Self-supervised AutoFlow
Supplementary Materials

We first discuss implementation and experiment details. Next, we present the ablation studies of our approach. Third, we provide additional analysis of the proposed design. Finally, we include more visual results.

A. Implementation Details

Training details. We use 80 k training steps for the rendering hyperparameter search. We include the following rendering hyperparameters for generating the S-AF data:

- Number of foreground objects
- Scale, rotation, translation, grid strength, grid size of the motion for foreground
- Scale, rotation, translation, grid strength, grid size of the motion for background
- Probability and strength of the mask blur
- Probability and strength of the motion blur
- Probability, density and brightness of the fog
- Minimum and maximum of the object's diagonal
- Minimum and maximum of the object's center location
- Irregularity and spikiness of the polygon

As for the hyperparameters in the search metric Eq. (2) in the main paper, we use $(w_{\text{smooth}}, w_{\text{distill}}){=}(0.6, 4)$ for the Sintel [1] and the DAVIS dataset [3], and $(w_{\text{smooth}}, w_{\text{distill}}){=}(1.2, 8)$ for the KITTI dataset [2]. We pretrain the model on a generated S-AF dataset $\mathbf{D}_{\text{auto}}$ for 3.2 M iterations for Sintel and 200 k iterations for KITTI and DAVIS. We randomly crop input images to size 368×496 at training time and use a batch size of 36.

We further fine-tune the model with the self-supervised loss (Eq. (3)) for 12 k iterations on the Sintel dataset, 75 k iterations on the KITTI dataset, and 100 k iterations on the Davis dataset. We further apply the multi-frame fine-tuning on Sintel and KITTI datasets for 30 k iterations (Eq. (4)) with the same parameter setting from SMURF [4]. We randomly crop input images to size 368×496 at training time and use a batch size of 8. We use the data augmentations from RAFT [5] including random cropping, stretching, scaling, flipping, and erasing. As for the photometric augmentations, we randomly adjust the contrast, saturation, brightness and hue.

Evaluation metrics. We use the average end-point error (AEPE) evaluation metric. For KITTI, we additionally report the outlier rate (Fl-all), *i.e.* the ratio (in %) of outlier pixels among all ground truth pixels. If an error of a pixel exceeds the 3-pixel threshold and 5% w.r.t. the ground truth, the pixel is considered as an outlier.

B. Ablation Studies

B.1. Training by individual S-AF dataset and mixed S-AF dataset

As described in Sec. 3.2 and Sec. 4.2, to improve the robustness of the algorithm, we sort the sets of hyperparameters returned by Self-AutoFlow according to the self-supervised search metric and choose the top-3 hyperparameter sets. We form our final Self-AutoFlow dataset by equally mixing a set of images generated from each hyperparameter set. For a fair comparison, we also prepare an equivalent model for AutoFlow, denoted as AF-mix. In addition to the results of training on the dataset generated by mixing the top-3 hyperparameters in Tab. 1, we report the results of training models on each individual S-AF and AF dataset in Tab. B.1. The models are trained for 0.2M iterations. We note that the top hyperparameters sets are selected according to the search, where the model is trained for 40K iterations, and here we report the results of models trained for 0.2M iterations, so the top-1 hyperparameters might not have the lowest AEPE for AF models.

Unlike supervised AutoFlow, the results of S-AF trained on the top-2 hyperparameters on Sintel Final and S-AF trained on the top-3 hyperparameters on KITTI show that there is no guarantee that the top candidates returned by self-supervised AutoFlow are the optimal set of hyperparameters. Mixing the top-3 datasets decreases the likelihood of sampling a set of poor-performing AutoFlow hyperparameters and improves the robustness of the algorithm.

Table B.1. **Training by individual S-AF dataset and mixed S-AF dataset.** We show that training on mix-3 datasets decreases the likelihood of sampling a poor-performing AutoFlow hyperparameters and improves the robustness of the algorithm.

	Sintel Clean [1]				Sintel Final [1]				KITTI 2015 [2]			
Method	top-1	top-2	top-3	mix-3	top-1	top-2	top-3	mix-3	top-1	top-2	top-3	mix-3
AF-mix (0.2M)	2.11	2.18	2.10	2.18	2.85	2.83	2.82	2.83	4.70	4.35	4.58	4.43
S-AF (0.2M)	2.16	2.14	2.13	2.22	2.83	**2.93**	2.84	2.84	4.65	4.06	**5.40**	4.58

B.2. Sequence losses in the search metric of S-AF

In Sec. 3.2, we mention that since there is no backpropagation to the model in the search of AutoFlow, the search metric uses only the final flow prediction of RAFT instead of all intermediate. In Fig. B.1, we conduct a study using the intermediate predictions of RAFT to compute the search metric. Specifically, we compute the search metric once for each intermediate prediction and we exponentially decay the weight for earlier predictions [4]. Since the search metric is computed at the original resolution of the target data, we use at most the last four predictions due to memory constraints.

We conduct the S-AF search using last-1 prediction (ours), last-2 prediction, and last-4 prediction as the search metric. We report the average AEPE of the top-3 models selected by the search metric. The models are trained for 40k iterations in the search. Empirically, we find that using the intermediate predictions in the search metric results in a higher AEPE and does not improve the S-AF search.

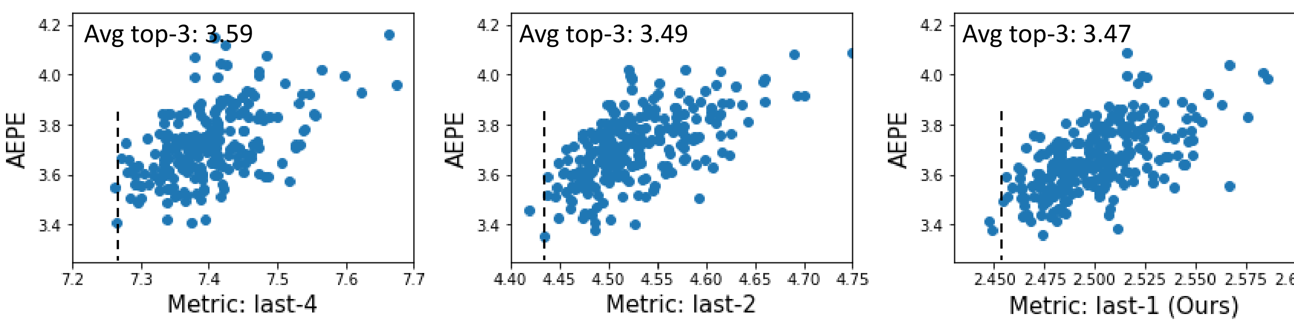

Figure B.1. **Sequence losses.** We find that using the intermediate predictions of RAFT to compute the search metric does not lead to a better set of S-AF hyperparameters.

C. Analysis and Discussion

Motion statistics of S-AF and AF We compute the statistics of the motion magnitude of the generated optical flow ground truth in S-AF and AF datasets in Fig. C.1. We find that when the target dataset is Sintel, the motion statistics of S-AF are similar to the statistics of Sintel data. In contrast, the motion statistics of AutoFlow are different from the Sintel data. In addition, S-AF focuses more on small motion compared to AF which focuses on the middle-range motion. We hypothesize that the self-supervised search metric may have much smaller values for middle/high-range motions compared to AEPE which penalizes significantly on the error at the regions of large motions. Therefore, the S-AF data does not focus on regions with large motions compared to AF. Similar to Tab. B.1, we also show the statistics of each individual S-AF dataset and the mixed dataset. We find the statistics are similar for each individual S-AF data.

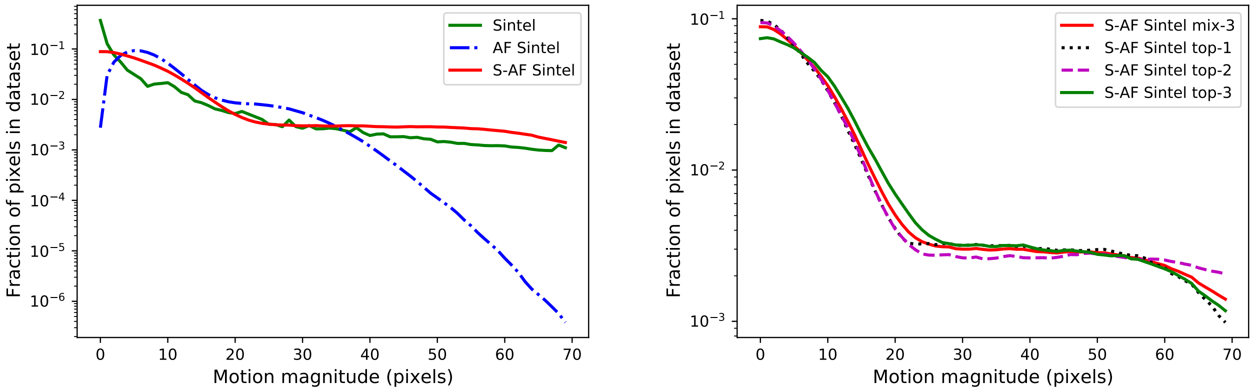

Figure C.1. **Histogram of motion magnitude.** We include the motion statistics of the generated flow field by Self-AutoFlow and AutoFlow. Interestingly, the Self-AutoFlow data focuses more on small motion compared to AutoFlow. Also, the statistics of Self-AutoFlow are closer to the statistics of Sintel data. In addition, we show the statistics of individual S-AF dataset and their mixed results.

C.1. AEPE versus self-supervised losses of SMURF and S-AF

We calculate the self-supervised losses and the AEPE on the target datasets for SMURF and S-AF models in Tab. C.1. The losses and errors are computed for the full target datasets and we report the average. In most cases, the SMURF models have lower self-supervised losses compared to the S-AF models, while the S-AF models have lower AEPE.

Although the self-supervised metric is highly correlated with the AEPE, optimizing it *directly* by backpropagation to the model might lead to a model with lower self-supervised loss and higher EPE. In contrast, our Self-AutoFlow method uses the self-supervised loss *indirectly* to assess the quality of a generated dataset, which results in a model with higher self-supervised loss and lower EPE. To conclude, Self-AutoFlow is a good strategy for using self-supervised losses.

Table C.1. **Self-supervised losses versus AEPE.** We compute the photometric, distillation and smoothness loss averaged on the training set. We show that our S-AF model which uses the self-supervised loss *indirectly* to assess the quality of a generated dataset results in a model with higher self-supervised loss and lower EPE.

	Sintel Final [1]					KITTI 2015 [2]				
Method	$\mathcal{L}_{\text{photo}} \downarrow$	$\mathcal{L}_{\text{distill}} \downarrow$	$\mathcal{L}_{\text{smooth}} \downarrow$	$\mathcal{L}_{\text{total}} \downarrow$	*AEPE*↓	$\mathcal{L}_{\text{photo}} \downarrow$	$\mathcal{L}_{\text{distill}} \downarrow$	$\mathcal{L}_{\text{smooth}} \downarrow$	$\mathcal{L}_{\text{total}} \downarrow$	*AEPE* ↓
SMURF Chairs [4]	2.20	0.70	**0.013**	2.92	3.35	2.61	**1.05**	**0.0046**	**3.67**	7.94
S-AF	2.20	**0.44**	0.017	**2.66**	**2.57**	**2.54**	1.24	0.0052	3.78	**4.28**
+SS Sintel/KITTI										
SMURF [4]	**2.17**	0.67	**0.012**	**2.86**	2.80	2.54	**0.80**	0.0046	**3.35**	2.01
S-AF	2.20	**0.65**	0.013	2.87	**2.40**	**2.49**	0.87	**0.0043**	3.37	**1.94**

D. Additional Results

D.1. Visualization of keypoint propagation on BADJA

We visualize the keypoint propagation results on BADJA sequences by SMURF and our S-AF in Fig. D.1. The keypoints correctly propagated are marked as a dot, and the keypoints with the wrong predicted trajectory are marked as a cross. Compared the results without self-supervised fine-tuning, S-AF tracks the three keypoints on the back (gray), left ear (red), and right ear (brown) correctly. On the other hand, SMURF loses the keypoint on the right ear (brown) since the second frame and loses the keypoint on the back (gray) since the third frame in the dog sequence. As for the models with self-supervised fine-tuning, we show the keypoint in the horsejump-low sequence. S-AF correctly predicts the trajectory of the purple keypoint on the tail, while SMURF loses it since the second frame.

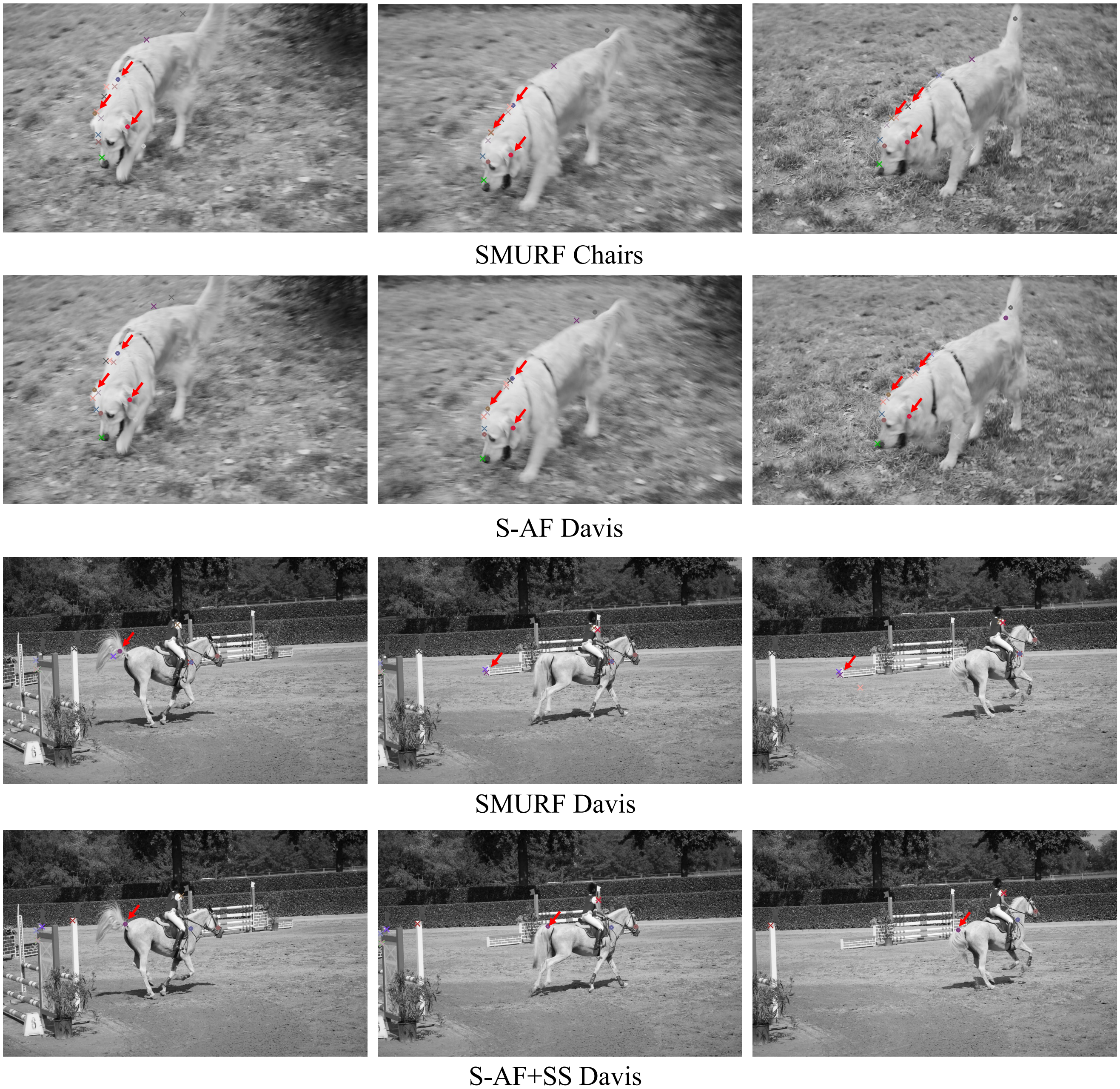

Figure D.1. **Visual results of keypoints on BADJA.** The keypoints correctly tracked are marked as a dot, and the keypoints with the wrong trajectory are marked as a cross. Compared the results without self-supervised fine-tuning in (a), S-AF tracks the three keypoints (gray, red, and brown) correctly, while SMURF loses the brown keypoint in the second frame and the gray keypoint in the third frame. As for the results with self-supervised fine-tuning, S-AF correctly tracks the purple keypoint on the tail, and SMURF loses it since the second frame.

D.2. Benchmark results

We provide the screenshots of both models on the public benchmarks in Fig. D.2 and Fig. D.3. As listed in Tab. 3, we provide the detailed performance of our S-AF+SS models on public benchmarks in Tab. D.1. The S-AF+SS model is more accurate in most cases while less accurate on unmatch and s0-10 for Sintel benchmark, and on F1-fg all for KITTI benchmark compared to SMURF. As shown in Tab. 5, we show the detailed performance of the supervised fine-tuning model RAFT-S-AF in Tab. D.2. For KITTI benchmark, RAFT-S-AF is more accurate in most cases while less accurate for F1-fg. RAFT-S-AF is more accurate for all cases for Sintel Clean and Sintel Final.

SfM-PM [116]	2.910	1.016	18.357	2.797	0.756	0.479	0.559	1.732	17.431	Visualize Results
FlowFields++ [117]	2.943	0.850	20.027	2.550	0.603	0.403	0.560	1.859	17.401	Visualize Results
GCA-Net [118]	2.947	1.032	18.585	2.900	0.830	0.456	0.602	1.645	17.753	Visualize Results
LiteFlowNet3 [119]	2.994	1.148	18.077	3.000	0.985	0.498	0.559	1.670	18.302	Visualize Results
ricom20201202 [120]	3.002	1.150	18.125	2.915	0.911	0.609	0.723	1.949	16.869	Visualize Results
risc [121]	3.005	1.150	18.147	2.930	0.908	0.607	0.721	1.957	16.885	Visualize Results
RAFT-SA [122]	3.026	1.122	18.578	2.577	0.988	0.579	0.410	1.191	20.477	Visualize Results
LiteFlowNet3-S [123]	3.028	1.173	18.182	3.079	0.996	0.527	0.574	1.646	18.566	Visualize Results
RICBCDN [124]	3.080	1.212	18.319	3.241	0.945	0.577	0.724	2.077	17.197	Visualize Results
FlowFields+ [125]	3.102	0.820	21.718	2.340	0.616	0.373	0.593	1.865	18.549	Visualize Results
DIP-Flow [126]	3.103	0.881	21.227	2.574	0.681	0.419	0.548	1.801	18.979	Visualize Results
PST [127]	3.110	0.942	20.809	2.759	0.664	0.378	0.635	2.069	17.919	Visualize Results
MPIF [128]	3.111	1.134	19.218	3.070	0.939	0.523	0.616	1.980	18.220	Visualize Results
LSM_FLOW_RVC [129]	3.142	1.395	17.394	2.557	1.091	0.873	0.361	1.202	21.652	Visualize Results
SMURF [130]	3.152	1.550	16.233	3.141	1.310	0.858	0.398	1.371	21.152	Visualize Results

(a) Sintel Clean

RAFT+ConvUp [93]	3.642	1.661	19.796	3.372	1.258	1.096	0.732	2.054	21.920	Visualize Results
CVPR-1235 [94]	3.649	1.912	17.818	3.857	1.576	1.144	0.823	2.965	19.311	Visualize Results
DCVNet [95]	3.655	1.986	17.243	3.822	1.556	1.296	0.772	2.409	20.937	Visualize Results
STaRFlow [96]	3.707	1.838	18.946	3.618	1.439	1.122	0.744	2.018	22.491	Visualize Results
C-RAFT_RVC [97]	3.795	1.919	19.089	3.591	1.660	1.236	0.674	2.369	22.723	Visualize Results
Flow1D [98]	3.806	1.949	18.946	3.604	1.756	1.394	0.738	2.479	22.221	Visualize Results
RAFT-SA [99]	3.977	1.889	21.005	3.995	1.575	1.077	0.889	2.234	23.527	Visualize Results
IOFPL-CVr8-ft [100]	4.014	1.906	21.194	3.246	1.418	1.374	0.656	1.905	25.767	Visualize Results
ADW-Net [101]	4.017	1.951	20.855	3.720	1.472	1.308	0.818	2.442	23.694	Visualize Results
DistillFlow+ft [102]	4.095	2.031	20.934	4.300	1.666	1.236	0.673	2.448	25.068	Visualize Results
ScopeFlow [103]	4.098	1.999	21.214	4.028	1.689	1.180	0.725	2.589	24.477	Visualize Results
ARFlow-mv-ft [104]	4.142	2.082	20.937	4.056	1.707	1.300	0.706	2.366	25.475	Visualize Results
vcn+MSDRNet [105]	4.143	1.999	21.621	3.932	1.637	1.266	0.763	2.387	25.165	Visualize Results
LSM_FLOW_RVC [106]	4.150	2.018	21.531	3.470	1.600	1.478	0.606	1.840	27.323	Visualize Results
MaskFlownet [107]	4.172	2.048	21.494	3.783	1.745	1.310	0.592	2.389	26.253	Visualize Results
SMURF [108]	4.183	2.138	20.861	4.198	1.744	1.296	0.740	2.302	25.819	Visualize Results

(b) Sintel Final

100	VCN		code	5.83 %	8.66 %	6.30 %	100.00 %	0.18 s	Titan X Pascal	
G. Yang and D. Ramanan: Volumetric Correspondence Networks for Optical Flow. NeurIPS 2019.										
101	Stereo expansion		code	5.83 %	8.66 %	6.30 %	100.00 %	2 s	GPU @ 2.5 Ghz (Python)	
G. Yang and D. Ramanan: Upgrading Optical Flow to 3D Scene Flow through Optical Expansion. CVPR 2020.										
102	Binary TTC			5.84 %	8.67 %	6.31 %	100.00 %	2 s	GPU @ 1.0 Ghz (Python)	
A. Badki, O. Gallo, J. Kautz and P. Sen: Binary TTC: A Temporal Geofence for Autonomous Navigation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2021.										
103	MonoComb			5.84 %	8.67 %	6.31 %	100.00 %	0.58 s	RTX 2080 Ti	
R. Schuster, C. Unger and D. Stricker: MonoComb: A Sparse-to-Dense Combination Approach for Monocular Scene Flow. ACM Computer Science in Cars Symposium (CSCS) 2020.										
104	HD^3-Flow		code	6.05 %	9.02 %	6.55 %	100.00 %	0.10 s	NVIDIA Pascal Titan XP	
Z. Yin, T. Darrell and F. Yu: Hierarchical Discrete Distribution Decomposition for Match Density Estimation. CVPR 2019.										
105	PRSM		code	5.33 %	13.40 %	6.68 %	100.00 %	300 s	1 core @ 2.5 Ghz (C/C++)	
C. Vogel, K. Schindler and S. Roth: 3D Scene Flow Estimation with a Piecewise Rigid Scene Model. ijcv 2015.										
106	RAFT-SA		code	5.90 %	11.09 %	6.76 %	100.00 %	1 s	1 core @ 2.5 Ghz (C/C++)	
107	MaskFlownet-S		code	6.53 %	8.21 %	6.81 %	100.00 %	0.03 s	NVIDIA TITAN Xp	
S. Zhao, Y. Sheng, Y. Dong, E. Chang and Y. Xu: MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2020.										
108	ScopeFlow		code	6.72 %	7.36 %	6.82 %	100.00 %	-1 s	Nvidia GPU	
A. Bar-Haim and L. Wolf: ScopeFlow: Dynamic Scene Scoping for Optical Flow. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2020.										
109	SMURF		code	6.04 %	10.75 %	6.83 %	100.00 %	.2 s	1 core @ 2.5 Ghz (C/C++)	
A. Stone, D. Maurer, A. Ayvaci, A. Angelova and R. Jonschkowski: SMURF: Self-Teaching Multi-Frame Unsupervised RAFT With Full-Image Warping. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021.										
110	RAFT-VM			6.49 %	8.65 %	6.85 %	100.00 %	0.4 s	GPU @ 2.5 Ghz (C/C++)	
111	OSF+TC			5.76 %	13.31 %	7.02 %	100.00 %	50 min	1 core @ 2.5 Ghz (C/C++)	
M. Neoral and J. Šochman: Object Scene Flow with Temporal Consistency. 22nd Computer Vision Winter Workshop (CVWW) 2017.										
112	IRR-full			6.99 %	7.57 %	7.09 %	100.00 %	0.2 s	1 core @ 2.5 Ghz (Python)	

(c) KITTI

Figure D.2. **Screenshot of S-AF+SS on public benchmark.** Our method was temporarily named as RAFT-SA.

GMA [19]	1.388	0.582	7.963	1.537	0.461	0.278	0.331	0.963	7.662	Visualize Results
GMFlowNet [20]	1.390	0.520	8.486	1.275	0.395	0.293	0.314	0.991	7.698	Visualize Results
GMA+LCT-Flow [21]	1.408	0.525	8.611	1.428	0.404	0.251	0.279	0.876	8.299	Visualize Results
AGF-Flow3 [22]	1.409	0.525	8.618	1.433	0.403	0.250	0.278	0.878	8.303	Visualize Results
RFPM [23]	1.411	0.494	8.884	1.335	0.400	0.221	0.273	0.879	8.345	Visualize Results
RAFT-OCTC [24]	1.419	0.541	8.574	1.455	0.442	0.242	0.301	0.940	8.118	Visualize Results
RAFT-SA+ [25]	1.421	0.535	8.654	1.495	0.451	0.207	0.260	0.896	8.460	Visualize Results
GMA-FS [26]	1.430	0.602	8.171	1.579	0.470	0.263	0.333	0.977	7.961	Visualize Results
AGFlow [27]	1.431	0.559	8.541	1.501	0.452	0.261	0.319	0.963	8.075	Visualize Results
DIP [28]	1.435	0.519	8.919	1.102	0.407	0.312	0.336	0.754	8.546	Visualize Results
CRAFT [29]	1.441	0.611	8.204	1.574	0.552	0.249	0.311	0.991	8.131	Visualize Results
ErrorMatch-GMA [30]	1.446	0.584	8.472	1.503	0.483	0.280	0.311	0.935	8.314	Visualize Results
GMA-base [31]	1.450	0.591	8.440	1.532	0.470	0.280	0.321	0.951	8.251	Visualize Results

(a) Sintel Clean

RAFT+NCUP [48]	2.692	1.323	13.854	3.139	1.086	0.636	0.635	1.844	14.949	Visualize Results
RAFT-it+_RVC [49]	2.696	1.317	13.929	2.486	0.929	0.839	0.440	1.456	16.880	Visualize Results
ERRN [50]	2.701	1.348	13.744	3.164	1.109	0.666	0.647	1.854	14.943	Visualize Results
CVE-RAFT [51]	2.707	1.227	14.776	2.942	1.054	0.617	0.580	1.726	15.634	Visualize Results
AGF-Flow3 [52]	2.733	1.217	15.105	2.419	0.912	0.737	0.463	1.440	17.133	Visualize Results
GMA+LCT-Flow [53]	2.734	1.218	15.103	2.419	0.914	0.738	0.465	1.441	17.131	Visualize Results
submission5367 [54]	2.742	1.282	14.656	3.027	1.110	0.644	0.562	1.743	15.980	Visualize Results
RAFT-SA+ [55]	2.749	1.375	13.943	2.634	1.132	0.872	0.469	1.545	16.967	Visualize Results
L2L-Flow-ext-warm [56]	2.780	1.319	14.697	3.098	1.145	0.637	0.656	1.879	15.502	Visualize Results
LCT-Flow2 [57]	2.781	1.349	14.465	2.720	0.989	0.895	0.620	1.582	16.405	Visualize Results
RAFT-FS [58]	2.785	1.341	14.557	3.114	1.104	0.649	0.681	1.850	15.487	Visualize Results
EMD-L [59]	2.790	1.260	15.258	2.629	0.981	0.837	0.537	1.595	16.856	Visualize Results
MFR [60]	2.801	1.380	14.385	3.075	1.112	0.772	0.674	1.829	15.703	Visualize Results
RAFTwarm+AOIR [61]	2.813	1.371	14.565	3.088	1.099	0.727	0.603	1.781	16.271	Visualize Results

(b) Sintel Final

4	RigidMask+ISF		code	2.63 %	7.85 %	3.50 %	100.00 %	3.3 s	GPU @ 2.5 Ghz (Python)	
G. Yang and D. Ramanan: Learning to Segment Rigid Motions from Two Frames. CVPR 2021.										
5	TPCV+RAFT3D			2.48 %	10.19 %	3.76 %	100.00 %	0.2 s	1 core @ 2.5 Ghz (C/C++)	
6	RAFT-it+_RVC			3.62 %	**5.33 %**	3.90 %	100.00 %	0.14 s	1 core @ 2.5 Ghz (Python)	
7	RAFT-OCTC			3.72 %	5.39 %	4.00 %	100.00 %	0.2 s	GPU @ 2.5 Ghz (Python)	
J. Jeong, J. Lin, F. Porikli and N. Kwak: Imposing Consistency for Optical Flow Estimation (Qualcomm AI Research). CVPR 2022.										
8	SF2SE3		code	3.17 %	8.79 %	4.11 %	100.00 %	2.7 s	GPU @ >3.5 Ghz (Python)	
L. Sommer, P. Schröppel and T. Brox: SF2SE3: Clustering Scene Flow into SE (3)-Motions via Proposal and Selection. DAGM German Conference on Pattern Recognition 2022.										
9	RAFT-CF-PL3			3.80 %	5.65 %	4.11 %	100.00 %	0.05 s	GPU @ 2.5 Ghz (Python)	
Z. Zhang, P. Ji, N. Bansal, C. Cai, Q. Yan, X. Xu and Y. Xu: CLIP-FLow: Contrastive Learning by semi-supervised Iterative Pseudo labeling for Optical Flow Estimation. 2022.										
10	RAFT-S-AF		code	3.86 %	5.38 %	4.12 %	100.00 %	1 s	1 core @ 2.5 Ghz (C/C++)	
11	MS_RAFT+_corr_RVC		code	3.83 %	5.71 %	4.15 %	100.00 %	0.65 s	GPU @ 2.5 Ghz (Python + C/C++)	
A. Jahedi, M. Luz, L. Mehl, M. Rivinius and A. Bruhn: High Resolution Multi-Scale RAFT. Robust Vision Challenge 2022, arXiv preprint arXiv:2210.16900 2022. A. Jahedi, L. Mehl, M. Rivinius and A. Bruhn: Multi-Scale Raft: Combining Hierarchical Concepts for Learning-Based Optical Flow Estimation. IEEE International Conference on Image Processing (ICIP) 2022.										
12	MS_RAFT+_RVC			3.89 %	5.67 %	4.19 %	100.00 %	0.65 s	GPU @ 2.5 Ghz (Python + C/C++)	
13	DIP		code	3.86 %	5.96 %	4.21 %	100.00 %	0.15 s	1 core @ 2.5 Ghz (Python)	
Z. Zheng, N. Nie, Z. Ling, P. Xiong, J. Liu, H. Wang and J. Li: DIP: Deep Inverse Patchmatch for High- Resolution Optical Flow. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022.										
14	RAFT-3D			3.39 %	8.79 %	4.29 %	100.00 %	2 s	GPU @ 2.5 Ghz (Python + C/C++)	
Z. Teed and J. Deng: RAFT-3D: Scene Flow using Rigid-Motion Embeddings. arXiv preprint arXiv:2012.00726 2020.										
15	RAFT-it			4.11 %	5.34 %	4.31 %	100.00 %	0.1 s	GPU @ 2.5 Ghz (Python)	
16	RCA-Flow			3.96 %	6.21 %	4.33 %	100.00 %	0.16 s	1 core @ 2.5 Ghz (Python)	

(c) KITTI

Figure D.3. **Screenshot of the supervised fine-tuning results of S-AF on public benchmark.** Our method was temporarily named as RAFT-SA+ and RAFT-S-AF.

Table D.1. Detailed performance of S-AF+SS on public benchmark.

Model	all	match	unmatch	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
SMURF	3.15	1.55	16.23	3.14	1.31	0.86	0.40	1.37	21.15
S-AF+SS	3.03	1.12	18.58	2.58	0.99	0.58	0.41	1.19	20.48

(a) Sintel Clean

Model	all	match	unmatch	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
SMURF	4.18	2.14	20.86	4.20	1.74	1.30	0.74	2.30	25.82
S-AF+SS	3.98	1.89	21.01	4.00	1.58	1.08	0.89	2.23	23.53

(b) Sintel Final

Model	All			Occ		
	Fl-bg	**Fl-fg**	**Fl-all**	**Fl-bg**	**Fl-fg**	**Fl-all**
SMURF	6.04 %	10.75 %	6.83 %	4.46 %	8.86 %	5.26 %
S-AF+SS	5.90 %	11.09 %	6.76 %	4.41 %	8.67 %	5.18 %

(c) KITTI

Table D.2. Detailed performance of the supervised fine-tuning results of S-AF on public benchmark.

Model	all	match	unmatch	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
RAFT-it	1.55	0.61	9.24	1.66	0.51	0.27	0.29	0.97	9.26
RAFT-S-AF	1.42	0.54	8.65	1.50	0.45	0.21	0.26	0.90	8.46

(a) Sintel Clean

Model	all	match	unmatch	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
RAFT-it	2.90	1.41	15.03	2.81	1.16	0.88	0.51	1.70	17.62
RAFT-S-AF	2.75	1.38	13.94	2.63	1.13	0.87	0.47	1.55	16.97

(b) Sintel Final

Model	All			Occ		
	Fl-bg	**Fl-fg**	**Fl-all**	**Fl-bg**	**Fl-fg**	**Fl-all**
RAFT-it	4.11 %	5.34 %	4.31 %	2.68 %	2.77 %	2.70 %
RAFT-S-AF	3.86 %	5.38 %	4.12 %	2.52 %	2.86 %	2.59 %

(c) KITTI

References

[1] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 1, 2, 3

[2] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3D estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015. 1, 2, 3

[3] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 1

[4] Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski. SMURF: Self-teaching multi-frame unsupervised raft with full-image warping. In *CVPR*, 2021. 1, 2, 3

[5] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 1