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 neutal 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 dimensiontal 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 $\hat{\mathcal{L}}_i^{\mathcal{F}}$ and $\hat{\mathcal{L}}_i^{\mathcal{D}}$ in detail. In addition to $\hat{\mathcal{L}}_i^{\mathcal{R}}$ and $\hat{\mathcal{L}}_i^{\mathcal{T}}$, PWVO is further optimized by $\hat{\mathcal{L}}_i^{\mathcal{D}}$ and $\hat{\mathcal{L}}_i^{\mathcal{F}} = \hat{\mathcal{L}}_i^{\mathcal{F}_x} + \hat{\mathcal{L}}_i^{\mathcal{F}_y}$, through the use of three different uncertainty maps $\tilde{\mathcal{U}}^{\mathcal{D}}$, $\tilde{\mathcal{U}}^{\mathcal{F}_x}$, and $\tilde{\mathcal{U}}^{\mathcal{F}_y}$. The loss terms can be formulated as follows:

$$\hat{\mathcal{L}}_i^{\mathcal{D}} = \frac{1}{\mathbf{N}} \sum_{p=1}^{\mathbf{N}} \sum_{j=1}^{3} [\frac{\mathcal{E}^{\mathcal{D}}(\mathbf{D}_{i+1}^p, \tilde{\mathbf{D}}_{i+1}^p)}{\mathcal{U}_i^{\tilde{\mathcal{D}}_j}(p) \cdot \lambda^{\mathcal{D}}} + \log(\mathcal{U}_i^{\tilde{\mathcal{D}}_j}(p) \cdot \lambda^{\mathcal{D}})], \tag{1}$$

$$\hat{\mathcal{L}}_i^{\mathcal{F}_x} = \frac{1}{\mathbf{N}} \sum_{p=1}^{\mathbf{N}} \sum_{j=1}^{5} \mathcal{M}_i^p \cdot [\frac{\mathcal{E}^{\mathcal{F}}(\mathbf{F_x}_i^{ego}(p), \tilde{\mathbf{F_x}}_i^{ego}(p))}{\tilde{\mathcal{U}}_i^{\mathcal{F}_x^j}(p) \cdot \lambda^{\mathcal{F}}} + \log(\tilde{\mathcal{U}}_i^{\mathcal{F}_x^j}(p) \cdot \lambda^{\mathcal{F}})], \tag{2}$$

$$\hat{\mathcal{L}}_i^{\mathcal{F}_y} = \frac{1}{\mathbf{N}} \sum_{p=1}^{\mathbf{N}} \sum_{j=1}^{5} \mathcal{M}_i^p \cdot [\frac{\mathcal{E}^{\mathcal{F}}(\mathbf{F_y}_i^{ego}(p), \tilde{\mathbf{F_y}}_i^{ego}(p))}{\tilde{\mathcal{U}}_i^{\mathcal{F}_y^j}(p) \cdot \lambda^{\mathcal{F}}} + \log(\tilde{\mathcal{U}}_i^{\mathcal{F}_y^j}(p) \cdot \lambda^{\mathcal{F}})], \tag{3}$$

$$\mathcal{M}_i^p = \begin{cases} 1 & \text{if } \mathcal{E}^{\mathcal{F}}(x, y) < \delta^{\mathcal{F}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{E}^{\mathcal{D}}(x, y) = \| \frac{1}{x+\epsilon} - \frac{1}{y+\epsilon} \|$, $\mathcal{E}^{\mathcal{F}}(x, y) = \|x - y\|$, j stands for the channel index of an uncertainty map, $\lambda^{\mathcal{D}}$ and $\lambda^{\mathcal{F}}$ represent the scaling factors, $\mathcal{E}^{\mathcal{D}}$ and $\mathcal{E}^{\mathcal{F}}$ denote

Table 1: The list of notations used in the supplementary material.

Symbol	Description
H	Height
W	Width
k	Patch size
$\mathbf{K}$	Camera intrinsic matrix
$\hat{\mathcal{L}}^{\mathcal{R}}$	Rotation loss of the camera in terms of angles (with uncertainty map)
$\hat{\mathcal{L}}^{\mathcal{T}}$	Translation loss of the camera (with uncertainty map)
$\hat{\mathcal{L}}^{\mathcal{F}} = \hat{\mathcal{L}}^{\mathcal{F}_x} + \hat{\mathcal{L}}^{\mathcal{F}_y}$	Ego flow loss (with uncertainty map)
$\hat{\mathcal{L}}^{\mathcal{D}}$	Depth loss with (uncertainty map)
$\tilde{\mathcal{U}}^{\mathcal{R}}$	Uncertainty map used in $\hat{\mathcal{L}}^{\mathcal{R}}$, where $\tilde{\mathcal{U}}^{\mathcal{R}} \in \mathbb{R}^{H \times W \times 3}$
$\tilde{\mathcal{U}}^{\mathcal{T}}$	Uncertainty map used in $\hat{\mathcal{L}}^{\mathcal{T}}$, where $\tilde{\mathcal{U}}^{\mathcal{T}} \in \mathbb{R}^{H \times W \times 3}$
$\tilde{\mathcal{U}}^{\mathcal{F}} = [\tilde{\mathcal{U}}^{\mathcal{F}_x}, \tilde{\mathcal{U}}^{\mathcal{F}_y}]$	Uncertainty map used in $\hat{\mathcal{L}}^{\mathcal{F}_x} + \hat{\mathcal{L}}^{\mathcal{F}_y}$, where $\tilde{\mathcal{U}}^{\mathcal{F}_x}, \tilde{\mathcal{U}}^{\mathcal{F}_y} \in \mathbb{R}^{H \times W \times 5}$
$\tilde{\mathcal{U}}^{\mathcal{D}}$	Uncertainty map used in $\hat{\mathcal{L}}^{\mathcal{D}}$, where $\tilde{\mathcal{U}}^{\mathcal{D}} \in \mathbb{R}^{H \times W \times 3}$
$\lambda^{\mathcal{D}}$	Scaling factor used in $\hat{\mathcal{L}}^{\mathcal{D}}$
$\lambda^{\mathcal{F}}$	Scaling factor used in $\hat{\mathcal{L}}^{\mathcal{F}}$
ζ	The mixtures of two distributions for parameter sampling (Eq. (11))
η	The first distribution in ζ
τ	The second distribution in ζ
μ	The mean for the two distributions η and τ
σ_1	The standard deviation of η
σ_2	The standard deviation of τ
g	The power of η (defined in Eq. (11))
ρ	The probability of sampling η in ζ
$c, \ d$	The lower and upper bounds of η (specified in Eq. (11))
$f, \bar{f}$	The focal length and the normalized focal length in $\mathbf{K}$
$\mathbf{D}_{bg}, \bar{\mathbf{D}}_{bg}$	Depth map and normalized depth map for the background
$\mathbf{D}_{obj}, \bar{\mathbf{D}}_{obj}$	Depth map and normalized depth map of an object
$\gamma = [\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z]$	Camera rotation angle in x, y, and z axes
$\varphi = [\mathbf{T}_x, \mathbf{T}_y, \mathbf{T}_z]$	Camera translation offset in x, y, and z axes

the distance functions used for calculating the differences between the ground truth labels and the predictions, and ϵ is set to $1e$-12 for $\mathcal{E}^{\mathcal{D}}$. Please note that the compositions of $\tilde{\mathcal{U}}_i^{\mathcal{D}_j}$, $\tilde{\mathcal{U}}_i^{\mathcal{F}_x^j}$, and $\tilde{\mathcal{U}}_i^{\mathcal{F}_y^j}$ are different, and can be represented as the following expressions:

$$\tilde{\mathcal{U}}_i^{\mathcal{D}} = [\tilde{\mathcal{U}}_i^{\mathcal{R}_x}, \tilde{\mathcal{U}}_i^{\mathcal{R}_y}, \tilde{\mathcal{U}}_i^{\mathcal{T}_z}], \tag{5}$$

$$\tilde{\mathcal{U}}_i^{\mathcal{F}_x} = [\tilde{\mathcal{U}}_i^{\mathcal{R}_x}, \tilde{\mathcal{U}}_i^{\mathcal{R}_y}, \tilde{\mathcal{U}}_i^{\mathcal{R}_z}, \tilde{\mathcal{U}}^{\mathcal{T}_x}, \tilde{\mathcal{U}}^{\mathcal{T}_z}], \tag{6}$$

$$\tilde{\mathcal{U}}_i^{\mathcal{F}_y} = [\tilde{\mathcal{U}}_i^{\mathcal{R}_x}, \tilde{\mathcal{U}}_i^{\mathcal{R}_y}, \tilde{\mathcal{U}}_i^{\mathcal{R}_z}, \tilde{\mathcal{U}}^{\mathcal{T}_y}, \tilde{\mathcal{U}}^{\mathcal{T}_z}], \tag{7}$$

where $(\tilde{\mathcal{U}}_i^{\mathcal{R}_x}, \tilde{\mathcal{U}}_i^{\mathcal{R}_y}, \tilde{\mathcal{U}}_i^{\mathcal{R}_z})$ and $(\tilde{\mathcal{U}}_i^{\mathcal{T}_x}, \tilde{\mathcal{U}}_i^{\mathcal{T}_y}, \tilde{\mathcal{U}}_i^{\mathcal{T}_z})$ can be obtained from $\tilde{\mathcal{U}}_i^{\mathcal{R}}$ and $\tilde{\mathcal{U}}_i^{\mathcal{T}}$, 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 $\mathbf{F}_x$; and (3) the rotation of the x, y, and z axes, and the translation of the y and z axes affect $\mathbf{F}_y$. In practice, Eqs.(8) - (10) are modified to predict log variance to stablize the training progress and avoid errors resulted from division by zero. These equations are reformulated as the following:

$$\hat{\mathcal{L}}_i^{\mathcal{D}} = \frac{1}{\mathbf{N}} \sum_{p=1}^{\mathbf{N}} \sum_{j=1}^{3} \exp\left(\tilde{s}_i^{\mathcal{D}_j}(p)\right) \cdot \mathcal{E}^{\mathcal{D}}(\mathbf{D}_{i+1}^p, \tilde{\mathbf{D}}_{i+1}^p) + \tilde{s}_i^{\mathcal{D}_j}(p), \tag{8}$$

$$\begin{aligned}\hat{\mathcal{L}}_i^{\mathcal{F}_x} = \frac{1}{\mathbf{N}} \sum_{p=1}^{\mathbf{N}} \sum_{j=1}^{5} \mathcal{M}_i^p \cdot [\tilde{s}_i^{\mathcal{F}_x^j}(p) \\ + \exp\left(\tilde{s}_i^{\mathcal{F}_x^j}(p) \cdot \mathcal{E}^{\mathcal{F}}(\mathbf{F_x}_i^{ego}(p), \tilde{\mathbf{F}_{\mathbf{x}}}_i^{ego}(p))\right)],\end{aligned} \tag{9}$$

$$\begin{aligned}\hat{\mathcal{L}}_i^{\mathcal{F}_y} = \frac{1}{\mathbf{N}} \sum_{p=1}^{\mathbf{N}} \sum_{j=1}^{5} \mathcal{M}_i^p \cdot [\tilde{s}_i^{\mathcal{F}_y^j}(p) \\ + \exp\left(\tilde{s}_i^{\mathcal{F}_y^j}(p) \cdot \mathcal{E}^{\mathcal{F}}(\mathbf{F_y}_i^{ego}(p), \tilde{\mathbf{F}_{\mathbf{y}}}_i^{ego}(p))\right)].\end{aligned} \tag{10}$$

where $\tilde{s}_i^{\mathcal{D}_j}(p) = \log(\tilde{\mathcal{U}}_i^{\mathcal{D}_j}(p) \cdot \lambda^{\mathcal{D}})$, $\tilde{s}_i^{\mathcal{F}_x^j}(p) = \log(\tilde{\mathcal{U}}_i^{\mathcal{F}_x^j}(p) \cdot \lambda^{\mathcal{F}})$, $\tilde{s}_i^{\mathcal{F}_y^j}(p) = \log(\tilde{\mathcal{U}}_i^{\mathcal{F}_y^j}(p) \cdot \lambda^{\mathcal{F}})$.

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 hyperparmeters employed for training our model are presented in Table 2.

Hyperparameters	Value
Batch size	100
Learning rate	1e-4
Numbers of epochs	100
Optimizer	Adam
Mean for normalizing $\mathbf{F}^{total}$	0
Standard deviation for normalizing input $\mathbf{F}^{total}$	200
k	32
$\lambda^{\mathcal{D}}$	0.25
$\lambda^{\mathcal{F}}$	0.1
$\delta^{\mathcal{F}}$	20

Table 2: The hyperparamters for training PWVO.

	μ	σ_1	σ_2	g	ρ	c	d
$\bar{f}$	0.03	0.5	0.08	2.0	0.4	0.0	1.0
$\bar{\mathbf{D}}_{bg}$	0.6	0.2	0.1	2.0	0.5	0.0	1.0
$\bar{\mathbf{D}}_{obj}$	0.6	0.6	0.2	1.5	0.4	0.0	1.0
$\mathbf{R}_x$	0.0	1.5	0.2	2.5	0.15	-9.0	9.0
$\mathbf{R}_y$	0.0	2.0	0.2	2.0	0.15	-9.0	9.0
$\mathbf{R}_z$	0.0	1.5	0.2	2.5	0.15	-8.0	8.0
$\mathbf{T}_x$	0.0	0.4	0.03	2.5	0.15	-0.7	0.7
$\mathbf{T}_y$	0.0	0.25	0.025	2.0	0.20	-0.4	0.4
$\mathbf{T}_z$	0.0	1.80	0.02	2.5	0.15	-5.0	5.0

Table 3: A summary of the parameters used in the proposed data generation workflow.

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 configurating their distributions are summarized in Table 3. Similar to the approach adopted in [1], the distributions ζ are defined as the following:

$$\zeta = \beta \cdot max(min(sign(\eta) \cdot |\eta|^g, c), d) + (1-\beta) \cdot \tau, \tag{11}$$

where β is a Bernoulli random variable. It takes on a one with probability ρ, and a zero with probability $1 - \rho$.

According to Eq. (11), the distributions ζ 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 $\bar{f}$ sampled from the distribution defined above, f can be obtained as follows:

$$f = \bar{f} \cdot (f_{max} - f_{min}) + f_{min}, \tag{12}$$

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

Background depth $\mathbf{D}_{bg}$ and object depth $\mathbf{D}_{obj}$. The background depth can be obtained by:

$$\begin{aligned}\mathbf{D}_{bg} = \beta_{bg} \cdot (\bar{\mathbf{D}}_{bg} \cdot (d_2 - d_1 + d_1) \\ + (1 - \beta_{bg}) \cdot h(x, d_2, d_3)\end{aligned} \tag{13}$$

where β_{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:

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

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:

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

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 ablatively 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 $\mathbf{Sintel}_{ego}$ than on $\mathbf{Sintel}_{total}$, 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 $\mathbf{Custom}_{total}$ exhibit better generalizability on $\mathbf{Sintel}_{total}$ than those trained on $\mathbf{Custom}_{ego}$, as the latter does not involve any moving object during the training.

	Train	Eval	EPE	R_{err}	T_{err}
VONet	$\mathbf{Custom}_{total}$	$\mathbf{Sintel}_{ego}$	0.513	0.086	0.057
		$\mathbf{Sintel}_{total}$	0.909	0.110	0.061
	$\mathbf{Custom}_{ego}$	$\mathbf{Sintel}_{ego}$	0.296	0.055	0.038
		$\mathbf{Sintel}_{total}$	1.373	0.157	0.055
PWVO (naive)	$\mathbf{Custom}_{total}$	$\mathbf{Sintel}_{ego}$	0.526	0.072	0.057
		$\mathbf{Sintel}_{total}$	0.829	0.091	0.061
	$\mathbf{Custom}_{ego}$	$\mathbf{Sintel}_{ego}$	0.259	0.050	0.043
		$\mathbf{Sintel}_{total}$	1.543	0.196	0.062
PWVO	$\mathbf{Custom}_{total}$	$\mathbf{Sintel}_{ego}$	0.396	0.067	0.039
		$\mathbf{Sintel}_{total}$	0.626	0.081	0.043
	$\mathbf{Custom}_{ego}$	$\mathbf{Sintel}_{ego}$	0.187	0.037	0.022
		$\mathbf{Sintel}_{total}$	1.490	0.174	0.043

Table 4: An analysis for examining the impact of moving objects on the performence of VO. $\mathbf{Custom}$ and $\mathbf{Sintel}$ denote the datasets generated by our workflow and Sintel, respectively. The subscripts $total$ and ego represent that the inputs involve $\mathbf{F}_{obj}$+$\mathbf{F}_{ego}$ or $\mathbf{F}_{ego}$ only, respectively.

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 $(\tilde{\mathcal{R}}_i, \tilde{\mathcal{T}}_i)$ and $(\tilde{\mathcal{U}}_i^{\mathcal{R}}(p), \tilde{\mathcal{U}}_i^{\mathcal{T}}(p))$ 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 $(\tilde{\gamma}_i, \tilde{\varphi}_i)$ 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.

	EPE	R_{err}	T_{err}
Patch size = $H \times W$	0.652	0.086	0.034
Patch size = 64×64	0.654	0.086	0.068
Patch size = 32×32 (default)	0.626	0.081	0.043
Patch size = 16×16	0.567	0.075	0.051
Patch size = 1×1	0.690	0.077	0.041

Table 5: An analysis for comparing the performances of PWVO for different choices of k.

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

[1] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2758–2766, 2015.

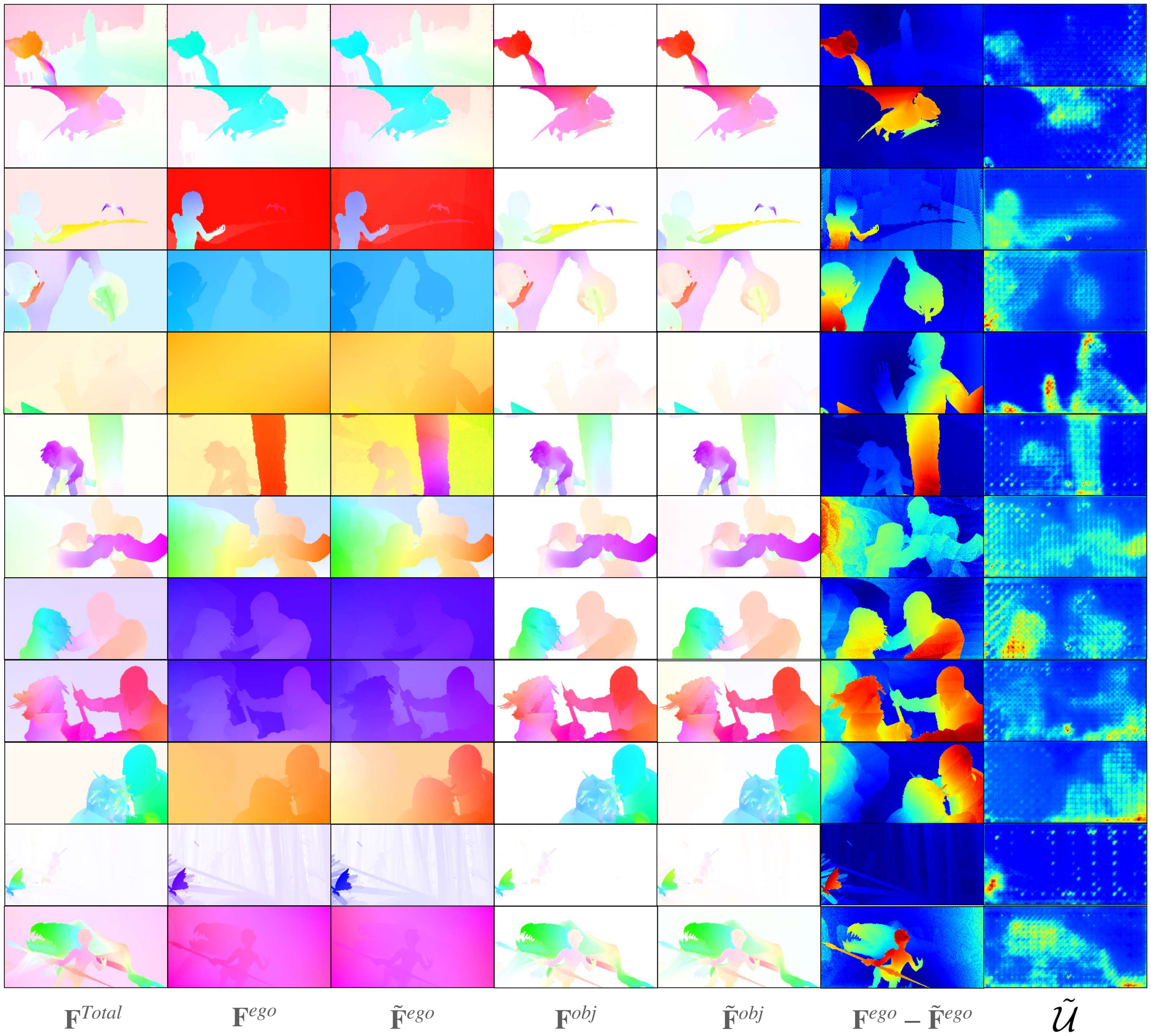

Figure 1: The qualitative results of PWVO evaluated on the Sintel dataset.

[2] Eddy Ilg, N. Mayer, Tonmoy Saikia, Margret Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017.

[3] Tak-Wai Hui and Chen Change Loy. Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation. *ArXiv*, abs/2007.09319, 2020.

[4] Deqing Sun, X. Yang, Ming-Yu Liu, and J. Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. In *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, volume 42, pages 1408–1423, 2020.

[5] Shengyu Zhao, Yilun Sheng, Y. Dong, E. Chang, and Y. Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6277–6286, 2020.

[6] Zhichao Yin, Trevor Darrell, and F. Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6037–6046, 2019.

[7] J. Wulff, Laura Sevilla-Lara, and Michael J. Black. Optical flow in mostly rigid scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6911–6920, 2017.

[8] A. Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2720–2729, 2017.

[9] Junhwa Hur and S. Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proc. IEEE*

Conf. on Computer Vision and Pattern Recognition (CVPR), pages 5747–5756, 2019.

[10] Zachary Teed and Jun Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. European Conf. on Computer Vision (ECCV)*, 2020.

[11] Gengshan Yang and D. Ramanan. Volumetric correspondence networks for optical flow. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

[12] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perciever io: A general architecture for structured inputs and outputs. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

[13] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. *ArXiv*, abs/2203.16194, 2022.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[15] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 611–625, 2012.

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. In *The International Journal of Robotics Research*, pages 1231–1237, 2013.

[17] Nikolaus Mayer, Eddy Ilg, Philip Häusser, 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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] Wenshan Wang, Yaoyu Hu, and Sebastian A. Scherer. Tartanvo: A generalizable learning-based vo. In *Proc. Conf. on Robot Learning (CoRL)*, 2020.