{U}_i^{F_x}(p) \cdot \lambda^F} + \log(\hat{U}_i^{F_x}(p) \cdot \lambda^F) \right],
\]

(2)

\[
\hat{L}_i^{F_y} = \frac{1}{N} \sum_{p=1}^{N} \sum_{j=1}^{5} M_i^p \cdot \left[ \frac{E^F(F_{y_i}^{ego}(p), \tilde{F}_{y_i}^{ego}(p))}{\hat{U}_i^{F_y}(p) \cdot \lambda^F} + \log(\hat{U}_i^{F_y}(p) \cdot \lambda^F) \right],
\]

(3)

\[
M_i^p = \begin{cases} 
1 & \text{if } E^F(x,y) < \delta^F \ 
0 & \text{otherwise} 
\end{cases}
\]

(4)

where \( E^D(x,y) = \| \frac{x+y}{2} - \frac{x+y}{2} \| \), \( E^F(x,y) = \| x-y \| \), \( j \) stands for the channel index of an uncertainty map, \( \lambda^D \) and \( \lambda^F \) represent the scaling factors, \( E^D \) and \( E^F \) denote...
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H$</td>
<td>Height</td>
</tr>
<tr>
<td>$W$</td>
<td>Width</td>
</tr>
<tr>
<td>$k$</td>
<td>Patch size</td>
</tr>
<tr>
<td>$K$</td>
<td>Camera intrinsic matrix</td>
</tr>
<tr>
<td>$\hat{L}^R$</td>
<td>Rotation loss of the camera in terms of angles (with uncertainty map)</td>
</tr>
<tr>
<td>$\hat{L}^T$</td>
<td>Translation loss of the camera (with uncertainty map)</td>
</tr>
<tr>
<td>$\hat{L}^F = \hat{L}^{Fx} + \hat{L}^{Fy}$</td>
<td>Ego flow loss (with uncertainty map)</td>
</tr>
<tr>
<td>$\hat{L}^D$</td>
<td>Depth loss with (uncertainty map)</td>
</tr>
<tr>
<td>$\tilde{U}$</td>
<td>Uncertainty map used in $\hat{L}^R$, where $\tilde{U}^R \in \mathbb{R}^{H \times W \times 3}$</td>
</tr>
<tr>
<td>$\tilde{U}^T$</td>
<td>Uncertainty map used in $\hat{L}^T$, where $\tilde{U}^T \in \mathbb{R}^{H \times W \times 3}$</td>
</tr>
<tr>
<td>$\tilde{U}^F = [\tilde{U}^{Fx}, \tilde{U}^{Fy}]$</td>
<td>Uncertainty map used in $\hat{L}^{Fx} + \hat{L}^{Fy}$, where $\tilde{U}^{Fx}, \tilde{U}^{Fy} \in \mathbb{R}^{H \times W \times 5}$</td>
</tr>
<tr>
<td>$\lambda^D$</td>
<td>Scaling factor used in $\hat{L}^D$, where $\lambda^D \in \mathbb{R}^{H \times W \times 3}$</td>
</tr>
<tr>
<td>$\lambda^F$</td>
<td>Scaling factor used in $\hat{L}^F$</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>The mixtures of two distributions for parameter sampling (Eq. (11))</td>
</tr>
<tr>
<td>$\eta$</td>
<td>The first distribution in $\zeta$</td>
</tr>
<tr>
<td>$\tau$</td>
<td>The second distribution in $\zeta$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>The mean for the two distributions $\eta$ and $\tau$</td>
</tr>
<tr>
<td>$\sigma_1$</td>
<td>The standard deviation of $\eta$</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>The standard deviation of $\tau$</td>
</tr>
<tr>
<td>$g$</td>
<td>The power of $\eta$ (defined in Eq. (11))</td>
</tr>
<tr>
<td>$\rho$</td>
<td>The probability of sampling $\eta$ in $\zeta$</td>
</tr>
<tr>
<td>$c, d$</td>
<td>The lower and upper bounds of $\eta$ (specified in Eq. (11))</td>
</tr>
<tr>
<td>$f, \bar{f}$</td>
<td>The focal length and the normalized focal length in $K$</td>
</tr>
<tr>
<td>$D_{bg}, D_{obj}$</td>
<td>Depth map and normalized depth map for the background</td>
</tr>
<tr>
<td>$D_{obj}$</td>
<td>Depth map and normalized depth map of an object</td>
</tr>
<tr>
<td>$\gamma = [R_x, R_y, R_z]$</td>
<td>Camera rotation angle in x, y, and z axes</td>
</tr>
<tr>
<td>$\varphi = [T_x, T_y, T_z]$</td>
<td>Camera translation offset in x, y, and z axes</td>
</tr>
</tbody>
</table>

The distance functions used for calculating the differences between the ground truth labels and the predictions, and $\varepsilon$ is set to $1e-12$ for $\mathcal{L}^D$. Please note that the compositions of $\tilde{U}^R_i, \tilde{U}^T_i, \tilde{U}^D_i, \tilde{U}^F_i$ are different, and can be represented as the following expressions:

$$
\tilde{U}^R_i = [\tilde{U}^{Rx}_i, \tilde{U}^{Ry}_i, \tilde{U}^{Rz}_i],
$$

$$
\tilde{U}^T_i = [\tilde{U}^{Tx}_i, \tilde{U}^{Ty}_i, \tilde{U}^{Tz}_i],
$$

$$
\tilde{U}^D_i = [\tilde{U}^{Dx}_i, \tilde{U}^{Dy}_i, \tilde{U}^{Dz}_i],
$$

$$
\tilde{U}^F_i = [\tilde{U}^{Fx}_i, \tilde{U}^{Fy}_i, \tilde{U}^{Fx}_i, \tilde{U}^{Fy}_i, \tilde{U}^{Fx}_i, \tilde{U}^{Fy}_i, \tilde{U}^{Fx}_i, \tilde{U}^{Fy}_i],
$$

where $\tilde{U}^{Rx}_i, \tilde{U}^{Ry}_i, \tilde{U}^{Rz}_i$ and $\tilde{U}^{Tx}_i, \tilde{U}^{Ty}_i, \tilde{U}^{Tz}_i$ can be obtained from $\tilde{U}^R_i$ and $\tilde{U}^T_i$, respectively. Please note that the composition of each uncertainty map is different from each other due to the following reasons: (1) only the rotation of the x and y axes, as well as the translation of the z axis affects the depth map; (2) the rotation of the x, y, and z axes, and the translation of the x and z axes affect $\mathcal{L}^D$; and (3) the...
rotation of the x, y, and z axes, and the translation of the y and z axes affect \( F_y \). In practice, Eqs. (8) - (10) are modified to predict log variance to stabilize the training progress and avoid errors resulted from division by zero. These equations are reformulated as the following:

\[
\hat{L}_i^{D} = \frac{1}{N} \sum_{p=1}^{N} \sum_{j=1}^{3} \exp \left( \hat{s}_i^{Dj} (p) \right) \cdot \mathcal{E}^{D} (D_{i+1}, D_{i+1}^{Dj}) + \hat{s}_i^{Dj} (p),
\]

(8)

\[
\hat{L}_i^{F_x} = \frac{1}{N} \sum_{p=1}^{N} \sum_{j=1}^{5} \mathcal{M}_i^{D} \cdot \left[ \hat{s}_i^{F_xj} (p) \right] + \exp \left( \hat{s}_i^{F_xj} (p) \cdot \mathcal{E}^{F} (F_{x_i}^{ego} (p), \tilde{F}_{x_i}^{ego} (p)) \right),
\]

(9)

\[
\hat{L}_i^{F_y} = \frac{1}{N} \sum_{p=1}^{N} \sum_{j=1}^{5} \mathcal{M}_i^{D} \cdot \left[ \hat{s}_i^{F_yj} (p) \right] + \exp \left( \hat{s}_i^{F_yj} (p) \cdot \mathcal{E}^{F} (F_{y_i}^{ego} (p), \tilde{F}_{y_i}^{ego} (p)) \right).
\]

(10)

where \( \hat{s}_i^{Dj} (p) = \log (\hat{\mathcal{U}}_i^{Dj} (p) \cdot \lambda^{D}) \), \( \hat{s}_i^{F_xj} (p) = \log (\hat{\mathcal{U}}_i^{F_xj} (p) \cdot \lambda^{F}) \), \( \hat{s}_i^{F_yj} (p) = \log (\hat{\mathcal{U}}_i^{F_yj} (p) \cdot \lambda^{F}) \).

### 3.2. Hyperparameters

PWVO is implemented using Tensorflow 2.0. All convolutional modules are initialized from scratch with random weights by using Glorot Normal Initializer. In addition, we use the Adam optimizer and clip the gradients norm to the range \([-1, 1]\). Then, we train and evaluate PWVO on an NVIDIA RTX 3090 GPU. The hyperparameters employed for training our model are presented in Table 2.

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>100</td>
</tr>
<tr>
<td>Learning rate</td>
<td>1e-4</td>
</tr>
<tr>
<td>Numbers of epochs</td>
<td>100</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Mean for normalizing F&lt;sub&gt;total&lt;/sub&gt;</td>
<td>0</td>
</tr>
<tr>
<td>Standard deviation for normalizing input F&lt;sub&gt;total&lt;/sub&gt;</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{k} )</td>
<td>32</td>
</tr>
<tr>
<td>( \lambda^D )</td>
<td>0.25</td>
</tr>
<tr>
<td>( \lambda^F )</td>
<td>0.1</td>
</tr>
<tr>
<td>( \delta^F )</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 2: The hyperparameters for training PWVO.

### 4. The Configurations of the Data Generation Workflow

As discussed in Section 4 of the main manuscript, the proposed data generation workflow allows a wide range of synthetic data to be generated. In order to comprehensively train the proposed PWVO model, a number of hyperparameters are sampled from distributions for generating training data. These hyperparameters and the parameters for configuring their distributions are summarized in Table 3. Similar to the approach adopted in [11], the distributions \( \zeta \) are defined as the following:

\[
\zeta = \beta \cdot \max (\min (\text{sign}(\eta) \cdot |\eta|^\rho, c), d) + (1 - \beta) \cdot \tau,
\]

(11)

where \( \beta \) is a Bernoulli random variable. It takes on a one with probability \( \rho \), and a zero with probability \( 1 - \rho \).

According to Eq. (11), the distributions \( \zeta \) used for sampling our hyperparameters consists of two parts: the former part involves a power of Gaussian distribution \( \eta \sim \mathcal{N}(\mu, \sigma_1^2) \), while the latter part also contains a Gaussian distribution \( \tau \sim \mathcal{N}(\mu, \sigma_2^2) \) but with a different variance.

#### Focal length \( f \)

With \( \hat{f} \) sampled from the distribution defined above, \( f \) can be obtained as follows:

\[
f = \hat{f} \cdot (f_{\max} - f_{\min}) + f_{\min},
\]

(12)

where \([f_{\min}, f_{\max}]\) are set to \([576, 3200]\) in the proposed data generation workflow.

#### Background depth \( D_{bg} \) and object depth \( D_{obj} \)

The background depth can be obtained by:

\[
D_{bg} = \beta_{bg} \cdot (D_{bg} \cdot (d_2 - d_1 + d_3) + (1 - \beta_{bg}) \cdot h(x, d_2, d_3)
\]

(13)

where \( \beta_{bg} \) is a Bernoulli random variables that takes on a 1 with probability 0.9 and a 0 with probability 0.1, \((d_1, d_2, d_3)\) are set to \((1, 80, 3200)\) in our workflow, and the
probability density function $h$ of the uniform distribution is defined as follows:

$$h(x, a, b) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{for } x > a \text{ or } x > b \end{cases}$$ (14)

where $h(x, a, b)$ is formulated as a probability density function of uniform distribution with the minimum and the maximum equal to $a$ and $b$, respectively. The object depth $D_{obj}$ can then be derived as the following equation:

$$D_{obj} = \bar{D}_{obj} \cdot (D_{bg} - D_{min}) + D_{min} = \frac{f}{f_{ratio}}$$ (15)

where $f_{ratio}$ is set to 5, 200 in our data generation workflow.

**Rotation $R$ and translation $T$.** The hyperparameters for rotation and translation listed in Table 3 are sampled from their corresponding distributions, which are defined in Eq. (11).

### 5. Additional Experimental Results

#### 5.1. The Impact of Object Noises on the Performance of VO

In this section, we ablativey examine the impact of the noises induced by moving objects on the performance of VO. To this end, we consider two configurations of synthetic datasets in our ablation analysis: (1) data samples involve the motions of the camera and moving objects, and (2) only the camera may move. The two configurations are denoted by subscripts total and ego, respectively. We train the VONet baseline, PWVO (naive), and PWVO by the training procedure described in the main manuscript, and then inspect their performance on the evaluation datasets under these two configurations. The results are presented in Table 4. It can be observed that all the methods deliver better evaluation results on Sintel_ego than on Sintel_total, validating our hypothesis that the noises from moving objects do affect the performance of VO. In addition, it can also be observed that in general, the methods trained on Custom_total exhibit better generalizability on Sintel_total than those trained on Custom_ego, as the latter does not involve any moving object during the training.

#### 5.2. The Influence of the Patch Size $k$ on the Performance of PWVO

In this section, we analyze the impact of the patch size $k$ on the performance of PWVO. As discuss in Section 3.4.2 of the main manuscript, the selection module derives $(\tilde{\gamma}_i, \tilde{\varphi}_i)$ from $(\bar{R}_i, \bar{T}_i)$ and $(\bar{U}_i^R(p), \bar{U}_i^T(p))$ through the use of patches of $k \times k$ pixels. This mechanism enables PWVO to suppress the influence of noisy regions that contain moving objects. In this experiment, we compare the performances of PWVO with different values of $k$, which corresponds to different fields of view. The results are summarized in Table 5. It can be observed that the choice of $k$ is crucial to the performance of PWVO. When the path size is equal to $H \times W$, PWVO simply selects the pixel coordinate with the lowest uncertainty for its final prediction, without taking into account the remaining regions in its input observations. In contrast, when $k$ is set to one, PWVO generates its final $(\tilde{\gamma}, \tilde{\varphi})$ by consideration the pixel-wise predictions from all the pixel coordinates. The results in Table 5 suggest that when $k = 16$, PWVO is able to deliver the best performance on the test set of Sintel. Based on the observation, this configuration is adopted for all the experiments in this work.

### 5.3. Additional Qualitative Results

In this section, we provide additional qualitative results of PWVO, which are evaluated on the Sintel [15] in Fig. 1.

### References

Figure 1: The qualitative results of PWVO evaluated on the Sintel dataset.


