DPFlow: Adaptive Optical Flow Estimation with a Dual-Pyramid Framework

Supplementary Material

1. Training and evaluation details

Table 1 shows the hyperparameters we used during each training stage.

2. Checkpoint results

We report the results of our model after each of the four training stages in Tab. 2. These results can be used as a sanity check to reproduce the training of our model.

3. Model details

Table 3 details different models' size, time, and memory costs with input resolutions ranging from 1K to 8K. Due to deep feature pyramids, DPFlow has low computational costs and offers relatively low latency, being faster than most top-performing methods. FlowFormer++ and MatchFlow can reduce memory consumption using input tiling, but this strategy increases inference time considerably. RPKNet can achieve lower computational costs, but DPFlow offers a 23% improvement on the KITTI 2015 benchmark.

4. Evaluation details

4.1. Metrics

This section explains how to calculate the metrics we use for the evaluations in each benchmark. Let $F \in \mathbb{R}^{N \times 2}$ be the predicted flow and $G \in \mathbb{R}^{N \times 2}$ the groundtruth, where $N = HW$ for height H and width W. Then, the metrics are calculated as follows.

End-Point-Error (EPE): consists of the average Euclidean distance between the predicted flow vectors and the groundtruth ones:

$$\text{EPE}(F, G) = \frac{1}{N} \sum_{i=1}^{N} \|F_i - G_i\|_2. \tag{1}$$

Outlier ratio: represents the percentage of predictions whose Euclidean distances are beyond a given threshold τ:

$$\text{Outlier}(F, G) = \frac{1}{N} \sum_{i=1}^{N} [\|F_i - G_i\|_2 > \tau], \tag{2}$$

where $[\cdot]$ is the Iverson bracket. Each benchmark selects a specific threshold value and represents its metric by a different name. For example, the KITTI benchmark [13] adopts $\tau = \max(3, 0.05\|G_i\|_2)$ and calls their metric **Fl-All**. On the other hand, the Spring benchmark [12] uses $\tau = 1$ and names their metric as **1px**.

Weighted Area Under the Curve (WAUC): used by the VIPER dataset [17] to integrate over multiple thresholds and give more importance to more accurate results.

$$\text{WAUC}(F, G) = \frac{100}{N \sum_{i=1}^{100} w_i} \sum_{i=1}^{100} w_i \sum_{j=1}^{N} [\|F_j - G_j\|_2 \leq \delta_i], \tag{3}$$

where $[\cdot]$ is the Iverson bracket, $\delta_i = \frac{i}{20}$ and $w_i = 1 - \frac{i-1}{100}$.

4.2. Spring dataset metrics

To improve the evaluation of predictions for very thin structures, the Spring benchmark [12] proposed to generate the groundtruth at doubled resolution (image is 2K and groundtruth is 4K). Therefore, each prediction (at 2K) is compared with its four corresponding groundtruth values at 4K, and the lowest error is chosen as the metric. We follow the same procedure when evaluating the train results at 2K in Tab. 3 in the main paper. Unlike the study by SEA-RAFT [22], we also use the images at full resolution instead of downsampling them to 1K, which explains the difference in our results.

For the Spring train 4K evaluation in Tab. 3 in the main paper, we use bicubic interpolation to upsample the images from 2K to 4K. We also double the flow groundtruth values since they are originally encoded for a 2K resolution evaluation. Since the upsampling may create artifacts at the motion boundaries, we create a mask to ignore pixels near the borders. We create this mask using a strategy similar to the one the authors proposed for handling thin structures. We first reduce the 4K groundtruth to 2K resolution by stacking 2×2 pixel blocks. Then, we calculate the EPE metric between all combinations of the four pixels inside each 2×2 block. If the maximum EPE within a block is greater than one, we consider this block to lie on a motion boundary and ignore all four pixels. The Python code describing this procedure is shown in Listing 1.

5. Choosing the evaluation checkpoint

Here, we include the individual results for each method we used to calculate the averages presented in Tab. 2 in the main paper.

5.1. Spring dataset

Due to the large size of the Spring dataset [12], we only use the first 10% samples from each sequence for this experiment (480 samples). The results are presented in Tab. 4.

Table 1. Hyperparameters for each training stage. Batch sizes $\times 4$ indicate that gradients were accumulated during 4 iterations before backpropagating. Dataset abbreviations denote C: FlyingChairs [3], T: FlyingThings3D [11], H: HD1K [10], S: Sintel [1], K: KITTI 2015 [13], Sp: Spring [12].

Stage	Datasets	Batch	Iters (Epochs)	LR	WD	GPU hrs.
1	C	10	100k (45)	2.5×10^{-4}	10^{-4}	70
2	T	6	1M (80)	1.25×10^{-4}	10^{-4}	700
3	T+H+S+K	6×4	120k (25)	1.25×10^{-4}	10^{-5}	300
4a	K	6	2k (250)	1×10^{-4}	10^{-5}	6
4b	Sp	4×4	120k (100)	1×10^{-4}	10^{-5}	360

Table 2. Results after each training stage. Values in parentheses indicate that the same dataset was used during training.

Stage	S. Clean	S. Final	KITTI 2015	
	EPE	EPE	EPE	Fl-All
1	2.74	3.91	11.1	32.1
2	1.02	2.26	3.37	11.1
3	(0.55)	(0.86)	(1.04)	(3.18)
4a	0.80	1.24	(0.83)	(2.24)
4b	1.37	2.45	6.36	15.7

5.2. VIPER dataset

Due to the large size of the VIPER dataset [17], we only use the first 10% samples from each sequence for this experiment (473 samples). The results are presented in Tab. 5.

5.3. Middlebury-ST dataset

We use all 23 training samples of the Middlebury-ST dataset [18] for this experiment. The results are presented in Tab. 6.

5.4. Kubric-1K dataset

We use all 600 samples at 1K resolution from our proposed Kubric-NK dataset for this experiment. The results are presented in Tab. 7.

6. Feature visualization

We study the activation patterns of the CGU cross-gate to see what type of pattern they focus on. Here, we select the output feature of the encoder network and save the gate activations using heatmaps. Figures 1 and 2 show the gate activations on samples from the Kubric-NK (8K resolution) and KITTI 2015 (1K), respectively. We observe that some gates can capture some structural information about the scene. For example, some channels focus on horizontal and vertical lines, while others separate background from foreground and encode motion boundaries.

Listing 1. Python code for masking out pixels on the motion boundaries to calculate the metric for the Spring train 4K evaluation in Tab. 3 in the main paper.

```python
import torch
import torch.nn.functional as F
from einops import rearrange

def compute_valid_mask(flow: torch.Tensor):
    # flow shape is [B, 2, H, W]
    # downsample by stacking 2x2 blocks
    flow_stack = rearrange(
        flow,
        "b c (h nh) (w nw) -> b (nh nw) c h w",
        nh=2,
        nw=2
    )
    # calculate EPE between all pairs
    # in each 2x2 block
    flow_stack4 = flow_stack.repeat(
        1, 4, 1, 1, 1)
    flow_stack4 = rearrange(
        flow_stack4,
        "b (m n) c h w -> b m n c h w",
        m=4)
    diff = flow_stack[:, :, None] - flow_stack4
    diff = rearrange(
        diff,
        "b m n c h w -> b (m n) c h w"
    )
    diff = torch.sqrt(torch.pow(diff, 2).sum(2))
    # mark pixels as valid only if the maximum
    # EPE of their block is below one
    max_diff, _ = diff.max(1)
    max_diff = F.interpolate(
        max_diff[:, None],
        scale_factor=2,
        mode="nearest"
    )
    valid_mask = max_diff < 1.0
    return valid_mask
```

7. Input downsampling

One popular strategy to avoid handling high-resolution inputs consists of first downsampling the inputs and then upsampling the predictions back to the original resolution. Here, we conduct a study to evaluate how this input down-

Table 3. Model details. □ denotes methods that use input tiling [5, 6]. * indicates that the model is running in local correlation mode, which decreases memory consumption but increases inference time. † results are not available due to running out of memory. Times are calculated using an NVIDIA RTX3090 GPU.

Model	KITTI 2015	Params	Time (s)				Memory (GB)			
	(Fl-All)	(M)	1K	2K	4K	8K	1K	2K	4K	8K
RAFT [21]*	5.10	5.25	0.51	1.36	4.76	†	0.65	1.71	6.40	†
GMA [9]	5.25	5.88	0.18	1.17	†	†	1.41	17.3	†	†
DIP [26]	4.21	5.37	0.51	2.11	8.60	†	1.59	5.46	20.9	†
SKFlow [20]	4.84	6.27	0.31	1.70	†	†	1.17	17.4	†	†
FlowFormer++ [19]□	4.52	16.1	1.17	2.68	9.02	29.9	4.24	4.37	6.10	18.7
MatchFlow [2]□	4.63	15.4	0.63	1.44	5.82	22.8	1.31	1.61	**3.04**	20.7
MS-RAFT+ [7]*	4.19	16.1	0.74	2.82	†	†	2.61	9.34	†	†
GMFlow+ [25]	4.49	7.36	0.23	1.69	†	†	2.99	20.6	†	†
RPKNet [15]*	4.64	**2.84**	**0.08**	**0.28**	**1.18**	**5.34**	**0.56**	**1.34**	4.42	**16.8**
SEA-RAFT (L) [22]*	4.30	19.6	0.24	0.70	2.60	†	0.79	2.02	7.21	†
CCMR+ [8]*	3.86	11.5	1.34	5.36	†	†	3.19	12.1	†	†
DPFlow*	**3.56**	10.0	0.16	0.53	1.98	8.40	0.76	1.90	6.70	20.7

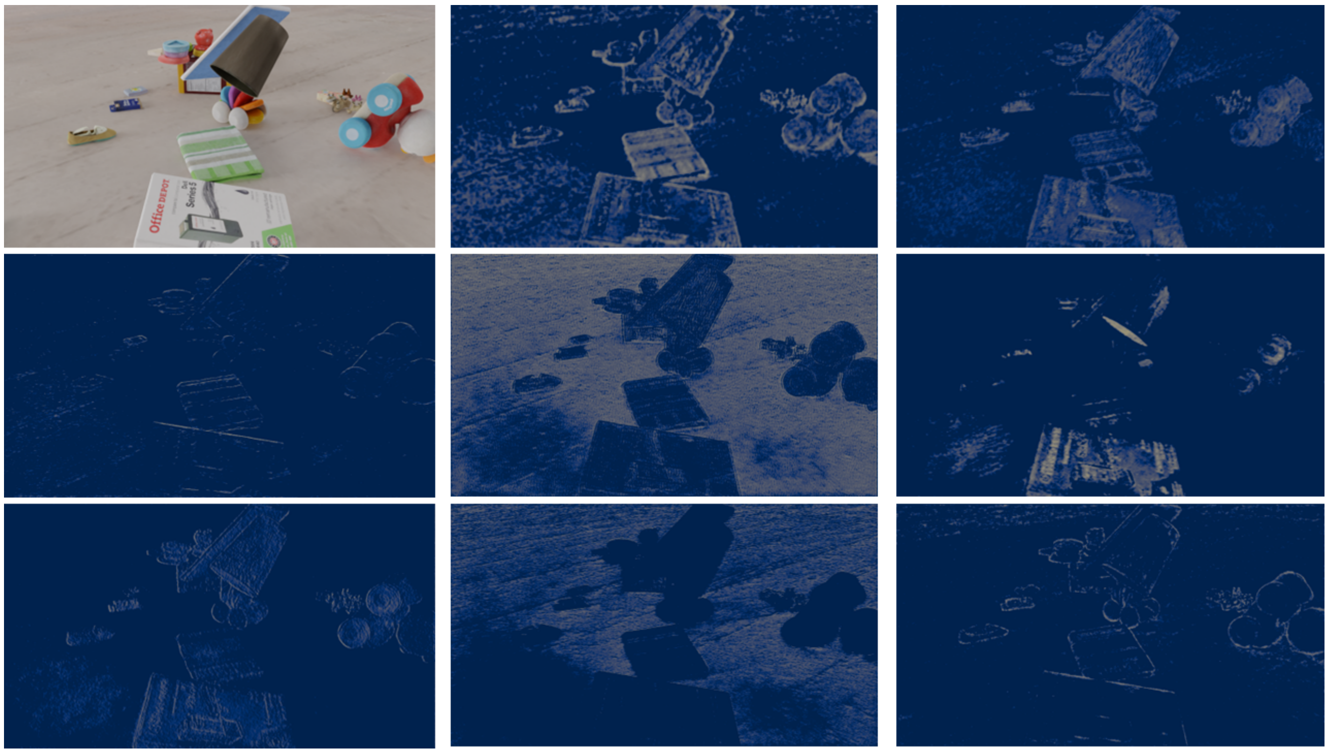


Figure 1. Visualization of cross-gate activations on an 8K sample from the Kubric-NK dataset.

sampling affects the results. We use samples from three high-resolution datasets: Spring [12] (2K), Middlebury-ST [18] (3K), and Kubric-NK (8K). For this study, we use bilinear interpolation to first downsample the inputs to 1K resolution, and then upsample the predictions back to the original sizes before computing the accuracy metrics. The results in Tab. 8 show some interesting observations. Downsampling from 2K to 1K in the Spring benchmark negatively impacts all methods except GMFlow+. This result indicates that most approaches are robust to this input size, but the global matching strategy from GMFlow+ may have more bias towards the training sizes. On the Middlebury-

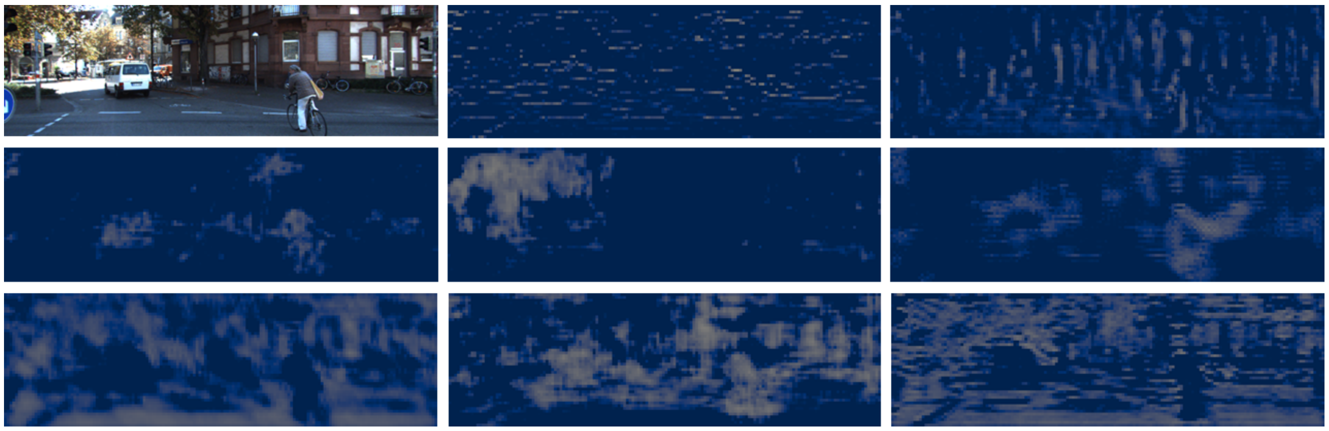


Figure 2. Visualization of cross-gate activations on a 1K sample from the KITTI 2015 dataset.

Table 4. Training stage experiments on the Spring dataset.

Method	1px ↓	
	Stage 2	Stage 3
PWC-Net	7.18	5.15
IRR	5.46	4.52
VCN	7.53	8.08
RAFT	4.45	3.85
GMA	4.33	3.67
DIP	4.36	4.21
GMFlow	5.92	6.83
FlowFormer	4.16	3.93
SKFlow	4.61	3.93
FlowFormer++	4.1	4.21
MatchFlow	4.33	4.15
GMFlow+	5.04	5.07
RPKNet	3.95	3.86
SEA-RAFT (M)	4.37	3.63
RAPIDFlow	4.32	3.94
DPFlow	4.76	3.53
Average	4.92	**4.53**

Table 5. Training stage experiments on the VIPER dataset.

Method	WAUC ↑	
	Stage 2	Stage 3
PWC-Net	51.2	61.3
IRR	58.1	61.4
VCN	55.6	58.4
RAFT	63.7	67.1
GMA	64	67.7
DIP	66.8	67.7
GMFlow	52.9	52.4
FlowFormer	66.1	68.8
SKFlow	64.4	67.8
FlowFormer++	66.8	68.6
MatchFlow	66.3	69.0
GMFlow+	64.8	65.8
RPKNet	66.9	68.7
SEA-RAFT (M)	68.9	71.8
RAPIDFlow	62.7	66.5
DPFlow	70.4	72.7
Average	63.1	**65.9**

ST 3K samples, we observe significant improvements in methods with fixed structures. On the other hand, adaptable methods like DPFlow and RAPIDFlow still benefit from processing the images at the original resolution. Using an eight-times downsampling on Kubric-NK benefits almost all methods. However, these results may be caused by the relatively simple shapes and motions generated by Kubric [4]. Since the objects do not present fine-grained details, downsampling them does not affect the input contents significantly. A more thorough study of the effects of downsampling would require a high-resolution dataset with finer and more complex motion patterns.

8. Kubric-NK

This section details the Kubric-NK dataset and presents additional discussions about the effect of input resolution on the performance of optical flow methods.

8.1. Dataset samples

Figure 3 shows an image pair and its respective optical flow annotation for each of the 30 sequences of the Kubric-NK dataset.

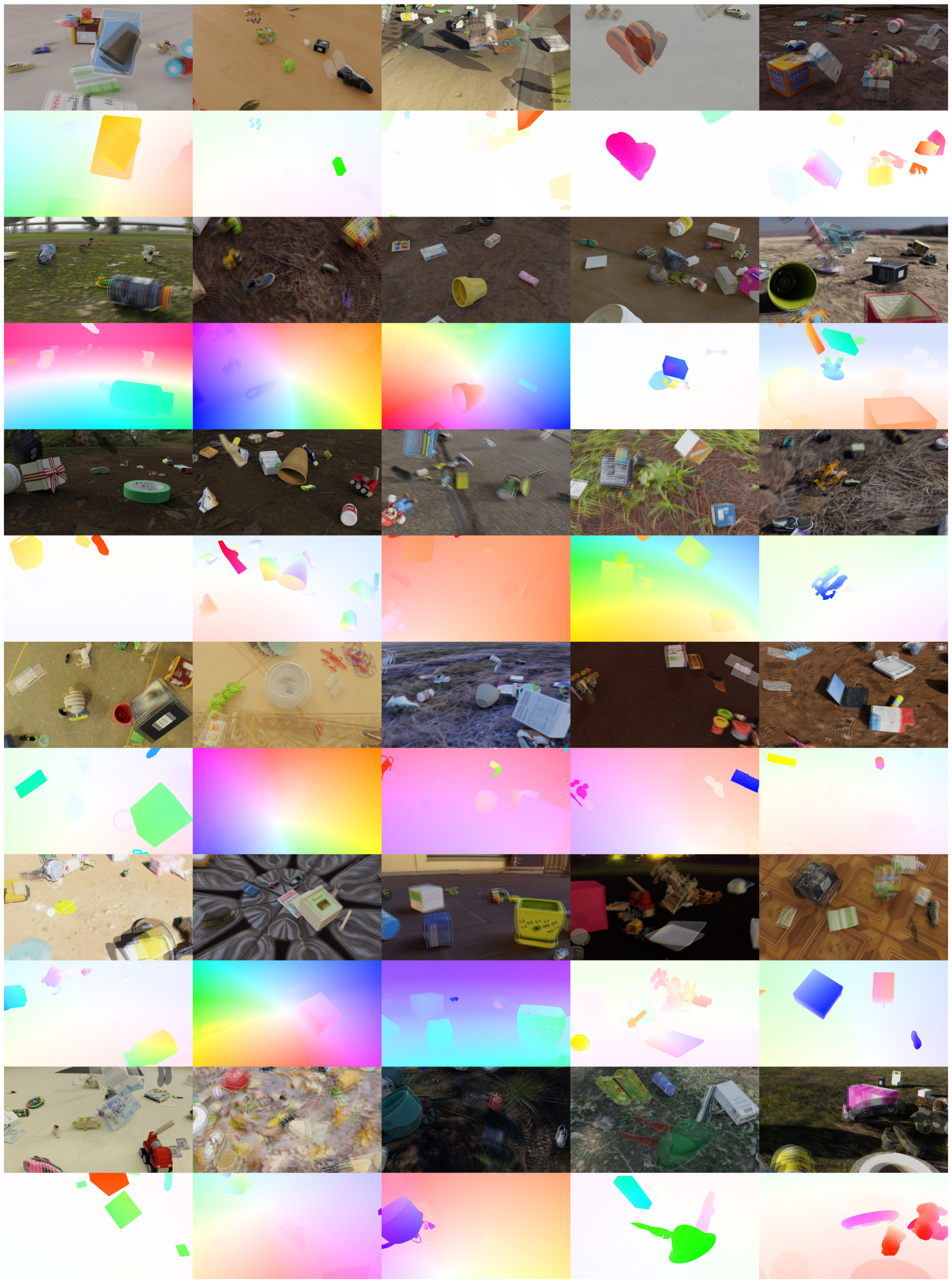


Figure 3. Samples from all 30 sequences from the Kubric-NK dataset. Odd rows show two consecutive images overlayed; even rows show their respective flow.

Table 6. Training stage experiments on the Middlebury-ST dataset.

Method	EPE ↓	
	Stage 2	Stage 3
PWC-Net	35.9	74.3
IRR	15.2	61.4
VCN	44.2	51
RAFT	35.2	51
DIP	16.7	21.2
FlowFormer	19.0	13.8
FlowFormer++	17.5	15.2
MatchFlow	16.7	19.4
RPKNet	15.0	16.4
SEA-RAFT (M)	65.1	59.2
RAPIDFlow	6.41	7.33
DPFlow	4.82	5.25
Average	**24.3**	32.9

Table 7. Training stage experiments on the Kubric-1K dataset.

Method	EPE ↓	
	Stage 2	Stage 3
PWC	1.4	1.28
IRR	0.95	1.1
VCN	1.23	1.09
RAFT	0.73	0.69
GMA	0.7	0.67
DIP	0.71	0.68
GMFlow	0.68	0.88
FlowFormer	0.5	0.52
SKFlow	0.69	0.62
FlowFormer++	0.48	0.46
MatchFlow	0.54	0.54
GMFlow+	0.48	0.45
MemFlow	0.57	0.55
SEA-RAFT(M)	0.56	0.53
RPKNet	0.61	0.6
RAPIDFlow	0.85	0.81
DPFlow	0.57	0.5
Average	0.72	**0.70**

8.2. Generalization to different resolutions

Optical flow estimation is fundamentally a point-matching problem between a pair of images. Therefore, the overall magnitude of the flow (correspondence) vectors tends to correlate with the size of the inputs, making it more difficult for higher-resolution inputs. There are two main optical flow estimation approaches: global and local matching. Global matching can theoretically handle changes in the input size and large motions. However, the cost of global search increases quadratically according to the input size, which makes it unfeasible for larger inputs. Local matching only searches for correspondences within a limited window (typically 9×9) around the origin point. This approach, however, requires the corresponding points to be within the range of the search window. This causes a problem for most local methods because they adopt an encoder with a fixed structure, which makes the size of the feature directly determined by the input resolution. Figure 4 illustrates how the increase in input size affects the matching procedure and how DPFlow's recurrent encoder alleviates this problem.

8.2.1. Training bias

Besides the problems caused by the limited search range, the training distribution also heavily affects the performance of non-adaptive models to different inputs. We demonstrate the effect of training bias with two studies based on the open-source models Flow1D [24] and CroCo [23].

Flow1D provides an additional model variant trained using a dataset with high-resolution samples. We compare the performance of Flow1D on Kubric-NK using the standard (trained for MPI-Sintel) and high-resolution training in Tab. 9. The results demonstrate that high-resolution training is effective and necessary to produce stable predictions at 4K resolution. However, we also observe that it negatively affects the results at the lower 1K resolution, thus clearly showing the effect of the training bias.

Our second study evaluates the performance of the CroCo model on the training split of the Middlebury v3 stereo-matching benchmark [18]. This benchmark contains 15 image pairs ranging from 1K to 3K resolution inputs. Since stereo-matching can be viewed as a subset of optical flow (matching only on the horizontal axis), optical flow models can be used for this task by zeroing the y-axis prediction. We chose the CroCo model because it provides trained variants for stereo and optical flow tasks using a common backbone network. The only differences between these variants are the number of output prediction heads (1 for stereo vs. 2 for optical flow) and the training samples.

Table 10 shows how using these two variants affects the Middlebury v3 training benchmark results. Although optical flow and stereo matching are both similar matching problems, directly transferring the optical flow-trained model for stereo estimation results in a noticeable degradation. Since both variants share the same architecture, these results further illustrate the effect of the training bias.

These studies demonstrated that the problems of handling different inputs can be mitigated with specialized training. In particular, training different variants for particular input resolutions can result in significant improvements. Nonetheless, this strategy is only palliative since it can only shift the problem to another resolution range. Moreover,

Table 8. Results on high-resolution benchmarks with and without downsampling the inputs. The @1K columns indicate that the inputs are downsampled to 1K resolution and the predictions are upsampled back to the original sizes. † results are not available due to running out of memory.

Method	Spring (1px ↓)		Mbury-ST (EPE ↓)		Kubric-NK (EPE ↓)	
	2K	@1K	3K	@1K	8K	@1K
RAFT [21]	3.85	4.52	35.2	8.01	82.7	5.09
GMA [9]	3.75	4.53	†	8.23	†	4.91
DIP [26]	3.64	3.87	16.7	8.01	†	5.42
SKFlow [20]	3.79	4.20	†	6.88	†	4.72
FlowFormer++ [19]	3.76	4.21	17.5	6.93	24.4	**3.39**
MatchFlow [2]	3.97	4.21	16.7	7.18	29.0	3.89
GMFlow+ [25]	5.72	4.00	†	<u>5.67</u>	†	4.43
RPKNet [15]	<u>3.28</u>	4.12	15.0	6.74	18.2	4.59
SEA-RAFT (M) [22]	3.47	**3.53**	65.1	6.71	51.8	4.07
RAPIDFlow [14]	3.73	5.06	<u>6.41</u>	8.99	<u>5.96</u>	5.99
DPFlow	**3.06**	<u>3.57</u>	**5.09**	**5.63**	**4.13**	<u>3.88</u>

Table 9. Flow1D results using standard-resolution stage 3 training (for MPI-Sintel benchmark) and after high-resolution re-training.

Train resolution	Kubric-NK (EPE ↓)		
	1K	2K	4K
Standard (Sintel)	0.85	1.80	32.7
High-res.	0.88	1.42	4.49

Table 10. Results of evaluating CroCo [23] variants trained on stereo and optical flow samples on the Middlebury v3 stereo matching benchmark.

Variant	Middlebury v3 (training)		
	EPE ↓	1px ↓	2px ↓
CroCo-Stereo	0.37	3.86	1.93
CroCo-Flow	7.17	29.7	20.1

training multiple variants of each model is expensive, especially with high-resolution inputs. One advantage of the proposed DPFlow is that it significantly reduces the input resolution gap by changing its pyramids on the fly without requiring additional training.

8.3. Additional qualitative results

This section shows some additional qualitative results of DPFlow on some high-resolution inputs. We demonstrate results on samples at 4K resolution from DAVIS [16] in Fig. 5, 3K from Middlebury-ST [18] in Fig. 6, and 1K to 8K from Kubric-NK in Figs. 7, 8, 9.

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012. 2

[2] Qiaole Dong, Chenjie Cao, and Yanwei Fu. Rethinking optical flow from geometric matching consistent perspective. In *CVPR*, pages 1337–1347, 2023. 3, 7

[3] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, et al. FlowNet: learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 2

[4] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, et al. Kubric: A scalable dataset generator. In *CVPR*, pages 3739–3751, 2022. 4

[5] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, et al. FlowFormer: A transformer architecture for optical flow. In *ECCV*, pages 668–685, 2022. 3

[6] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, et al. Perceiver IO: a general architecture for structured inputs & outputs. In *ICLR*, 2022. 3

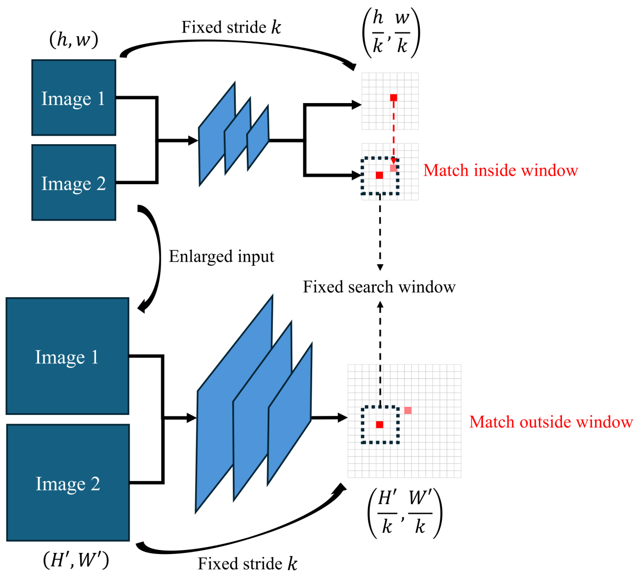
[7] Azin Jahedi, Maximilian Luz, Marc Rivinius, Lukas Mehl, and Andrés Bruhn. MS-RAFT+: High resolution Multi-Scale RAFT. *IJCV*, 2023. 3

[8] Azin Jahedi, Maximilian Luz, Marc Rivinius, and Andrés Bruhn. CCMR: High resolution optical flow estimation via coarse-to-fine context-guided motion reasoning. In *WACV*, 2024. 3

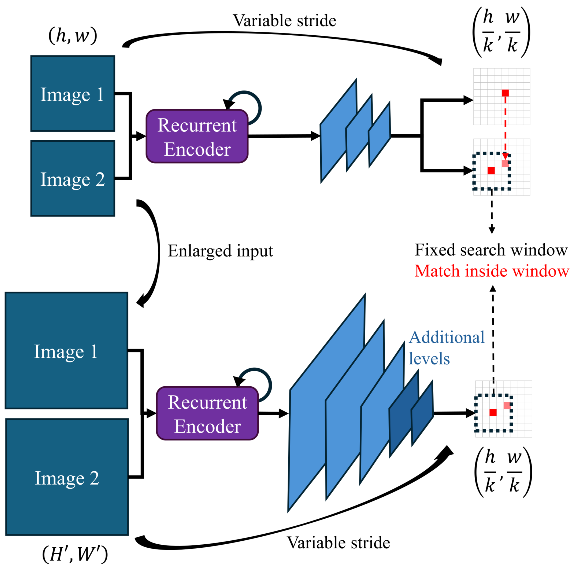
[9] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *ICCV*, pages 9752–9761, 2021. 3, 7

[10] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, et al. The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *CVPR Workshops*, pages 19–28, 2016. 2

[11] N. Mayer, E. Ilg, P. Häusser, P. Fischer, et al. A large dataset to train convolutional networks for disparity, optical



(a) Local search with fixed encoder



(b) DPFlow's adaptable encoder

Figure 4. Simple example of the local matching procedure on typical optical flow models. (a) The spatial size of the feature map extracted by an encoder with a fixed structure is determined by the encoder's total stride k (*e.g.*, 8 times smaller than the input). When the input resolution increases, the matched points are beyond the local window's range, causing a mismatch. (b) DPFlow adopts a recurrent encoder that can be dynamically unfolded to change the number of encoding levels and control the size of the initial feature map. This alleviates the window range problem.

flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016. 2

[12] Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *CVPR*, pages 4981–4991, 2023. 1, 2, 3

[13] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015. 1, 2

[14] Henrique Morimitsu, Xiaobin Zhu, Roberto M. Cesar-Jr, Xiangyang Ji, and Xu-Cheng Yin. RAPIDFlow: Recurrent Adaptable Pyramids with Iterative Decoding for efficient optical flow estimation. In *ICRA*, 2024. 7

[15] Henrique Morimitsu, Xiaobin Zhu, Xiangyang Ji, and Xu-Cheng Yin. Recurrent partial kernel network for efficient optical flow estimation. In *AAAI*, pages 4278–4286, 2024. 3, 7

[16] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, et al. The 2017 DAVIS challenge on video object segmentation. *ArXiv*, 2017. 7, 9

[17] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *ICCV*, pages 2232–2241, 2017. 1, 2

[18] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, et al. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, pages 31–42. Springer, 2014. 2, 3, 6, 7, 10

[19] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, et al. FlowFormer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *CVPR*, pages 1599–1610, 2023. 3, 7

[20] Shangkun Sun, Yuanqi Chen, Y. Zhu, Guodong Guo, and Gezhong Li. SKFlow: Learning optical flow with super kernels. In *NeurIPS*, 2022. 3, 7

[21] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419, 2020. 3, 7

[22] Yihan Wang, Lahav Lipson, and Jia Deng. SEA-RAFT: Simple, Efficient, Accurate RAFT for optical flow. In *ECCV*, 2024. 1, 3, 7

[23] Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, et al. CroCo v2: Improved cross-view completion pre-training for stereo matching and optical flow. In *ICCV*, pages 17969–17980, 2023. 6, 7

[24] Haofei Xu, Jiaolong Yang, Jianfei Cai, Juyong Zhang, and Xin Tong. High-resolution optical flow from 1D attention and correlation. In *ICCV*, pages 10478–10487, 2021. 6

[25] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, et al. Unifying flow, stereo and depth estimation. *TPAMI*, 45(11): 13941–13958, 2023. 3, 7

[26] Zihua Zheng, Ni Nie, Zhi Ling, Pengfei Xiong, et al. DIP: Deep inverse patchmatch for high-resolution optical flow. In *CVPR*, pages 8915–8924, 2022. 3, 7

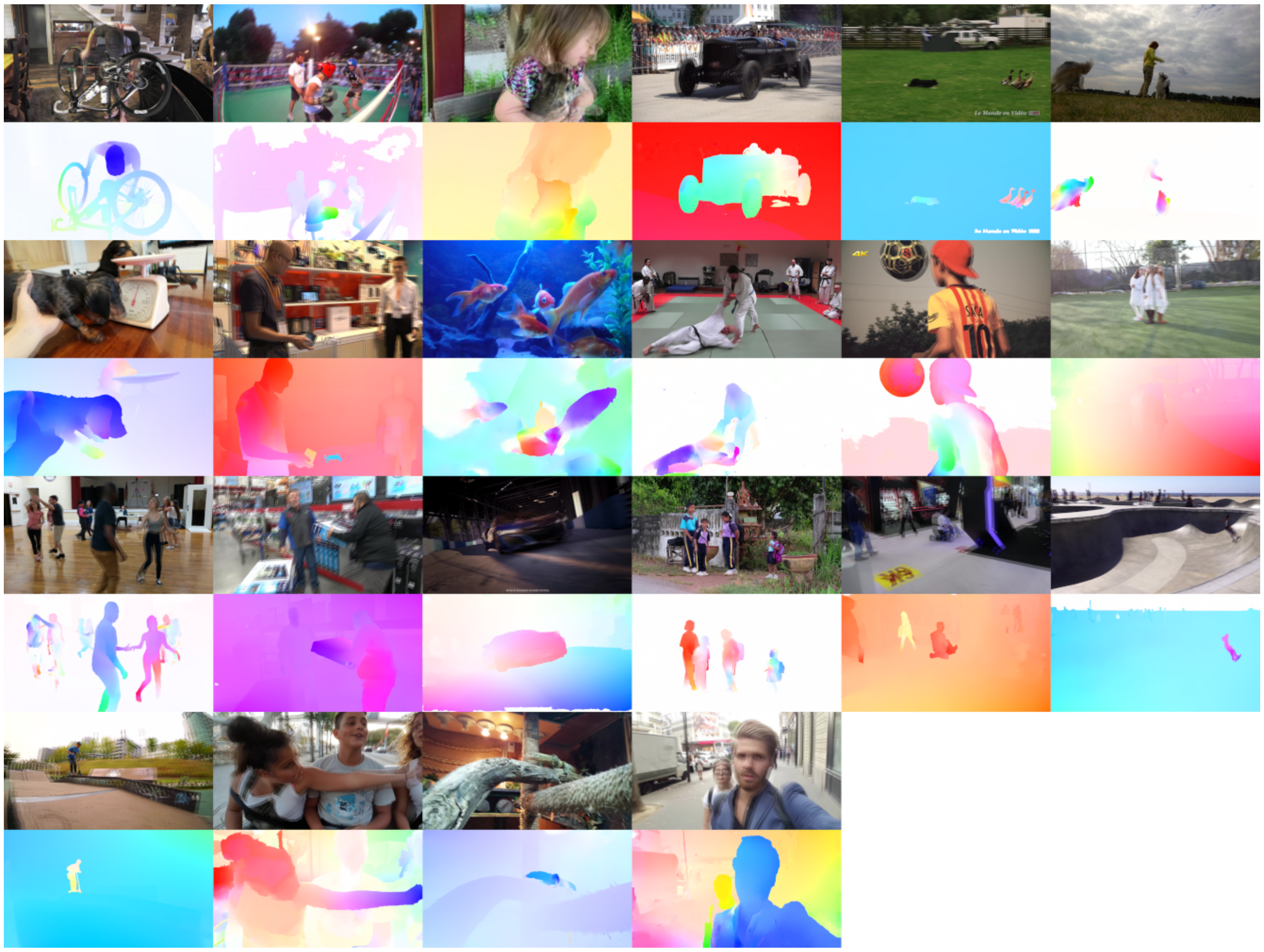


Figure 5. Qualitative results on 4K samples from the DAVIS dataset [16].

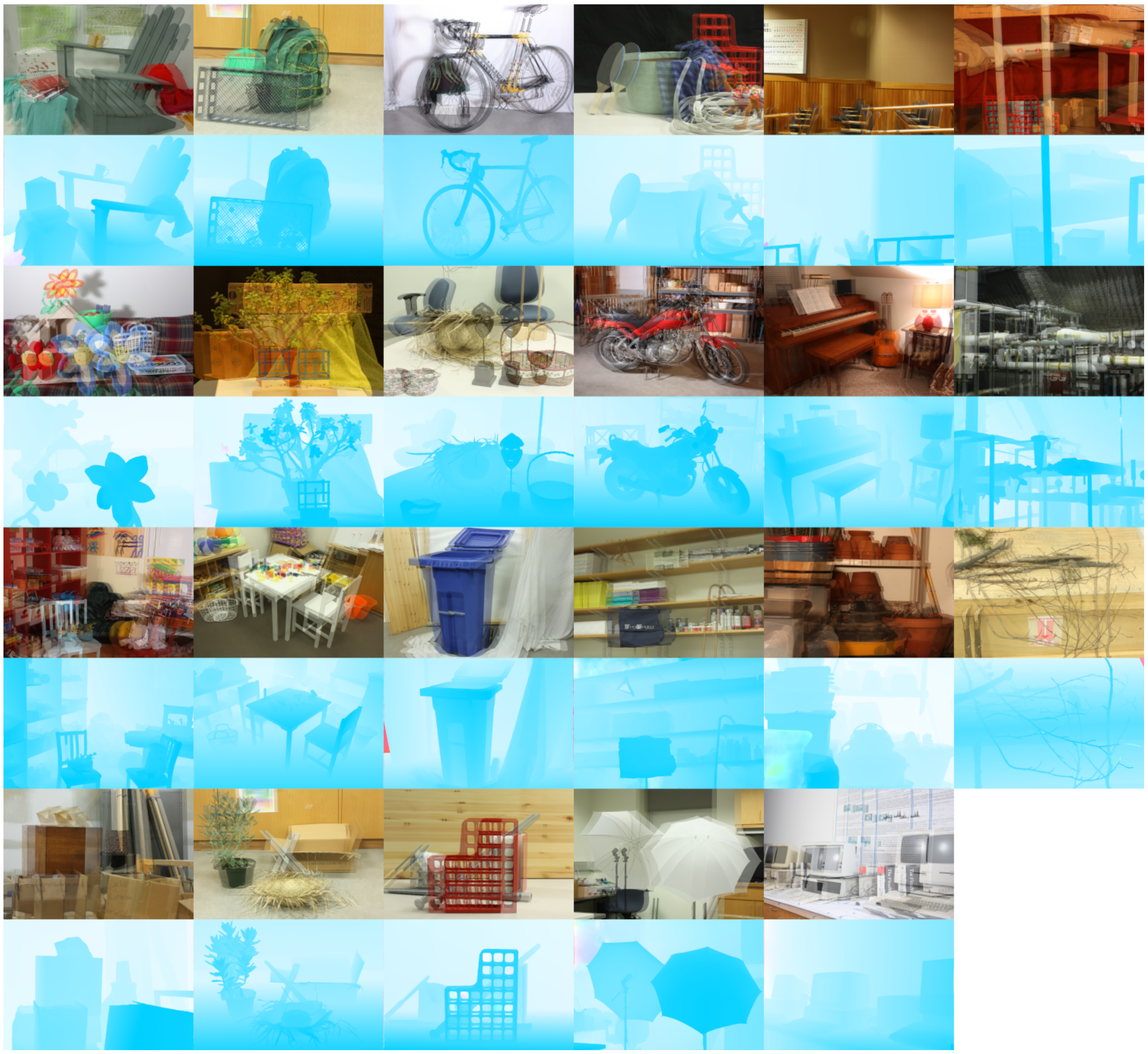


Figure 6. Qualitative results on 3K samples from the Middlebury-ST dataset [18].

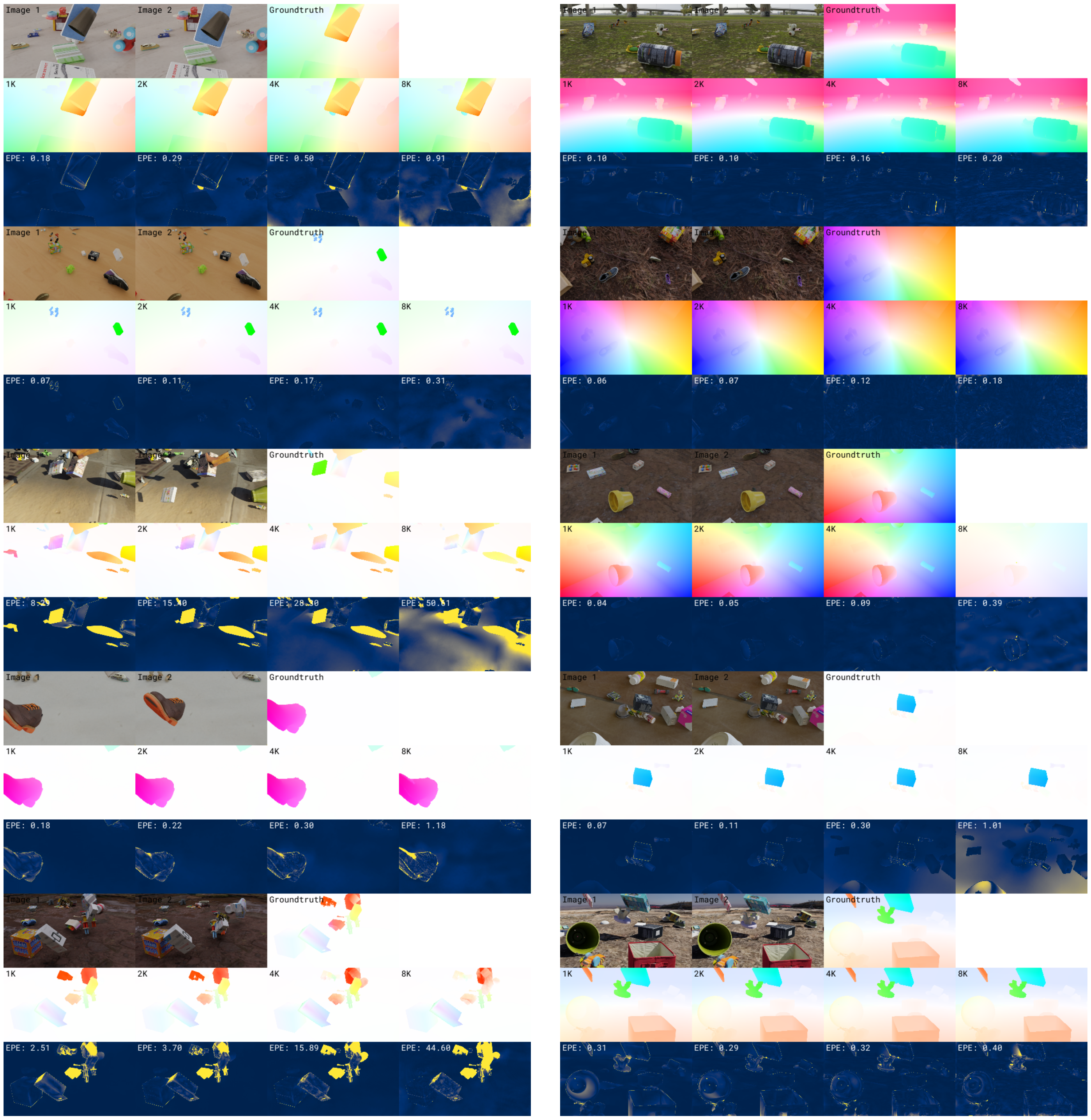


Figure 7. Qualitative results on 1K to 8K samples from our Kubric-NK dataset.

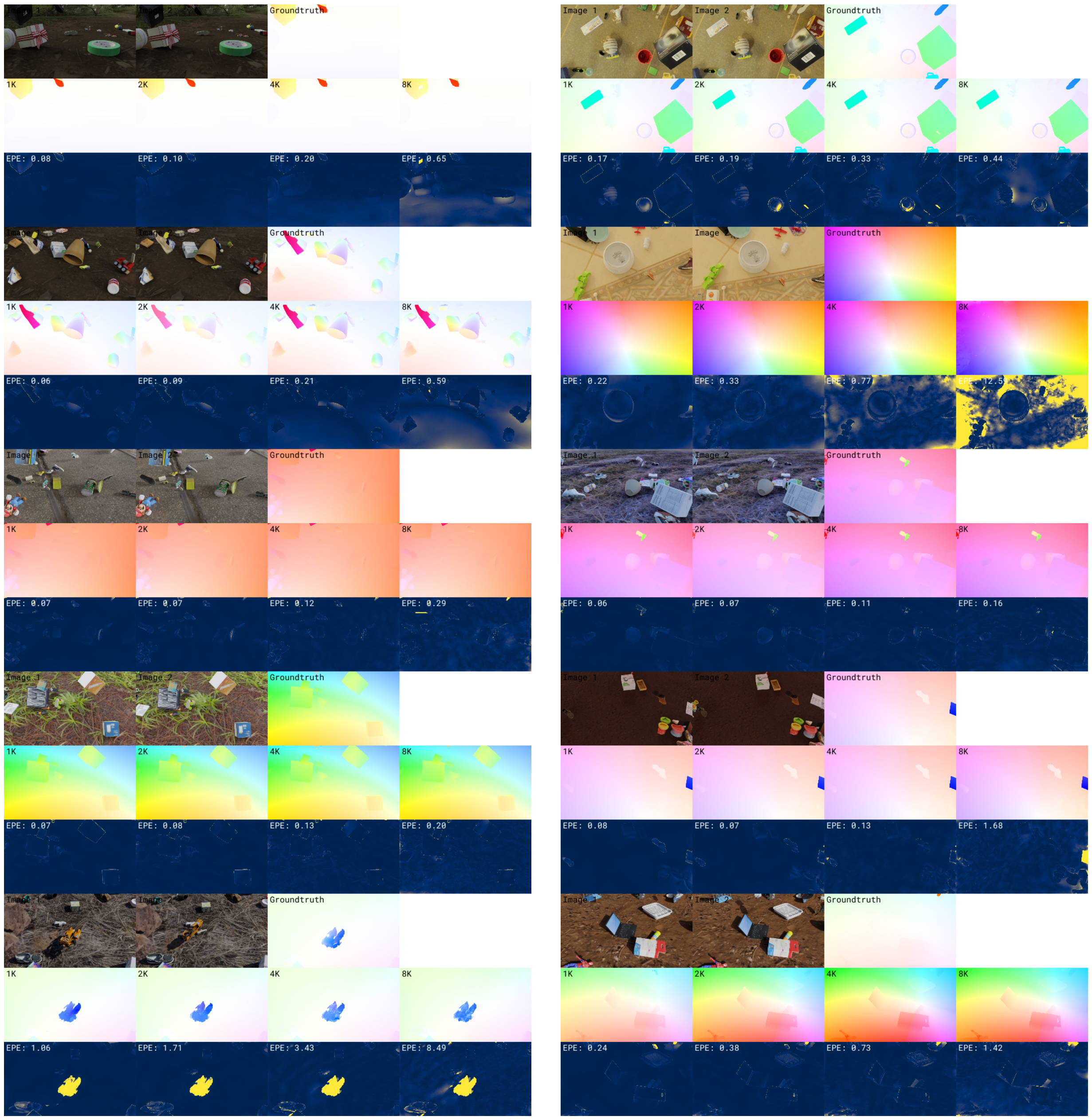


Figure 8. Qualitative results on 1K to 8K samples from our Kubric-NK dataset.

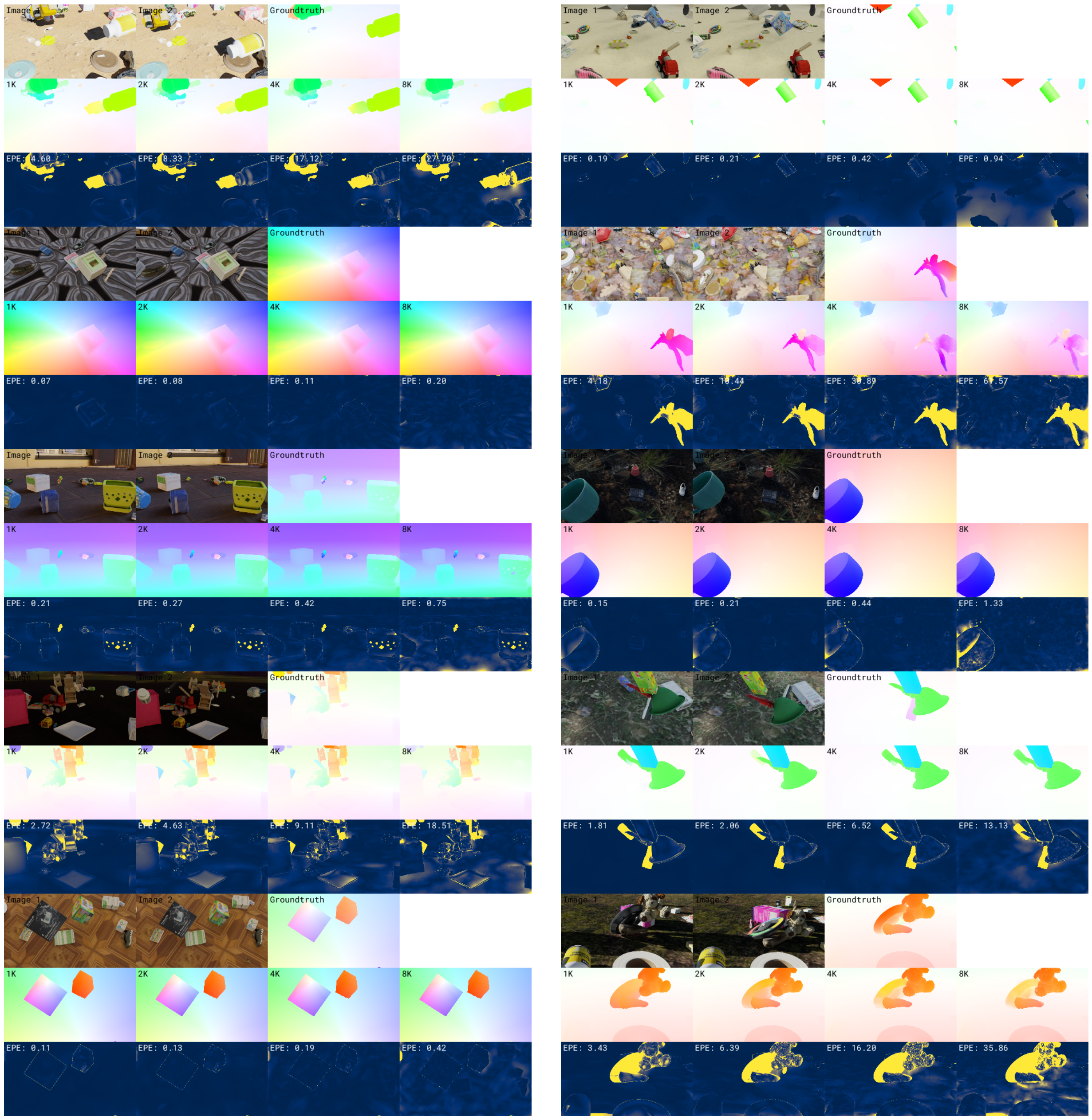


Figure 9. Qualitative results on 1K to 8K samples from our Kubric-NK dataset.