SEA-RAFT: Simple, Efficient, Accurate RAFT for Optical Flow

Yihan Wang, Lahav Lipson, and Jia Deng

Department of Computer Science, Princeton University
{yw7685, llipson, jiadeng}@princeton.edu

Abstract. We introduce SEA-RAFT, a more simple, efficient, and accurate RAFT for optical flow. Compared with RAFT, SEA-RAFT is trained with a new loss (mixture of Laplace). It directly regresses an initial flow for faster convergence in iterative refinements and introduces rigid-motion pre-training to improve generalization. SEA-RAFT achieves state-of-the-art accuracy on the Spring benchmark with a 3.69 endpoint-error (EPE) and a 0.36 1-pixel outlier rate (1px), representing 22.9% and 17.8% error reduction from best published results. In addition, SEA-RAFT obtains the best cross-dataset generalization on KITTI and Spring. With its high efficiency, SEA-RAFT operates at least 2.3× faster than existing methods while maintaining competitive performance. The code is publicly available at https://github.com/princeton-vl/SEA-RAFT.

1 Introduction

Optical flow is a fundamental task in low-level vision and aims to estimate per-pixel 2D motion between video frames. It is useful for various downstream tasks including action recognition [39, 49, 67], video in-painting [10, 22, 60], frame interpolation [15, 27, 61], 3D reconstruction and synthesis [33, 69].

Although traditionally formulated as an optimization problem [5, 13, 62], almost all recent methods are based on deep learning [6,8,11,14,24,29,42–45,48,50, 54–57,63,66,68]. In particular, many state-of-the-art methods [14,29,43,44,50,66] have adopted architectures based on RAFT [50], which uses a recurrent network to iteratively refine a flow field.

In this paper, we introduce SEA-RAFT, a new variant of RAFT that is more efficient and accurate. When compared against all existing approaches, SEA-RAFT has the best accuracy-efficiency Pareto frontier (Fig. 1):

- *Accuracy*: On Spring [35], SEA-RAFT achieves a new state of the art, outperforming the next best by a large margin: 18% error reduction on 1px-outlier rate (3.686 vs. 4.482) and 24% error reduction on endpoint-error (0.363 vs. 0.471). On Sintel [3] and KITTI [36], it outperforms all other methods that have similar computational costs.
- *Efficiency*: On each benchmark tested, SEA-RAFT runs at least 2.3× faster than existing methods that have comparable accuracy. Our smallest model,

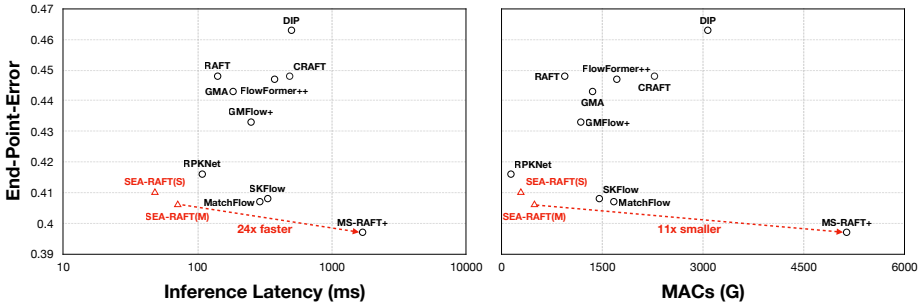

Fig. 1: Zero-shot performance of SEA-RAFT and existing methods on the Spring [35] training split. Latency is measured on an RTX3090 with a batch size of 1 and input resolution 540 × 960. SEA-RAFT has an accuracy close to the best one achieved by MS-RAFT+ [19] but is 11× smaller and 24× faster.

which still outperforms all other methods on Spring, can run at 21fps when processing 1080p images on an RTX3090, 3× faster than the original RAFT.

We achieve this by introducing a combination of improvements over the original RAFT:

- *Mixture of Laplace Loss:* Instead of the standard L_1 loss, we train the network to predict parameters of a mixture of Laplace distributions to maximize the log-likelihood of the ground truth flow. As we will demonstrate, this new loss reduces overfitting to ambiguous cases and improves generalization.
- *Directly Regressed Initial Flow:* Instead of initializing the flow field to zero before iterative refinement, we directly predict the initial flow by reusing the existing context encoder and feeding it the stacked input frames. This simple change introduces minimal overhead but is surprisingly effective in reducing the number of iterations and improving efficiency.
- *Rigid-Flow Pre-Training:* We find that pre-training on TartanAir [52], which can significantly improve generalization, despite the limited diversity of flow, which is induced purely by camera motion in a static scene.

These improvements are novel in the context of RAFT-style methods for optical flow. Moreover, they are orthogonal to the improvements proposed in existing RAFT-style methods, which focus on replacing certain blocks with newer designs, such as replacing convolutional blocks with transformers.

Besides the main improvements above, SEA-RAFT also incorporates architectural changes that greatly simplify the original RAFT. In particular, we find that certain custom designs of the original RAFT are unnecessary and can be replaced with standard off-the-shelf modules. For example, the original feature encoder and context encoder were custom-designed and must use different normalization layers for stable training; we replaced each with a standard ResNet. In addition, we replace the original convolutional GRU with a simple RNN con-

sisting entirely of ConvNext blocks. Such simplifications make it easy for SEA-RAFT to incorporate new neural building blocks and scale to larger datasets.

We perform extensive experiments to evaluate SEA-RAFT on standard benchmarks including Spring, Sintel, and KITTI. We also validate the effectiveness of our improvements through ablation studies.

2 Related Works

Estimating Optical Flow Classical approaches treated optical flow as an optimization problem that maximizes visual similarity between corresponding pixels, with strong regularization. [5, 13, 62]. Current methods [6, 9, 14, 16–19, 24, 30, 31, 42–45, 48, 50, 54–57, 65, 66, 68] are mostly based on deep learning. FlowNets [9, 17] regarded optical flow as a dense regression problem and used stacked convolution blocks for prediction. DCNet [58] and PWC-Net [45] introduced 4D cost-volume to explicitly model pixel correspondence. RAFT [50] further combined multi-scale 4D cost-volume with recurrent iterative refinements, achieving large improvements and spawning many follow-ups [14, 19, 30, 31, 43, 44, 48, 66, 68].

Our method is a new variant of RAFT [50] with several improvements including a new loss function, direct regression of initial flow, rigid-flow pre-training, and architectural simplifications. All of these improvements are new compared to existing RAFT variants. In particular, our direct regression of initial flow is new compared to existing efficient RAFT variants [6, 11, 37], which mainly focus on efficient implementations of RAFT modules. This direct regression is a simple change with minimal overhead, but substantially reduces the number of RAFT iterations needed.

Data for Optical Flow FlyingChairs and FlyingThings3D [9,34] are commonly used datasets for optical flow. They provide a large amount of synthetic data but have limited realism. Sintel [3], VIPER [41], Infinigen [40], and Spring [35] are more realistic, using open-source 3D animations, games or procedurally generated scenes. Besides synthetic data, Middlebury, KITTI, and HD1K [1, 12, 23, 36] provide annotations for real-world image pairs. These datasets are limited in both quantity and diversity due to the difficulty of accurately annotating optical flow in the real world. To leverage more data, several methods [8, 42, 54, 55] pre-train their models on different tasks. MatchFlow [8] pre-trains on geometric image matching (GIM) using MegaDepth [26]. Croco-Flow [54, 55], DDVM [42], and Flowformer++ [43] pre-train on unlabeled data. We pre-train SEA-RAFT on rigid flow using TartanAir [52]. Though TartanAir [52] has been used in other methods such as DDVM [42] and CroCo-Flow [54, 55], our adoption of rigid-flow pre-training is new in the context of RAFT-style methods.

Predicting Probability Distributions Predicting probability distributions is a common practice in computer vision [2, 4, 25, 32, 47, 51, 53, 64]. In tasks closely related to optical flow such as keypoint matching [4, 47, 51, 64], the variance of the probability distribution reflects uncertainty of predictions and therefore is useful for many applications. For example, LoFTR [47] filters out uncertain matching pairs. Aspanformer [4] adjusts the look-up radius based on uncertainty.

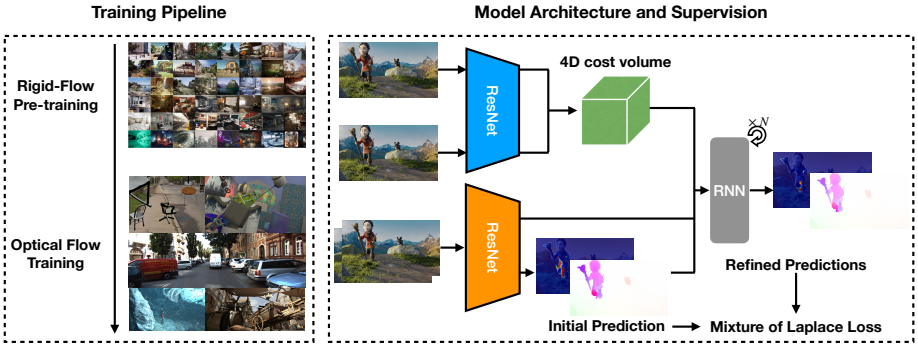

Fig. 2: Compared with RAFT [50], SEA-RAFT introduces (1) rigid-flow pre-training, (2) mixture of Laplace loss, and (3) direct regression of initial flow.

To handle the ambiguity caused by heavy occlusion, SEA-RAFT predicts a mixture of Laplace (MoL) distribution. Although MoL has been used in keypoint matching methods such as PDC-Net+ [51], our use of MoL is new in the context of RAFT-style methods. In addition, our formulation is different in that we require one mixture component to have a constant variance, making it equivalent to the L_1 loss that aligns better with the optical flow evaluation metrics. This difference is crucial for achieving competitive performance in optical flow, where every pixel needs accurate correspondence, unlike keypoint matching, where a subset of reliable matches suffices.

3 Method

In this section, we first describe the iterative refinement in RAFT and then introduce the improvements that lead to SEA-RAFT.

3.1 Iterative refinement

Given two adjacent RGB frames, RAFT predicts a field of pixel-wise 2D vectors through iterative refinement that consists of two parts: (1) feature and context encoders, which transform images into lower-resolution dense features, and (2) an RNN unit, which iteratively refines the predictions.

Given two images $I_1, I_2 \in \mathbb{R}^{H\times W\times 3}$, the feature encoder F takes I_1, I_2 as inputs separately and outputs a lower-resolution feature $F(I_1), F(I_2) \in \mathbb{R}^{h\times w\times D}$. The context encoder C takes source image I_1 as input and outputs a context feature $C(I_1) \in \mathbb{R}^{h\times w\times D}$. A multi-scale 4D correlation volume $\{V_k\}$ is then built with the features from feature encoder F:

$$V_k = F(I_1) \circ \texttt{AvgPool}(F(I_2), 2^k)^\top \in \mathbb{R}^{h\times w\times \frac{h}{2^k}\times \frac{w}{2^k}},$$

where $\circ$ represents the correlation operator, which computes similarities (as dot products of feature vectors) between all pairs of pixels across two feature maps.

Several works [18, 19] have explored the optimal choices of the number of levels in the cost volume(k) and the feature resolution (h, w). In SEA-RAFT, we simply follow the original setting in RAFT [50]: $(h, w) = \frac{1}{8}(H, W), k = 4$.

RAFT iteratively refines a flow prediction μ. Initially, μ is set to be all zeros. Each refinement step uses the current flow prediction μ to fetch a D_M-dim motion feature M from the multi-scale correlation volume $\{V_k\}$ with a look-up radius r:

$$M = \texttt{MotionEncoder}(\texttt{LookUp}(\{V_k\}, \mu, r)) \in \mathbb{R}^{h \times w \times D_M},$$

where the `Lookup` operator returns a motion feature vector for each pixel in I_1, consisting of similarities between the pixel in I_1 and its current correspondence's neighboring pixels in I_2 within the radius r. The motion feature vector is further transformed by a motion encoder.

Existing works [4, 11, 21] have explored dynamic radius and look-up when obtaining the motion features from $\{V_k\}$. For simplicity of design, SEA-RAFT follows the original RAFT and sets the look-up radius $r = 4$ to a fixed constant. The motion feature M is fed into the RNN cell along with hidden state h and context feature $C(I_1)$. From the new hidden state h', the residual flow $\Delta\mu$ is regressed by a 2-layer `FlowHead`:

$$h' = \texttt{RNN}(h, M, C(I_1))$$
$$\Delta\mu = \texttt{FlowHead}(h')$$

Methods using RAFT-Style iterative refinement [14, 50] usually need many iterations: 12 in training and as many as 32 in inference. As a result, RNN-based iterative refinement is a significant bottleneck in latency. Though there have been attempts [6, 11] to reduce the number of iterations, the performance drastically drops with fewer iterations. In contrast, SEA-RAFT only needs 4 iterations in training and up to 12 iterations in inference to achieve competitive performance.

3.2 Mixture-of-Laplace Loss

Most prior works are supervised using an endpoint-error loss on all pixels. However, optical flow training data often contains ambiguous, unpredictable samples, which can dominate this loss empirically.

Ambiguous Cases Ambiguous cases of optical flow can arise with heavy occlusion Fig. 3. While in many cases the motion of occluded pixels can be predicted, sometimes the ambiguity can be too large to predict a single outcome. We examined 10 samples with the highest endpoint-error in the training and validation sets of FlyingChairs [9] and found that ambiguous cases dominate the error.

Review of Probabilistic Regression Prior works for image-matching have proposed probabilistic losses to enable their model to express aleatoric or epistemic uncertainty [4,47,51,51,53,55,64]. These approaches regress the parameters of the probabilistic model and maximize the log-likelihood of the ground truth during training.

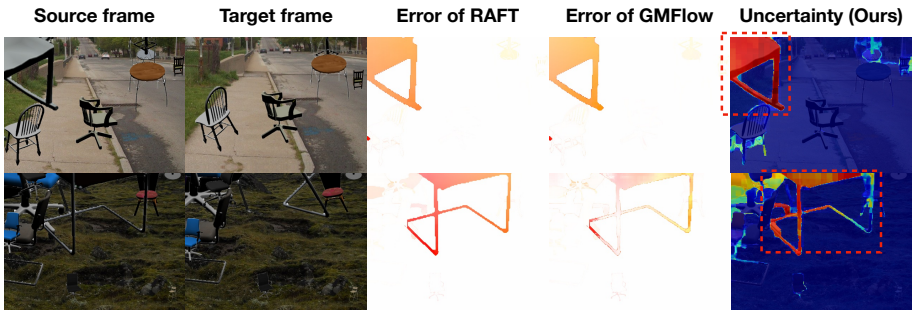

Fig. 3: Ambiguous cases can occur frequently in training data where flow is unpredictable due to occlusion. Such cases can dominate the L_1 loss (shown as an error map) used by current methods [50, 56]. Our new training loss allows the model to account for such uncertainty.

Given an image pair $\{I_1, I_2\}$ and the flow ground truth μ_{gt}, the training loss is

$$\mathcal{L}_{prob} = -\log p_\theta(\mu = \mu_{gt} | I_1, I_2)$$

where the probability density function p_θ is parameterized by the network. Prior work has formulated p_θ as a Gaussian or a Laplace distribution with a predicted mean and variance. For example, we can formulate a naive version of probabilistic regression by assuming: (1) p_θ is Laplace with mean $\mu \in \mathbb{R}^{H \times W \times 2}$ and scale $b \in \mathbb{R}^{H \times W \times 1}$ predicted by the network, (2) the flow distribution is pixel-wisely independent, and (3) the x-direction flow and the y-direction flow are independent but share the same scale parameter b:

$$\mathcal{L}_{Lap} = \frac{1}{HW} \sum_u \sum_v (\log 2b(u,v) + \frac{\|\mu_{gt}(u,v) - \mu(u,v)\|_1}{2b(u,v)}) \tag{1}$$

where u, v are indices to the pixels. The Laplace loss can be regarded as an extended version of L_1 loss with an extra penalty term b. During inference, μ represents the flow prediction, and the scale factor b provides an estimation of uncertainty. However, we find this naive probabilistic regression does not work well on optical flow, which has also been pointed out by prior work [64].

Mixture of Laplace One reason that naive probabilistic regression performs poorly is numerical instability as the loss contains a log term. To address this issue, we regress $b(u, v)$ directly in log-space. This approach makes training more stable compared to previous approaches which clamp b to $[\epsilon, \infty)$, where ϵ is a small positive number.

Another reason that naive probabilistic regression performs poorly is that it deviates from the standard endpoint-error metric, which only cares about the L_1 difference, but not the uncertainty estimation. Thus, we propose to use a mixture of two Laplace distributions: one for ordinary cases, and the other for

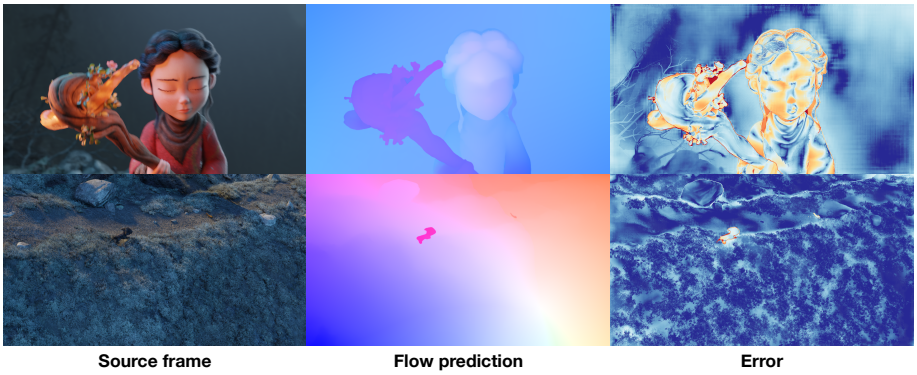

Fig. 4: Visualization on Spring [35] test set.

ambiguous cases, with mixing coefficient $\alpha \in [0, 1]$:

$$MixLap(x; \alpha, \beta_1, \beta_2, \mu) = \alpha \cdot \frac{e^{-\frac{|x-\mu|}{e^{\beta_1}}}}{2e^{\beta_1}} + (1-\alpha) \cdot \frac{e^{-\frac{|x-\mu|}{e^{\beta_2}}}}{2e^{\beta_2}}$$

Intuitively, at each pixel, we want the first component of the mixture to be aligned with the endpoint-error metric, and the second component to account for ambiguous cases. To explicitly enforce this, we fix $\beta_1 = 0$, such that the network is encouraged to optimize for the L1 loss when possible. This leads to the following Mixture-of-Laplace (MoL) loss:

$$\mathcal{L}_{MoL} = -\frac{1}{2HW} \sum_u \sum_v \sum_{d \in \{x,y\}} \log \left[MixLap(\mu_{gt}(u,v)_d; \alpha(u,v), 0, \beta_2(u,v), \mu(u,v)_d) \right] \quad (2)$$

where d indexes the axe of the flow vector (the x direction or y direction).

The free parameters α, β_2, μ of $\mathcal{L}_{MoL}$ are predicted by the network. Intuitively, a higher α means the flow prediction of this pixel is more "ordinary" instead of "ambiguous". Mathematically, a higher α makes $\mathcal{L}_{moL}$ behave like an L_1 loss. In Sec. 4.3, we then show that this property leads to better accuracy.

Note that though the mixture model has been used in keypoint matching [4, 47, 51], its application to optical flow requires a different formulation because the goal is substantially different. In keypoint matching, the goal is to identify a *subset* of reliable matches for downstream applications such as camera pose estimation. Predicting uncertainty serves to filter out unreliable matches, and there is no explicit penalty for predicting few correspondences. As a result, it is not essential for them to align a mixing component to L_1 loss. In optical flow, we are evaluated on the flow prediction for *every* pixel.

Implementation Details We set an upper bound for β to 10 in the loss to make the training more stable. We also re-predict α and β every update iteration. We

Fig. 5: Visualization on Sintel [3], KITTI [36], and Middlebury [1].

can similarly define the probabilistic sequence loss as:

$$\mathcal{L}_{all} = \sum_{i=1}^{N} \gamma^{N-i} \mathcal{L}_{MoL}^{i} \tag{3}$$

where $\mathcal{L}_{mix}^{i}$ denotes the probabilistic loss in iteration i, N denotes the number of iterations, and $\gamma < 1$ exponentially downweights the early iterations. We empirically observe that our method significantly reduces the number of update iterations needed in inference. In fact, $N = 4$ is sufficient for SEA-RAFT to take first place on the Spring [35] benchmark. We provide detailed ablations in Tab. 4.

3.3 Direct Regression of Initial Flow

RAFT-style iterative refinements [8, 14, 31, 37, 48, 66] typically zero-initialize the flow field. However, zero-initialization may deviate substantially from the ground truth, thus needing many iterations. In SEA-RAFT, we borrow an idea from the FlowNet family of methods [9, 17] to predict an initial estimate of optical flow from the context encoder, given both frames as input. We also predict an associated MoL (see Sec. 3.2).

This simple modification also significantly improves the convergence speed of the iterative refinement framework, allowing one to use fewer iterations during inference. Detailed ablations are shown in Tab. 4.

3.4 Large-Scale Rigid-Flow Pre-Training

Most prior works train on a small number of datasets with limited size, diversity and realism [9, 34]. To improve generationalization, we pre-train SEA-RAFT on TartanAir [52], which provides optical flow annotations between a pair of (non-rectified) stereo cameras. This type of motion field is a special case of optical flow due to viewpoint change in a rigid static scene. Despite its limited motion diversity, it enables SEA-RAFT to train on data with higher realism and scene diversity, leading to better generalization.

3.5 Simplifications

We also provide a few architecture changes that greatly simplify the original RAFT [50]. First, we adopt truncated, ImageNet [7] pre-trained ResNets for the backbones. We also substitute the ConvGRU in RAFT with two ConvNeXt [28] blocks, which we show provides better efficiency and training stability. The detailed ablations of these changes are shown in Tab. 4.

4 Experiments

We evaluate SEA-RAFT on Spring [35], KITTI [12], and Sintel [3]. Following previous works, we also incorporate FlyingChairs [9], FlyingThings [34], and HD1K [23] into our training pipeline. To verify the effectiveness of TartanAir [52] rigid-flow pre-training, we provide the performance gain from it in different settings.

Model Details SEA-RAFT is implemented in PyTorch [38]. There are three different types of SEA-RAFT and we denote them as **SEA-RAFT(S/M/L)**. The only differences among them are the backbone choices and the number of iterations in inference. Specifically, SEA-RAFT(S) uses the first 6 layers of ResNet-18 as the feature/context encoder, and SEA-RAFT(M) uses the first 13 layers of ResNet-34. The pre-trained weights we use are downloaded from torchvision. SEA-RAFT(S) and SEA-RAFT(M) use the same architecture for the recurrent units and keep the number of iterations $N = 4$ in both training and inference. SEA-RAFT(L) can be regarded as an extension based on SEA-RAFT(M): they share the same weights, but SEA-RAFT(L) uses $N = 12$ iterations in inference. Following RAFT [50], we stop the gradient for μ when computing $\mu' = \mu + \Delta\mu$ and only propagate the gradient for residual flow $\Delta\mu$.

Training Details As mentioned in Sec. 3.4, We pre-train SEA-RAFT on TartanAir [52] for 300k steps with a batch size of 32, input resolution 480×640 and learning rate 4×10^{-4}. Similar to RAFT [50], MaskFlowNet [65] and PWC-Net+ [45], we then train our models on FlyingChairs [9] for 100k steps with a batch size of 16, input resolution 368×496, learning rate 2.5×10^{-4} and FlyingThings3D [34] for 120k steps with a batch size of 32, input resolution 432×960, learning rate 4×10^{-4} (denoted as "C+T" following previous works). For the submissions on Sintel [3] benchmark, we fine-tune the model from "C+T" on a mixture of Sintel [3], FlyingThings3D clean pass [34], KITTI [12] and HD1K [23] for 300k steps with a batch size of 32, input resolution 432×960 and learning rate 4×10^{-4} (denoted as "C+T+S+K+H" following previous works). Different from previous methods, we reduce the percentage of Sintel [3] in the mixture dataset, which is usually more than 70% in previous papers. Details will be mentioned in the supplementary material. For KITTI [12] submissions, we fine-tune our models from "C+T+S+K+H" on the KITTI training set for extra 10k steps with a batch size of 16, input resolution 432×960 and learning rate 10^{-4}. For Spring [35] submissions, we fine-tune our models from "C+T+S+K+H" on the

Extra Data	Method	Spring (train)			Spring (test)			
		Fine-tune	1px↓	EPE↓	1px↓	EPE↓	Fl↓	WAUC↑
	PWC-Net [45]	✗	-	-	82.27*	2.288*	4.889*	45.670*
	FlowNet2 [17]	✗	-	-	6.710*	1.040*	2.823*	90.907*
	RAFT [50]	✗	4.788	0.448	6.790*	1.476*	3.198*	90.920*
	GMA [20]	✗	4.763	0.443	7.074*	0.914*	3.079*	90.722*
	RPKNet [37]	✗	4.472	0.416	4.809	0.657	1.756	92.638
	DIP [68]	✗	4.273	0.463	-	-	-	-
	SKFlow [48]	✗	4.521	0.408	-	-	-	-
	GMFlow [56]	✗	29.49	0.930	10.355*	0.945*	2.952*	82.337*
	GMFlow+ [57]	✗	4.292	0.433	-	-	-	-
	Flowformer [14]	✗	4.508	0.470	6.510*	0.723*	2.384*	91.679*
	CRAFT [44]	✗	4.803	0.448	-	-	-	-
	SEA-RAFT(S)	✗	4.077	0.415	-	-	-	-
	SEA-RAFT(M)	✗	**4.060**	**0.406**	-	-	-	-
MegaDepth [26]	MatchFlow(G) [8]	✗	4.504	0.407	-	-	-	-
YouTube-VOS [59]	Flowformer++ [43]	✗	4.482	0.447	-	-	-	-
VIPER [41]	MS-RAFT+ [19]	✗	**3.577**	**0.397**	5.724*	0.643*	2.189*	92.888*
TartanAir [52]	**SEA-RAFT(S)**	✗	4.161	0.410	-	-	-	-
TartanAir [52]	**SEA-RAFT(M)**	✗	3.888	0.406	-	-	-	-
CroCo-Pretrain	CroCoFlow [55]	✓	-	-	4.565	0.498	1.508	93.660
CroCo-Pretrain	Win-Win [24]	✓	-	-	5.371	0.475	1.621	92.270
TartanAir [52]	**SEA-RAFT(S)**	✓	-	-	3.904	0.377	1.389	94.182
TartanAir [52]	**SEA-RAFT(M)**	✓	-	-	**3.686**	**0.363**	**1.347**	**94.534**

Table 1: SEA-RAFT outperforms existing methods on Spring [35] in different settings. * denotes the results submitted by Spring [35] team. By default, all methods have undergone "C+T+S+K+H" training. We list the data used by each method beyond default in the "Extra Data" column. On Spring(test), even our smallest model SEA-RAFT(S) surpasses existing methods by a significant margin. Without fine-tuning on Spring(train), SEA-RAFT outperforms all other methods that do not use extra data.

Spring training set for extra 120k steps with a batch size of 32, input resolution 540×960 and learning rate 4×10^{-4}.

Metrics We adopt the widely used metrics in this study: endpoint-error (EPE), 1-pixel outlier rate (1px), Fl-score and WAUC error. Definitions can be found in [12, 35, 41].

4.1 Results on Spring

Zero-Shot Evaluation We compare several representative existing methods with SEA-RAFT using the checkpoints and configurations for Sintel [3] submission on the Spring [35] training split. For fair comparisons, we remove the test-time optimizations such as tiling in this setting, which will significantly slow down the inference speed. All experiments follow the same downsample-upsample protocol: We first downsample the 1080p images by 2×, do inference, and then bi-linearly upsample the flow field back to 1080p, which ensures the input resolution in inference is similar to their training resolution in "C+T+S+K+H".

Extra Data	Method	Sintel		KITTI	
		Clean↓	Final↓	Fl-epe↓	Fl-all↓
	PWC-Net [45]	2.55	3.93	10.4	33.7
	RAFT [50]	1.43	2.71	5.04	17.4
	GMA [20]	1.30	2.74	4.69	17.1
	SKFlow [48]	1.22	2.46	4.27	15.5
	FlowFormer [14]	1.01	**2.40**	4.09†	14.7†
	DIP [68]	1.30	2.82	4.29	13.7
	EMD-L [6]	**0.88**	2.55	4.12	13.5
	CRAFT [44]	1.27	2.79	4.88	17.5
	RPKNet [37]	1.12	2.45	-	13.0
	GMFlowNet [66]	1.14	2.71	4.24	15.4
	CCMR+ [18]	0.98	2.36	-	**12.9**
	SEA-RAFT(M)	1.21	4.04	4.29	14.2
	SEA-RAFT(L)	1.19	4.11	**3.62**	**12.9**
Youtube-VOS [59]	Flowformer++ [43]	0.90	2.30	3.93†	14.2†
DDVM-Pretrain	DDVM [42]	1.24	2.00	2.19	7.58
DDVM-Pretrain	RAFT [50]	1.27	2.28	2.71	9.16
	GMFlow [56]	1.08	2.48	11.2*	28.7*
TartanAir [52]	GMFlow [56]	-	-	8.70 (-22%)*	24.4 (-15%)*
	SEA-RAFT(S)	1.27	4.32	4.61	15.8
TartanAir	**SEA-RAFT(S)**	1.27	3.74 (-13%)	4.43	15.1
K+H	**SEA-RAFT(S)**	1.32	2.95 (-32%)	-	-
TartanAir+K+H	**SEA-RAFT(S)**	1.30	2.79 (-35%)	-	-

Table 2: SEA-RAFT achieves the best zero-shot performance on KITTI(train). By default, all methods are trained with "C+T". We list the extra data in the first column. † denotes the method uses tiling in inference. * denotes the GMFlow [56] ablation with 200k training steps. We use K and H to denote KITTI [36] and HD1K [23] respectively.

As shown in Tab. 1, SEA-RAFT achieves the best results among representative existing methods without using extra data, which demonstrates the superiority of our mixture loss and architecture design. When allowed to use extra data, SEA-RAFT falls slightly behind MS-RAFT+ [19] but is 24× faster and 11× smaller as mentioned in Fig. 1.

Fine-Tuning Test SEA-RAFT ranks 1st on the public test benchmark: SEA-RAFT(M) outperforms all other methods by at least 22.9% on average EPE(endpoint-error) and 17.8% on 1px (1-pixel outlier rate), and SEA-RAFT(S) outperforms other methods by at least 20.0% on EPE and 12.8% on 1px. Besides the strong performance, our method is notably fast. SEA-RAFT(S) is at least 2.3× faster than existing methods which can achieve similar performance. As we still follow the downsample-upsample protocol without using any test-time optimizations in submissions, the inference latency directly reflects our speed in handling 1080p images, which means over 20fps on a single RTX3090.

Extra Data	Method	Sintel		KITTI			Inference Cost	
		Clean↓	Final↓	Fl-all↓	Fl-bg↓	Fl-fg↓	#MACs	Latency
	PWC-Net+ [46]	3.45	4.60	7.72	7.69	7.88	**101.3G**	**23.82ms**
	RAFT [50]	1.61⋆	2.86⋆	5.10	4.74	6.87	938.2G	140.7ms
	GMA [20]	1.39⋆	2.47⋆	5.15	-	-	1352G	183.3ms
	DIP [68]	1.44⋆	2.83⋆	4.21	3.86	5.96	3068G	498.9ms
	GMFlowNet [66]	1.39	2.65	4.79	4.39	6.84	1094G	244.3ms
	GMFlow [56]	1.74	2.90	9.32	9.67	7.57	602.6G	138.5ms
	CRAFT [44]	1.45⋆	2.42⋆	4.79	4.58	5.85	2274G	483.4ms
	FlowFormer [14]	1.20	2.12	4.68†	4.37†	6.18†	1715G	335.6ms
	SKFlow [48]	1.28⋆	2.23⋆	4.85	4.55	6.39	1453G	331.9ms
	GMFlow+ [57]	**1.03**	2.37	4.49	4.27	5.60	1177G	249.6ms
	EMD-L [6]	1.32	2.51	4.49	4.16	6.15	1755G	OOM
	RPKNet [37]	1.31	2.65	4.64	4.63	**4.69**	137.0G	183.3ms
VIPER [41]	CCMR+ [18]	1.07	2.10	3.86	3.39	6.21	12653G	OOM
MegaDepth [26]	MatchFlow(G) [8]	1.16⋆	2.37⋆	4.63†	4.33†	6.11†	1669G	290.6ms
YouTube-VOS [59]	Flowformer++ [43]	1.07	**1.94**	4.52†	-	-	1713G	373.4ms
CroCo-Pretrain	CroCoFlow [55]	1.09†	2.44†	3.64†	3.18†	5.94†	57343G†	6422ms†
DDVM-Pretrain	DDVM [42]	1.75†	2.48†	**3.26**†	**2.90**†	5.05†	-	-
TartanAir [52]	**SEA-RAFT(M)**	1.44	2.86	4.64	4.47	5.49	486.9G	70.96ms
TartanAir [52]	**SEA-RAFT(L)**	1.31	2.60	4.30	4.08	5.37	655.1G	108.0ms

Table 3: Compared with other methods that achieve competitive performance, SEA-RAFT is at least 1.8× faster on Sintel(test) [3] and 4.6× faster on KITTI(test) [36]. All methods have undergone "C+T+S+K+H" training by default and we list the extra data each method uses in the first column. We measure latency on an RTX3090 with a batch size of 1 and input resolution 540 × 960. ⋆ denotes the method uses warm-start [50] strategy. † denotes that the corresponding methods use tiling-based test-time optimizations.

4.2 Results on Sintel and KITTI

Zero-Shot Evaluation Following previous works, we evaluate the zero-shot performance of SEA-RAFT given training schedule "C+T" on Sintel(train) [3] and KITTI(train) [36]. The results are provided in Tab. 2. On KITTI(train), SEA-RAFT outperforms all prior works by a large margin, improving Fl-epe from 4.09 to 3.62 and Fl-all from 13.7 to 12.9. On Sintel(train), SEA-RAFT achieves competitive results on the clean pass but, for reasons unclear to us, underperforms existing methods on the final pass. Note that although this "C+T" zero-shot setting is standard, it is of limited relevance to real-world applications, which do not need to restrict the training data to only C+T. Indeed, we show that by adding a small amount of high-quality real-world data (KITTI + HD1K, about 1.2k image pairs compared with 80k image pairs in FlyingThings3D [34]), the performance gap on the Sintel(train) final pass can be remarkably reduced.

Fine-Tuning Test Results are shown in Tab. 3. Compared with RAFT [50], SEA-RAFT achieves 19.9% improvements on the Sintel clean pass, 4.2% improvements on the Sintel final pass, and 15.7% improvements on KITTI Fl-all score. SEA-RAFT is also competitive among all existing methods in terms of performance-speed trade-off: It is the only method that can achieve results better than RAFT [50] with latency around 70ms. On Sintel(test), methods with

Experiment	Init.	Pre-Training		RNN		Loss Design		#MACs	EPE
		Img [7]	Tar [52]	GRU	#blocks	Type	Params		
SEA-RAFT (w/o Tar.)	Direct Reg.	✓	-	-	2	Mixture-of-Laplace	$\beta_1 = 0, \beta_2 \in [0, 10]$	284.7G	0.187
SEA-RAFT (w/ Tar.)	Direct Reg.	✓	✓	-	2	Mixture-of-Laplace	$\beta_1 = 0, \beta_2 \in [0, 10]$	284.7G	0.179
w/o Img.	Direct Reg.	-	-	-	2	Mixture-of-Laplace	$\beta_1 = 0, \beta_2 \in [0, 10]$	284.7G	0.194
w/o Direct Reg.	Zero Init. Warm Start	✓	-	-	2	Mixture-of-Laplace	$\beta_1 = 0, \beta_2 \in [0, 10]$	277.3G	0.201 0.202
RAFT GRU	Direct Reg.	✓	-	✓	-	Mixture-of-Laplace	$\beta_1 = 0, \beta_2 \in [0, 10]$	297.9G	0.189
More ConvNeXt Blocks	Direct Reg.	✓	-	-	4	Mixture-of-Laplace	$\beta_1 = 0, \beta_2 \in [0, 10]$	314.7G	0.189
Naive Laplace	Direct Reg.	✓	-	-	2	Naive Single Laplace Naive Mixture-of-Laplace	$\beta \in [-10, 10]$ $\beta_1, \beta_2 \in [-10, 10]$	284.7G	0.217 0.248
L1	Direct Reg.	✓	-	-	2	L_1	✗	284.7G	0.206
Gaussian	Direct Reg.	✓	-	-	2	Mixture-of-Gaussian	$\sigma_1 = 1, \sigma_2 = e^{\beta_2}, \beta_2 \in [0, 10]$	284.7G	0.210

Table 4: We ablate pretraining, direct regression, RNN design, and loss designs on Spring [35] subval. The effect of changes can be identified through comparisons with the first row. See Sec. 4.3 for details.

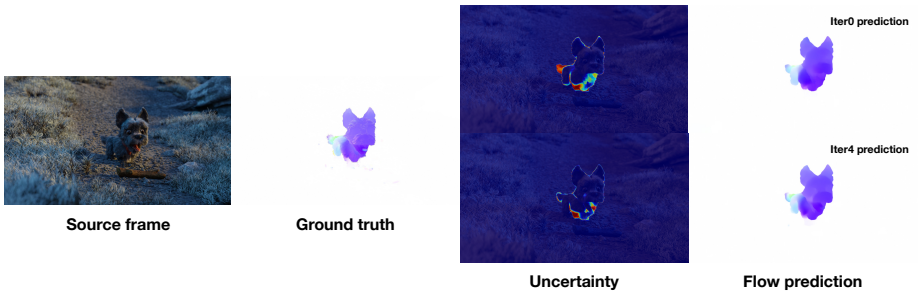

Fig. 6: More iterations produce lower variance in the Mixture of Laplace, indicating that the model becomes more confident after each iteration.

similar performance are at least 1.8× slower than us. On KITTI(test), methods with similar performance are at least 4.6× slower than us.

4.3 Ablations and Analysis

Ablation experiments are conducted on the Spring [35] dataset based on SEA-RAFT(S). We separate a subval set (sequence 0045 and 0047) from the original training set, train our model on the remaining training data and evaluate the performance on subval. The model is trained with a batch size of 32, input resolution 540×960, and tested following "downsample-upsample" protocol mentioned in Sec. 4.1. We describe the details of ablation studies in the following and show the results in Tab. 4:

Pretraining We test the performance of TartanAir [52] rigid-flow pre-training on different datasets(see Tabs. 1, 2 and 4 for details). Though TartanAir has been incorporated in several existing methods [42, 55], we clearly demonstrate its usefulness by ablations on SEA-RAFT and non-RAFT-style GMFlow [56]. Without TartanAir, SEA-RAFT already provides strong performance, and the

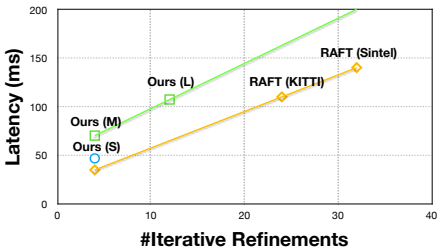

Fig. 7: Iterative refinements are not hardware-friendly: The latency almost linearly increases with the number of iterations.

Method	#Iters	Latency (ms)	
		Total	Iter.
RAFT [50]	24 (K)	111	90.3 (82%)
	32 (S)	141	120 (86%)
SEA-RAFT	4 (S)	47.5	18.5 (39%)
	4 (M)	70.9	18.5 (26%)
	12 (L)	108	55.5 (51%)

Table 5: Compared with RAFT, SEA-RAFT significantly reduces the cost of iterative refinements, which allows larger backbones while still being faster. We use K and S to denote RAFT submissions on KITTI and Sintel respectively.

rigid-flow pre-training makes it better. We also show that ImageNet pre-trained weights are effective.

RNN Design Our new RNN designs can reduce the computation without performance loss compared with the GRU used in RAFT [50]. We also show that on Spring subval, 4 ConvNeXt blocks do not work better than 2 ConvNeXt blocks.

Loss Design Naive Laplace regression does worse than the original L_1 loss. Also, it is important to set β_1 to 0 in the MoL loss, which aligns the MoL loss to L_1 for ordinary cases. Besides, we find that the mixture of Gaussian loss does not work well for optical flow, even though it has been found to be useful for image matching [4].

Direct Regression of Initial Flow The regressed flow initialization significantly improves accuracy without introducing much overhead. Also, we notice that the warm-start strategy, which initializes the flow with previous results, does not improve the performance in Spring ablations.

Inference Time Breakdown In Fig. 7, we show how the computational cost increases when we add more refinements. The cost bottleneck for SEA-RAFT is no longer iterative refinements (Tab. 5), which allows us to use larger backbones given the same computational cost constraint as RAFT [50].

5 Conclusion

We have introduced SEA-RAFT, a simpler, more efficient and accurate variant of RAFT. It achieves high accuracy across a diverse range of datasets, strong cross-dataset generalization, and state-of-the-art accuracy-speed trade-offs, making it useful for real-world high-resolution optical flow.

Acknowledgements

This work was partially supported by the National Science Foundation.

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. International journal of computer vision **92**, 1–31 (2011)
2. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: International conference on machine learning. pp. 1613–1622. PMLR (2015)
3. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: European conference on computer vision. pp. 611–625. Springer (2012)
4. Chen, H., Luo, Z., Zhou, L., Tian, Y., Zhen, M., Fang, T., Mckinnon, D., Tsin, Y., Quan, L.: Aspanformer: Detector-free image matching with adaptive span transformer. In: European Conference on Computer Vision. pp. 20–36. Springer (2022)
5. Chen, Q., Koltun, V.: Full flow: Optical flow estimation by global optimization over regular grids. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4706–4714 (2016)
6. Deng, C., Luo, A., Huang, H., Ma, S., Liu, J., Liu, S.: Explicit motion disentangling for efficient optical flow estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9521–9530 (2023)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
8. Dong, Q., Cao, C., Fu, Y.: Rethinking optical flow from geometric matching consistent perspective. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1337–1347 (2023)
9. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2758–2766 (2015)
10. Gao, C., Saraf, A., Huang, J.B., Kopf, J.: Flow-edge guided video completion. In: European Conference on Computer Vision. pp. 713–729. Springer (2020)
11. Garrepalli, R., Jeong, J., Ravindran, R.C., Lin, J.M., Porikli, F.: Dift: Dynamic iterative field transforms for memory efficient optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2219–2228 (2023)
12. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
13. Horn, B.K., Schunck, B.G.: Determining optical flow. Artificial intelligence **17**(1-3), 185–203 (1981)
14. Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K.C., Qin, H., Dai, J., Li, H.: Flowformer: A transformer architecture for optical flow. In: European Conference on Computer Vision. pp. 668–685. Springer (2022)
15. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Rife: Real-time intermediate flow estimation for video frame interpolation. arXiv preprint arXiv:2011.06294 (2020)

16. Hui, T.W., Tang, X., Loy, C.C.: Liteflownet: A lightweight convolutional neural network for optical flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8981–8989 (2018)
17. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2462–2470 (2017)
18. Jahedi, A., Luz, M., Rivinius, M., Bruhn, A.: Ccmr: High resolution optical flow estimation via coarse-to-fine context-guided motion reasoning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 6899–6908 (2024)
19. Jahedi, A., Luz, M., Rivinius, M., Mehl, L., Bruhn, A.: Ms-raft+: High resolution multi-scale raft. International Journal of Computer Vision pp. 1–22 (2023)
20. Jiang, S., Campbell, D., Lu, Y., Li, H., Hartley, R.: Learning to estimate hidden motions with global motion aggregation. arXiv preprint arXiv:2104.02409 (2021)
21. Jung, H., Hui, Z., Luo, L., Yang, H., Liu, F., Yoo, S., Ranjan, R., Demandolx, D.: Anyflow: Arbitrary scale optical flow with implicit neural representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5455–5465 (2023)
22. Kim, D., Woo, S., Lee, J.Y., Kweon, I.S.: Deep video inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5792–5801 (2019)
23. Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., et al.: The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 19–28 (2016)
24. Leroy, V., Revaud, J., Lucas, T., Weinzaepfel, P.: Win-win: Training high-resolution vision transformers from two windows. arXiv preprint arXiv:2310.00632 (2023)
25. Li, J., Bian, S., Zeng, A., Wang, C., Pang, B., Liu, W., Lu, C.: Human pose regression with residual log-likelihood estimation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 11025–11034 (2021)
26. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2041–2050 (2018)
27. Liu, X., Liu, H., Lin, Y.: Video frame interpolation via optical flow estimation with image inpainting. International Journal of Intelligent Systems **35**(12), 2087–2102 (2020)
28. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11976–11986 (2022)
29. Lu, Y., Wang, Q., Ma, S., Geng, T., Chen, Y.V., Chen, H., Liu, D.: Transflow: Transformer as flow learner. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18063–18073 (2023)
30. Luo, A., Yang, F., Li, X., Liu, S.: Learning optical flow with kernel patch attention. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8906–8915 (2022)
31. Luo, A., Yang, F., Li, X., Nie, L., Lin, C., Fan, H., Liu, S.: Gaflow: Incorporating gaussian attention into optical flow. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9642–9651 (2023)

32. Luo, A., Yang, F., Luo, K., Li, X., Fan, H., Liu, S.: Learning optical flow with adaptive graph reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2022)
33. Ma, Z., Teed, Z., Deng, J.: Multiview stereo with cascaded epipolar raft. In: European Conference on Computer Vision. pp. 734–750. Springer (2022)
34. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4040–4048 (2016)
35. Mehl, L., Schmalfuss, J., Jahedi, A., Nalivayko, Y., Bruhn, A.: Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4981–4991 (2023)
36. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3061–3070 (2015)
37. Morimitsu, H., Zhu, X., Ji, X., Yin, X.C.: Recurrent partial kernel network for efficient optical flow estimation (2024)
38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
39. Piergiovanni, A., Ryoo, M.S.: Representation flow for action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9945–9953 (2019)
40. Raistrick, A., Lipson, L., Ma, Z., Mei, L., Wang, M., Zuo, Y., Kayan, K., Wen, H., Han, B., Wang, Y., et al.: Infinite photorealistic worlds using procedural generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12630–12641 (2023)
41. Richter, S.R., Hayder, Z., Koltun, V.: Playing for benchmarks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2213–2222 (2017)
42. Saxena, S., Herrmann, C., Hur, J., Kar, A., Norouzi, M., Sun, D., Fleet, D.J.: The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. Advances in Neural Information Processing Systems **36** (2024)
43. Shi, X., Huang, Z., Li, D., Zhang, M., Cheung, K.C., See, S., Qin, H., Dai, J., Li, H.: Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1599–1610 (2023)
44. Sui, X., Li, S., Geng, X., Wu, Y., Xu, X., Liu, Y., Goh, R., Zhu, H.: Craft: Cross-attentional flow transformer for robust optical flow. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 17602–17611 (2022)
45. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8934–8943 (2018)
46. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Models matter, so does training: An empirical study of cnns for optical flow estimation. IEEE transactions on pattern analysis and machine intelligence **42**(6), 1408–1423 (2019)
47. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: Detector-free local feature matching with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8922–8931 (2021)

48. Sun, S., Chen, Y., Zhu, Y., Guo, G., Li, G.: Skflow: Learning optical flow with super kernels. Advances in Neural Information Processing Systems **35**, 11313–11326 (2022)
49. Sun, S., Kuang, Z., Sheng, L., Ouyang, W., Zhang, W.: Optical flow guided feature: A fast and robust motion representation for video action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1390–1399 (2018)
50. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020)
51. Truong, P., Danelljan, M., Timofte, R., Van Gool, L.: Pdc-net+: Enhanced probabilistic dense correspondence network. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
52. Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., Scherer, S.: Tartanair: A dataset to push the limits of visual slam. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4909–4916. IEEE (2020)
53. Wannenwetsch, A.S., Keuper, M., Roth, S.: Probflow: Joint optical flow and uncertainty estimation. In: Proceedings of the IEEE international conference on computer vision. pp. 1173–1182 (2017)
54. Weinzaepfel, P., Leroy, V., Lucas, T., Brégier, R., Cabon, Y., Arora, V., Antsfeld, L., Chidlovskii, B., Csurka, G., Revaud, J.: Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion. Advances in Neural Information Processing Systems **35**, 3502–3516 (2022)
55. Weinzaepfel, P., Lucas, T., Leroy, V., Cabon, Y., Arora, V., Brégier, R., Csurka, G., Antsfeld, L., Chidlovskii, B., Revaud, J.: Croco v2: Improved cross-view completion pre-training for stereo matching and optical flow. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17969–17980 (2023)
56. Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Tao, D.: Gmflow: Learning optical flow via global matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8121–8130 (2022)
57. Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Yu, F., Tao, D., Geiger, A.: Unifying flow, stereo and depth estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
58. Xu, J., Ranftl, R., Koltun, V.: Accurate optical flow via direct cost volume processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1289–1297 (2017)
59. Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., Huang, T.: Youtube-vos: A large-scale video object segmentation benchmark. arXiv preprint arXiv:1809.03327 (2018)
60. Xu, R., Li, X., Zhou, B., Loy, C.C.: Deep flow-guided video inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3723–3732 (2019)
61. Xu, X., Siyao, L., Sun, W., Yin, Q., Yang, M.H.: Quadratic video interpolation. Advances in Neural Information Processing Systems **32** (2019)
62. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime tv-l 1 optical flow. In: Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29. pp. 214–223. Springer (2007)
63. Zhai, M., Xiang, X., Lv, N., Ali, S.M., El Saddik, A.: Skflow: Optical flow estimation using selective kernel networks. Ieee Access **7**, 98854–98865 (2019)
64. Zhang, S., Sun, X., Chen, H., Li, B., Shen, C.: Rgm: A robust generalist matching model. arXiv preprint arXiv:2310.11755 (2023)

65. Zhao, S., Sheng, Y., Dong, Y., Chang, E.I., Xu, Y., et al.: Maskflownet: Asymmetric feature matching with learnable occlusion mask. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6278–6287 (2020)
66. Zhao, S., Zhao, L., Zhang, Z., Zhou, E., Metaxas, D.: Global matching with overlapping attention for optical flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17592–17601 (2022)
67. Zhao, Y., Man, K.L., Smith, J., Siddique, K., Guan, S.U.: Improved two-stream model for human action recognition. EURASIP Journal on Image and Video Processing **2020**(1), 1–9 (2020)
68. Zheng, Z., Nie, N., Ling, Z., Xiong, P., Liu, J., Wang, H., Li, J.: Dip: Deep inverse patchmatch for high-resolution optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8925–8934 (2022)
69. Zuo, Y., Deng, J.: View synthesis with sculpted neural points. arXiv preprint arXiv:2205.05869 (2022)