International Journal of Computer Vision (2024) 132:1835–1856
https://doi.org/10.1007/s11263-023-01930-7

MS-RAFT+: High Resolution Multi-Scale RAFT

Azin Jahedi[1] · Maximilian Luz[1,2] · Marc Rivinius[3] · Lukas Mehl[1] · Andrés Bruhn[1]

Received: 16 December 2022 / Accepted: 9 October 2023 / Published online: 18 December 2023

Abstract
Hierarchical concepts have proven useful in many classical and learning-based optical flow methods regarding both accuracy and robustness. In this paper we show that such concepts are still useful in the context of recent neural networks that follow RAFT's paradigm refraining from hierarchical strategies by relying on recurrent updates based on a single-scale all-pairs transform. To this end, we introduce MS-RAFT+: a novel recurrent multi-scale architecture based on RAFT that unifies several successful hierarchical concepts. It employs a coarse-to-fine estimation to enable the use of finer resolutions by useful initializations from coarser scales. Moreover, it relies on RAFT's correlation pyramid that allows to consider non-local cost information during the matching process. Furthermore, it makes use of advanced multi-scale features that incorporate high-level information from coarser scales. And finally, our method is trained subject to a sample-wise robust multi-scale multi-iteration loss that closely supervises each iteration on each scale, while allowing to discard particularly difficult samples. In combination with an appropriate mixed-dataset training strategy, our method performs favorably. It not only yields highly accurate results on the four major benchmarks (KITTI 2015, MPI Sintel, Middlebury and VIPER), it also allows to achieve these results with a single model and a single parameter setting. Our trained model and code are available at https://github.com/cv-stuttgart/MS_RAFT_plus.

Keywords Optical flow · Recurrent neural networks · Multi-Scale concepts · Coarse-to-fine schemes · Feature pyramid networks · Robust training loss · Generalization across datasets · Robust Vision Challenge

1 Introduction

Accurate motion estimation is essential for reliable visual perception tasks allowing, among other things, the avoidance of collisions and the inference of 3D geometry from monocular vision. Hence, the investigation and estimation of motion cues is a central and long-studied problem in computer vision. A specific instance of this problem is the estimation of optical flow, which denotes the motion between two consecutive frames of an image sequence encoded as a 2D vector field in the image plane. Optical flow in particular has found widespread use for providing essential motion cues to a diverse set of applications, including video manipulation and enhancement (Lai et al. 2018; Tu et al. 2022), biometrics (Singh et al. 2021; Mahfouf et al. 2018) and biomedical imaging (Philipp et al. 2022), action recognition (Sevilla-Lara et al. 2019; Yao et al. 2019), visual surveillance (Kajo et al. 2015; Sreenu and Saleem Durai 2019), human and object tracking (Ranjan et al. 2020; Luo et al. 2021; Ali et al. 2016), as well as autonomous driving and robot navigation (Janai et al. 2020; Giachetti et al. 1998; Chao et al. 2013; Yousif et al. 2015).

This diversity naturally triggers the demand for generally applicable methods, i.e., approaches that generalize well across datasets (Zendel et al. 2022). That this poses a problem to recent neural networks can be seen from the fact that they typically require individual fine-tuning on each dataset to achieve state-of-the-art results (Teed and Deng 2020; Xu et al. 2021; Zhang et al. 2021; Jiang et al. 2021; Huang et al. 2022). In this context, they also often apply a different num-

Communicated by Oliver Zendel.

✉ Azin Jahedi
azin.jahedi@vis.uni-stuttgart.de

✉ Maximilian Luz
luz@cs.uni-freiburg.de

[1] Institute for Visualization and Interactive Systems, University of Stuttgart, Stuttgart, Germany

[2] Robot Learning Lab, University of Freiburg, Freiburg, Germany

[3] Institute of Information Security, University of Stuttgart, Stuttgart, Germany

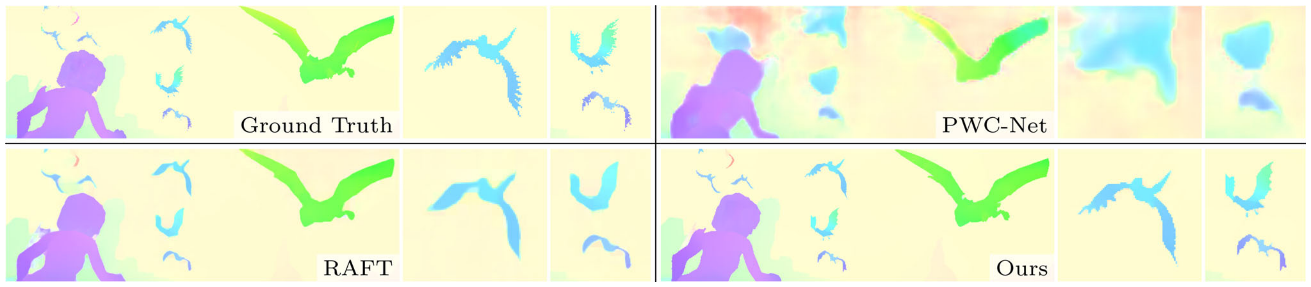

Fig. 1 Comparison of the estimation quality of our MS-RAFT+ method to PWC-Net (Sun et al. 2018) and RAFT (Teed and Deng 2020) by the example of an image sequence from the MPI Sintel benchmark (clean render pass). Note the small details captured by our method visible in the zoom-ins on the right

ber of recurrent iterations for each dataset (Teed and Deng 2020; Long and Lang 2022; Xu et al. 2021; Zhang et al. 2021). Besides, some methods rely on resolution-dependent tiling (Jaegle et al. 2022; Huang et al. 2022) or perform dataset-specific augmentations (Sun et al. 2018; Teed and Deng 2020). This clearly shows the need for optical flow methods that are robust in the sense that they work well on a variety of datasets with the same model and the same settings.

Besides the lack of robustness, recent neural networks also exhibit a shortcoming that concerns the quality of their output. To reduce both memory requirements and computational effort such networks typically rely on downsampled representations. In fact, in most cases they operate at only $1/8$ of the input size (Teed and Deng 2020; Long and Lang 2022; Xu et al. 2021; Jiang et al. 2021; Zhang et al. 2021; Huang et al. 2022), finally restoring the original resolution by a subsequent learned upsampling (Teed and Deng 2020). Not surprisingly, this leads to a loss of fine details which are lost during downsampling and cannot be recovered. This in turn shows the need for methods that operate on finer resolutions and hence are able to provide more detailed flow fields.

Finally, there is also an intrinsic motivation to our approach from the methodological side. Hierarchical concepts, and in particular coarse-to-fine schemes, have a long-standing history in optical flow computation. Introduced in classical methods (Lucas and Kanade 1981; Anandan 1989; Black and Anandan 1991; Brox et al. 2004; Xu et al. 2011; Hu et al. 2016; Enkelmann 1988), they have not only been essential to capture large motion by allowing a wider field of perception at coarser resolutions, but also allow to keep complexity and memory requirements small at the same time. While such approaches have also been used in earlier neural networks (Ranjan and Black 2017; Sun et al. 2018; Yang and Ramanan 2019), contemporary methods largely focus on single-scale flow estimation (Teed and Deng 2020; Jiang et al. 2021; Jaegle et al. 2022).

The idea behind these recent single-scale methods is to enable a wider perceptive range by using techniques such as dilated convolutions (Lu et al. 2020), iterative refinement (Teed and Deng 2020; Jiang et al. 2021), or full all-pairs matching on improved features (Xu et al. 2022). In this way, they aim at avoiding the main problem of coarse-to-fine schemes: the handling of small but fast-moving objects. Such objects do not appear at those coarser scales that would allow to estimate their motion reliably (Brox et al. 2009).

The use of single-scale methods, however, comes at the expense of robustness: While coarse-to-fine schemes increase the perceptive range, they also render the estimation more robust w.r.t. local minima. For instance, in case of repetitive structures or textureless areas, ambiguities can typically be resolved when including coarser scales. Single-scale methods do not have this advantage. Hence, RAFT (Teed and Deng 2020) and its many variants (e.g., Jiang et al. 2021; Zhang et al. 2021; Xu et al. 2021) try to tackle this shortcoming by considering a hierarchical cost volume. Our results, however, indicate that this strategy alone may not be sufficient. In fact, considering additional multi-scale concepts in both architecture and training loss turns out to be strongly beneficial.

1.1 Contributions

In our paper, we address the aforementioned shortcomings of recent methods regarding details and robustness by proposing MS-RAFT+: a novel multi-scale recurrent optical flow method that combines the benefits of RAFT-based recurrent neural networks with the advantages of established coarse-to-fine estimation schemes. To this end, we cast four hierarchical concepts into a single estimation framework: a coarse-to-fine scheme, a hierarchical cost volume, a U-Net-style feature extractor and a multi-scale multi-iteration loss. In this context, our contributions are fourfold:

(i) We develop a 4-level coarse-to-fine scheme that finally operates at $1/2$ of the original resolution. By sharing not only the recurrent update block but also the $\times 2$ learned convex upsampler among all scales it avoids both the use of larger upsampling factors and the use of nonlearned usampling strategies, hence allowing to recover fine details and crisp motion boundaries (see Fig. 1). Please note that we still employ RAFT's hierarchical cost volume. It complements our coarse-to-fine scheme by introducing non-local information in the recurrent update process.

(ii) We propose a U-Net-style multi-scale feature extractor that considers additional high-level information from coarser scales. In this way, we incorporate multi-scale information at feature level to improve the matching itself.

(iii) We suggest a sample-wise robust multi-scale multi-iteration loss. While the multi-scale multi-iteration loss guarantees a close supervision of intermediate results, the sample-wise robust penalization allows to discard particularly difficult frames during training.

(iv) By combining our sample-wise robust multi-scale multi-iteration loss with a suitable training strategy we are able to train a single model performing well across all commonly used optical flow benchmarks: KITTI 2015, MPI Sintel, Middlebury and VIPER. Most importantly, our model does so without requiring any benchmark-specific modifications and fine-tuning, highlighting its generalization capabilities.

Due to these properties, MS-RAFT+ won the ECCV 2022 Robust Vision Challenge (Zendel et al. 2022) in the category optical flow. This clearly confirms our findings that hierarchical concepts are still useful in motion estimation.

The present article extends our preliminary conference paper (Jahedi et al. 2022) in a number of ways. Apart from a more extensive discussion of related work and a significantly more detailed description of our method, we (i) extend our multi-scale architecture by an additional finer scale which requires the cost of matching-candidates to be computed on-demand, (ii) further improve training and results by a specifically tailored mixed-dataset fine-tuning, and (iii) present a more detailed experimental evaluation that includes additional datasets and ablations.

1.2 Organization

The paper is organized as follows. After discussing related work in Sect. 2, we start by reviewing the original RAFT method in Sect. 3. Based on its single-scale architecture we then propose our novel multi-scale approach in Sect. 4. Thereby we elaborate on all details of our approach with a particular focus on the different multi-scale concepts that are combined within our method. In Sect. 5 we then thoroughly evaluate our novel multi-scale approach using four major benchmarks. In this context, we also perform ablations to justify our architectural choices. Finally, we discuss limitations of our approach in Sect. 6 and conclude the paper with a summary in Sect. 7.

2 Related Work

In the following, we comment on related work on coarse-to-fine estimation, iterative refinement, and the combination thereof. Afterwards, we discuss related work on multi-scale concepts for feature extraction and training supervision.

2.1 Coarse-to-Fine Estimation

In classical optical flow approaches, hierarchical concepts were essential to enable both the handling of large motions and the robustness w.r.t. local minima (Anandan 1989; Black and Anandan 1991; Brox et al. 2004). One of the most prevalent concepts in this context are coarse-to-fine warping schemes. Such schemes were first introduced by (Lucas and Kanade, 1981) and later on established by (Black and Anandan, 1991) and (Brox et al., 2004). The main idea of coarse-to-fine warping is the estimation of the residual motion at each scale. Thus the flow is iteratively refined from coarsest to finest scale, each time warping the second frame towards the first using the current estimate.

In the context of neural networks, SpyNet (Ranjan and Black 2017) introduced a straight-forward adaption of the coarse-to-fine warping strategy. More precisely, it combined a spatial image pyramid with image warping. PWC-Net (Sun et al. 2018) and LiteFlowNet (Hui et al. 2018) improved upon this by constructing feature pyramids and warping the features instead. Therewith, they laid the foundation for many subsequent (e.g.,Yang and Ramanan 2019; Yin et al. 2019) and contemporary approaches (e.g., Zheng et al. 2022), including our approach.

2.2 Iterative Refinement

While most coarse-to-fine warping approaches can be seen as inherently iterative due to the repetition of identical structures across all scales, there is a group of methods that fits this description more aptly. On the one hand, (Hur and Roth, 2019) suggests the use of weight-sharing between the coarse-to-fine levels of PWC-Net. On the other hand, Devon (Lu et al. 2020) proposes to estimate the flow iteratively on a single level while emulating the general coarse-to-fine scheme with increasingly more local dilated matching-cost computation. In addition, a sampling-based strategy is employed to facilitate residual flow estimation without the need for warping. Finally, RAFT (Teed and Deng 2020) considers a similar single-scale estimation strategy also based on sampling. However, it uses a pooled cost volume to create a hierarchical context and introduces a recurrent network for refinement, leading to an approach that still shapes contemporary methods (Jiang et al. 2021; Zhang et al. 2021).

This shift towards sampling-based methods avoids the main drawback of warping-based approaches. As noted by (Hofinger et al., 2020), warping can lead to ambiguities through duplication, so-called ghosting, when objects and background move at different speeds. Sampling-based approaches, on the other hand, overcome these problems by adjusting all operations to sample directly from the unmodified image, offsetting this process by the current flow estimate.

2.3 Combined Approaches

With MS-RAFT+ we combine the recurrent iterative updates of RAFT (Teed and Deng 2020) with a sampling-based coarse-to-fine estimation framework. In addition, we share the weights of the recurrent update network and the learned convex upsampler among scales. Our approach therefore integrates the findings and best practices of several of the works discussed above (in particular Hur and Roth 2019; Hofinger et al. 2020).

Closest in spirit to our multi-scale approach is the PRAFlow_RVC method by (Wan et al., 2020) that explores only two scales in a direct RAFT-based setting without further modifications. In contrast to this, our approach investigates up to four scales, embeds higher-level multi-scale features, relies on a sample-wise robust multi-scale multi-iteration loss and shares the learned interscale upsampling among all scales.

Two-scale refinement schemes are also employed by GMFlow (Xu et al. 2022) and DIP (Zheng et al. 2022). The former forgoes the use of any recurrent strategies and instead focuses on feature enhancement via a transformer architecture in combination with a fixed softmax regression. The latter, in contrast, proposes a recurrent estimation approach based on the classical PatchMatch method by (Barnes et al., 2009). Neither of them, however, explores hierarchical concepts beyond their use as an additional refinement stage.

Finally, in the field of disparity estimation, there is the CREStereo method (Li et al. 2022). Developed in parallel to our approach, it employs a similar coarse-to-fine estimation scheme, using independent recurrent refinement on each scale with shared weights across scales. In contrast to our approach, however, the correlation volume computed on each coarse-to-fine scale is not hierarchical. Moreover, learned convex upsampling is only applied at the finest scale to interpolate the results to the original resolution instead of sharing such an upsampling module across all scales.

In contrast to all four of the aforementioned methods that use $1/4$ of the original resolution as final scale, our approach operates at $1/2$ of the input size, which is made possible by on-demand cost sampling. In conjunction with the sharing of the learned convex upsampler across all scales, our method therefore allows sharper motion boundaries and more structural details, which is also confirmed by our experimental evaluation.

2.4 Multi-Scale Feature Extraction

Now we discuss related work on the extraction of advanced hierarchical features. On the one hand, in the context of scene flow (Saxena et al. 2019; Rishav et al. 2021) and optical flow (Rishav et al. 2021) there are advanced encoders based on the concepts of feature pyramid networks (FPNs) (Lin et al. 2017) or feature pyramid networks with residual connections (ResFPNs) (Rishav et al. 2021). The idea of such encoders is to combine higher-level feature representations from coarser resolutions with better localized information from finer scales. On the other hand, there are approaches based on residual feature downsampling (RFD) explicitly designed for optical flow (Long and Lang 2022). Those approaches, in turn, allow a better preservation of details in the downsampling process by storing additional repair masks. Our feature encoder follows a U-Net style structure (Ronneberger et al. 2015) and is based on the concepts of FPNs and top-down modulation (TDM) (Shrivastava et al. 2017). However, it is equipped with additional residual blocks to better aggregate multi-scale features.

2.5 Multi-Scale Supervision

Finally, let us comment on related work in the field of multi-scale supervision. While SPyNet (Ranjan and Black 2017) and the more recent LiteFlowNet-based approaches (Hui et al. 2018, 2021; Hui and Loy 2020) employ a gradual strategy, pre-training on coarsest scales first before adding finer scales, newer networks tend to follow the strategy introduced by PWC-Net (Sun et al. 2018) and are trained as a single unit, using a multi-scale loss term (see e.g., Yang and Ramanan 2019; Yin et al. 2019). Generally, weights are set manually for each scale. Somewhat similarly, RAFT (Teed and Deng 2020) directly supervises each intermediate recurrent output at its single scale, however, it employs an exponential weighting scheme to reduce the impact of earlier predictions.

CREStereo (Li et al. 2022) combines both concepts into a multi-scale multi-iteration loss by summing together independent RAFT loss terms, one per scale, using the same weights for all scales. Our multi-scale multi-iteration loss, in contrast, unrolls the exponential weighting scheme of RAFT across all scales and hence leads to a lesser influence of coarser scales in the final loss. Apart from that, our loss is not only pixel-wise robust as most of the aforementioned methods, but also sample-wise robust, i.e., not only single pixels but entire samples are downweighted during training.

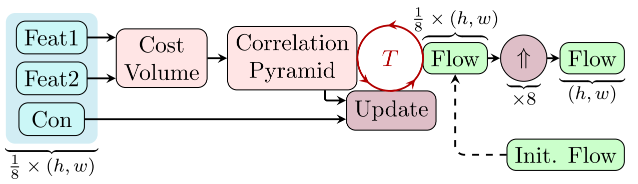

Fig. 2 RAFT's architecture. The flow-increment is estimated T times via the update block using the current flow estimate, looked-up costs which are outputs of the correlation pyramid, and context features denoted by *Con*. Note that the flow estimate at each iteration is used for looking up the costs in the correlation pyramid. The resulting flow is finally interpolated to the original resolution using a $\times 8$ convex upsampler

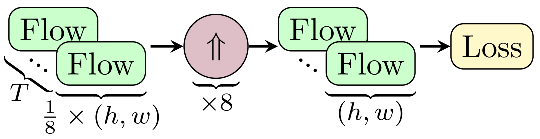

Fig. 3 RAFT's multi-iteration loss. Flow estimates of all iterations are upsampled to the original resolution using the convex upsampler and contribute to the overall loss

3 RAFT

Before explaining our RAFT-based multi-scale approach, we first review the most important concepts of single-scale RAFT (Teed and Deng 2020). We start by reviewing RAFT's architecture, followed by a discussion of its training loss.

3.1 Architecture

RAFT estimates the optical flow by processing feature representations of the input images at $1/8$ of their resolution. It computes all-pairs matching costs and, based on the costs and features, estimates the flow iteratively using convolutional and recurrent units. In the following, we elaborate on the estimation strategy of RAFT in more detail. The underlying architecture is illustrated in Fig. 2.

3.1.1 Feature Extraction

At the beginning, image features *Feat1*, *Feat2* of both input frames and context features *Con* of the first frame are computed. While a shared feature encoder is employed to extract the image features, a separate encoder with the same architecture is used for computing the context features. Each of the three resulting feature volumes has the spatial size $(h/8, w/8)$.

3.1.2 Correlation Pyramid

Based on the aforementioned image features *Feat1*, *Feat2*, an all-pairs cost volume is constructed, where the normalized dot product of the features is computed between all pairs of pixels, i.e., between each feature in the first frame and each feature in the second frame. This 4D cost volume—with size $(h/8, w/8, h/8, w/8)$—is then average-pooled three times along its last two dimensions. The pooled volumes together with the original cost volume yield a four-level correlation pyramid, where the coarsest level is of size $(h/8, w/8, h/64, w/64)$. This pyramid is later used for sampling the costs of match-candidates. Using all levels of the pyramid for such a cost sampling allows to capture non-local information in the matching process.

3.1.3 Recurrent Updates

At each recurrent iteration, matching costs are collected from the correlation pyramid. They are sampled from a neighborhood around the estimated correspondences at each pyramid level, given by the current flow estimate—which is typically initialized to zero or derived from a prior estimate. These costs, along with the current flow estimate, are passed through a motion encoder, where both are processed separately via convolutional layers. The output of the motion encoder together with a part of the context features is then fed into a Gated Recurrent Unit (GRU). This unit updates the hidden state over the recurrent iterations, which was initialized by a separate part of the context features.

The output of the GRU is then processed to compute a residual flow update at $1/8$ the resolution using another convolutional layer. These updates are summed up over several recurrent iterations yielding the final flow estimate. For simplicity, the update block in Fig. 2 contains the above-mentioned sub-networks to estimate the flow update from the current flow estimate, the context features, and the looked-up costs. Furthermore, a convex upsampler, learned from the hidden-state, is applied to upsample the flow estimated at $1/8$ the resolution to the original scale.

3.2 Supervision

In RAFT, all intermediate and final flow estimates at different iterations contribute in supervising the network. Unlike in PWC-Net (Sun et al. 2018), the difference between all flow estimates and the ground truth flow is computed at the original resolution. Therefore, in this case, downsampling the ground truth flow is not required. Figure 3 shows the inputs of RAFT's loss. Flow estimates from all iterations are upsampled to the original resolution using the learned convex upsampler and used for supervision. The deviations of the upsampled flow estimates from the ground truth flow are penalized at each iteration. This results in the following

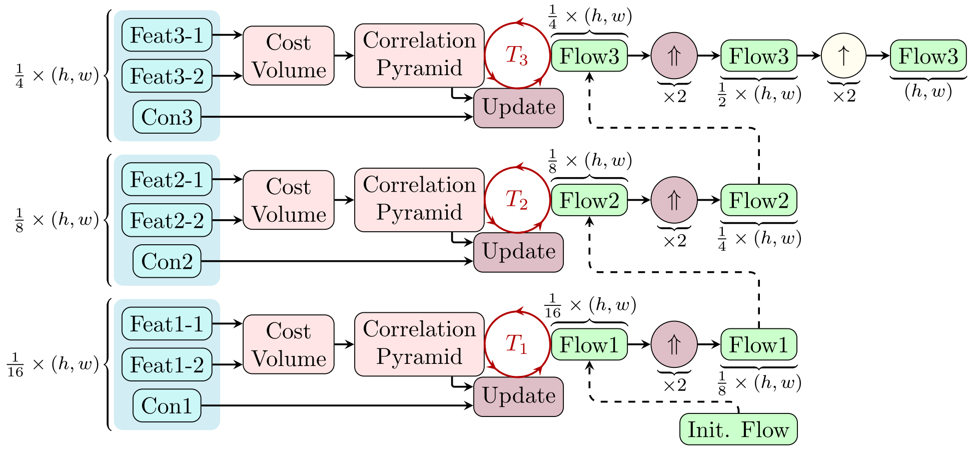

Fig. 4 Coarse-to-fine estimation scheme of our 3-scale MS-RAFT approach. The optical flow is estimated from the coarsest scale (bottom) to the finest scale (top). The flow computed on coarser scales is used as initialization for the finer scales after upsampling. The update module, which is responsible for estimating the flow-update and the learned convex upsampler, are shared among scales

multi-iteration loss

$$\mathcal{L} = \sum_{t=1}^{T} \gamma^{T-t} \|\mathbf{f}^t - \mathbf{f}^{gt}\|_1, \tag{1}$$

where $\mathbf{f}^t$ represents the estimated flow at iteration t, $\mathbf{f}^{gt}$ denotes the ground truth flow, and T is the number of recurrent iterations during training. Further, $\|\cdot\|_1$ indicates the L1 norm and γ^{T-t} is the weight for the penalized deviation at iteration t. Setting $\gamma = 0.8$ yields an exponential increase in the impact of later iterations on the loss.

4 Multi-Scale RAFT

Having reviewed RAFT, we can now discuss our novel multi-scale approach. In this approach, we take advantage of finer representations of the input images than 1/8 the original resolution, while benefiting from improved initializations from coarser scales. Ideally, one could compute the flow from feature representations as fine as the original images, provided that such features are high-level enough. But, the search range in that case would need to be very large for a single-scale approach in order to find the correspondences for objects with large motion. However, estimating the optical flow from such high-resolution features is memory-wise prohibitive for RAFT's architecture due to the computation of the all-pairs cost volume and the subsequent construction of the correlation pyramid.

Therefore, in this section, we discuss two different versions of our multi-scale approach. Firstly, we present a 3-scale version, where the finest scale is at 1/4 the original resolution, for which the computation of the all-pairs cost volume and the correlation pyramid is typically still feasible[1], both time- and memory-wise. We call this model *MS-RAFT* in this work. Secondly, we present an extension, exploiting an additional finer scale. Since including the additional scale and computing the all-pairs cost volume is memory-wise infeasible, we rely on an on-demand cost computation and sampling. We refer to this extended model as *MS-RAFT+*. In our evaluation in Sect. 5 we show that our 4-scale MS-RAFT+ approach yields substantial improvements over both RAFT and MS-RAFT.

In the following, we elaborate on each of the multi-scale concepts employed in our models, namely: a coarse-to-fine estimation scheme, a multi-scale U-Net-style feature encoder and a multi-scale multi-iteration loss. We first detail on the coarse-to-fine estimation strategy for both models and then discuss the other concepts, with an emphasis on the extended MS-RAFT+ model.

4.1 Multi-Scale Estimation

Let us start by discussing our multi-scale coarse-to-fine estimation scheme. For our 3-scale MS-RAFT model this scheme is shown in Fig. 4. We will talk about different aspects step-by-step, beginning with the introduction of the inputs of this module, followed by a discussion of the general estimation procedure. Lastly, we explain how we make use of

[1] By feasible in this case, we mean that one could train a method using 2 A100 GPUs, with 40 GB VRAM each, using the same batch-size as in RAFT without requiring gradient accumulation. This amount of memory is much larger than RAFT requires for training, however compared to other transformer-based methods such as GMFlow (Xu et al. 2022) and Flowformer (Huang et al. 2022), where four A100 and four V100 GPUs are used, respectively, our method still requires less memory.

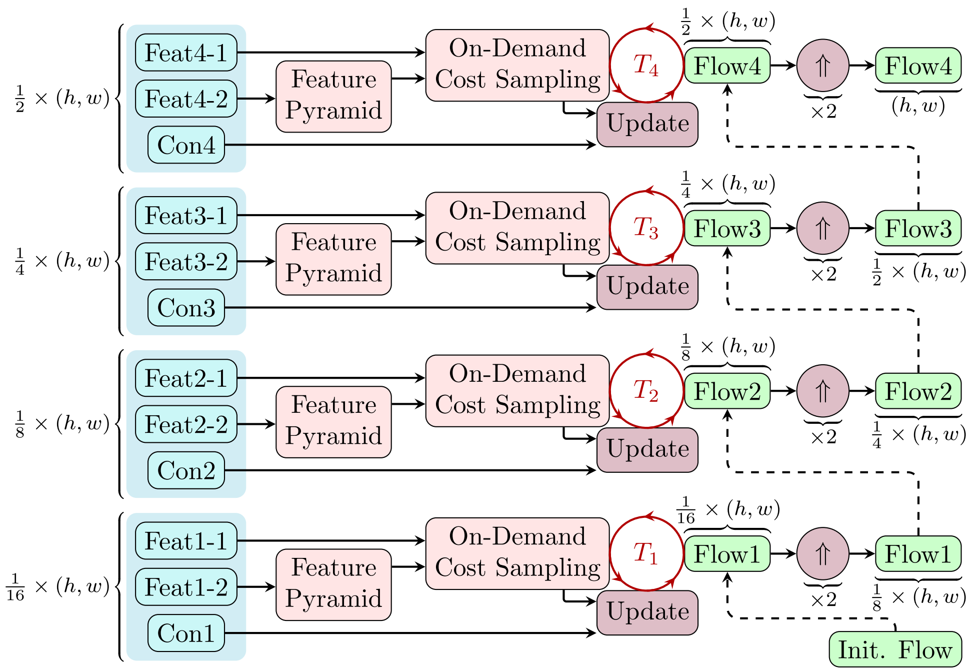

Fig. 5 Coarse-to-fine estimation scheme of our 4-scale MS-RAFT+ approach. Note the different estimation process between this figure and Fig. 4, i.e., no all-pairs cost volume is computed. The cost is computed and sampled on demand. Furthermore, using the additional scale allows to compute the flow at full resolution using the shared convex upsampler without any additional bilinear upsampling step

an additional finer scale in our 4-scale MS-RAFT+ model, shown in Fig. 5.

4.1.1 Inputs

As shown in Fig. 4, our matching process depends on first- and second-frame image features (*FeatX-1* and *FeatX-2*) as well as context features (*ConX*) at different scales. The spatial size of these features increases from bottom to top. They are computed using a U-Net-style feature extractor, which we will discuss in Sect. 4.2.

4.1.2 General Procedure

Given the image and context features mentioned previously at different resolution scales, the flow is first estimated at the coarsest scale S_1. Like in RAFT, the all-pairs cost volume and subsequently the correlation pyramid are computed, and the current flow estimate, the context features and the sampled matching costs from the correlation pyramid are employed to compute the flow update iteratively (cf. Sect. 3.1.1).

Like in RAFT, a part of the context features in the coarsest scale is passed to the recurrent unit as the overall initial hidden state. This hidden state gets updated at each iteration. After T_1 recurrent flow-updates at the coarsest scale (1/16 resolution), we perform a $\times 2$ convex upsampling ($\Uparrow$ in Fig. 4). The resulting flow is then used as initialization at the next finer scale S_2. At this scale, a part of the finer context features (*Con2*) of the current scale is used to re-initialize the hidden state of the recurrent unit. The flow is then updated for T_2 iterations, upsampled, and used as initialization at the next scale. This procedure is repeated until the flow at 1/4 the resolution has been computed at the 3rd scale, S_3. This flow is finally upsampled to the full resolution using the mentioned $\times 2$ convex upsampler followed by a $\times 2$ bilinear upsampler ($\uparrow$ in the figure). Importantly, the flow estimation module—update block in Fig. 4—and the $\times 2$ learned convex upsampler are shared across scales.

Note that after the flow is upsampled in the finest scale to 1/2 the resolution by the convex upsampler, one cannot simply apply the same convex upsampler again to compute the flow at full resolution. The reason is that the convex upsampler relies on the hidden state of the recurrent unit at the current scale. Therefore, to upsample the flow from 1/2 the resolution to full resolution, one would need the context features, the matching costs, and the estimated flow at 1/2 resolution.

4.1.3 On-Demand Cost Sampling

In the last section, we explained how the flow is computed in the coarse-to-finer manner in our 3-scale MS-RAFT model. In the following, we discuss how to utilize an additional finer scale, so that we improve the estimation by taking advantage of even finer features. Figure 5 demonstrates our extended MS-RAFT+ estimation scheme.

As elaborated previously, estimating the flow on a finer scale via the all-pairs cost volume is very memory-intensive. (Teed and Deng, 2020) have shown that the correlation costs of match candidates can instead be computed on-demand during training and inference[2]. Without the pre-computed all-pairs cost volume, however, the correlation pyramid can also not be pre-computed. Instead, a pyramid of the second frame's features (*FeatX-2* in Fig. 5) is constructed via successive pooling operations. A specialized CUDA kernel is then used to compute the desired costs directly: It first computes correlation costs via a dot-product between first- and the pooled second-frame image features in a local neighborhood based on the current flow estimate. Thereafter, it samples from this cost window to obtain the required cost values at the sub-pixel looked-up positions (*On-Demand Cost Sampling* in Fig. 5). Due to this modification, the extended coarse-to-fine scheme with the additional finer scale did not need extra memory for training or inference. However, both training and inference time were increased.

Compared to our 3-scale MS-RAFT model, the upsampled estimated flow at 1/2 the resolution at scale S_3 is now used as initialization for the estimation in the last scale, S_4. After performing T_4 recurrent updates at this scale and one last convex upsampling step, the flow is output at full resolution. Essentially, with this extension, unlike in MS-RAFT, the convex upsampler can now be effectively used to upsample the flow estimated in the last scale from half to full resolution.

4.2 Multi-Scale Feature Extraction

In this section, we elaborate on the feature extractor that computes the image and context features in different scales used in Sect. 4.1. It is illustrated in Fig. 6. On the left side, RAFT's feature encoder is shown, which has been extended to compute coarser scale features at 1/16 the resolution. This is realized by applying an additional residual unit, with stride two. Unlike RAFT's encoder, this encoder also outputs intermediate features at 1/4 and 1/2 the resolution. Note that the latter scale is not used in our 3-scale MS-RAFT model. The feature enhancement block on the right hand side is responsible for the aggregation of multi-scale features from coarser scales, providing higher-level features.

In contrast to (Hur and Roth, 2019; Sun et al., 2018), and (Wan et al., 2020), the outputs (intermediate features; *Int-featX* in Fig. 6) of the encoder on the left are not employed directly in the estimation process of Sect. 4.1, except for

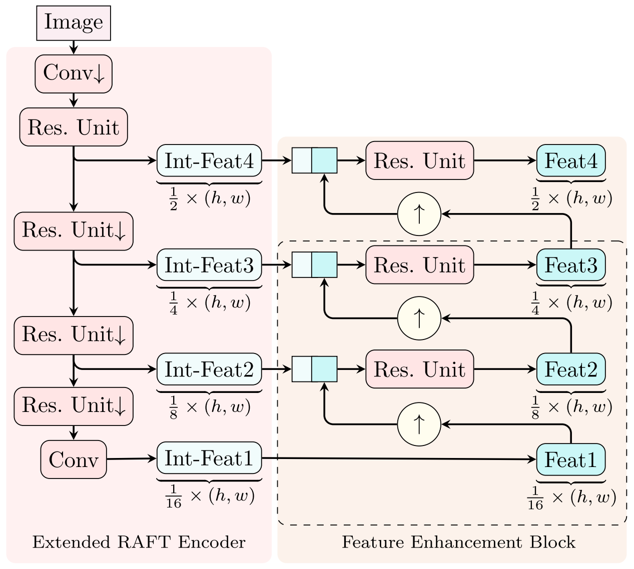
Image
Conv↓
Res. Unit
Res. Unit↓
Res. Unit↓
Res. Unit↓
Conv
Int-Feat4
$\frac{1}{2} \times (h, w)$
Int-Feat3
$\frac{1}{4} \times (h, w)$
Int-Feat2
$\frac{1}{8} \times (h, w)$
Int-Feat1
$\frac{1}{16} \times (h, w)$
Res. Unit
Res. Unit
Res. Unit
↑
↑
↑
Feat4
$\frac{1}{2} \times (h, w)$
Feat3
$\frac{1}{4} \times (h, w)$
Feat2
$\frac{1}{8} \times (h, w)$
Feat1
$\frac{1}{16} \times (h, w)$
Extended RAFT Encoder
Feature Enhancement Block

Fig. 6 U-Net-style feature extractor. The left side shows RAFT's feature encoder which is extended by an additional coarser scale. The feature enhancement sub-network is added to this encoder to generate multi-scale features. While the entire enhancement block is used in our 4-scale MS-RAFT+ approach, the dashed box shows the reduced block of our 3-scale MS-RAFT model

the coarsest scale. For other scales, similar to (Saxena et al., 2019), the intermediate finer features (*Int-feat s*) are combined with the deeper coarser features (*Feat s-1*) for enhancement. First *Feat s-1* is upsampled via bilinear interpolation to have the same spatial size as the corresponding finer scale intermediate features. *Int-feat s* and the upsampled *Feat s-1* are then stacked and processed via a residual block for feature consolidation. The cyan features on the right (*Feat s*) are the outputs of this module which are employed in the coarse-to-fine estimation scheme.

Similar to the U-Net architecture (Ronneberger et al. 2015), the number of channels increases as the coarser features are computed, and it decreases as finer features are constructed. Note that the same U-Net-style feature extractor is used to compute the image and context features, with the difference that for the context features, the depth of the output stays constant for all scales. This characteristic is necessary to allow sharing the update block among scales (see Sect. 4.1).

4.3 Multi-Scale Supervision

In the previous sections, we discussed the architecture of our method. Let us now elaborate on the loss function used for training the network.

As shown in Sect. 4.1, to estimate the flow update at each iteration of each scale, candidate costs are computed and sampled based on the current flow estimate. This means that the computation of each flow update is highly dependent on

[2] (Teed and Deng, 2020) released the code for on-demand cost computation for inference. We slightly adjusted the corresponding module. Moreover, for training, the implementation of the backward pass was added.

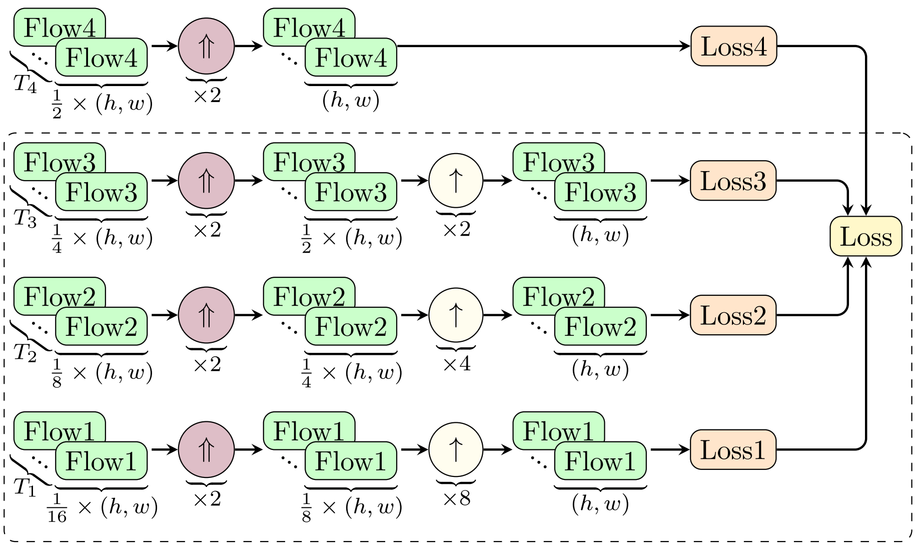
Flow4
Flow4
T_4
$\frac{1}{2} \times (h,w)$
×2
Flow4
Flow4
(h,w)
Loss4
Flow3
Flow3
T_3
$\frac{1}{4} \times (h,w)$
×2
$\frac{1}{2} \times (h,w)$
×2
(h,w)
Loss3
Loss
Flow2
T_2
$\frac{1}{8} \times (h,w)$
$\frac{1}{4} \times (h,w)$
×4
Loss2
Flow1
T_1
$\frac{1}{16} \times (h,w)$
$\frac{1}{8} \times (h,w)$
×8
Loss1

Fig. 7 Computation of the multi-scale multi-iteration loss. At each scale and each iteration, the flow is upsampled to the original resolution using a combination of learned convex upsampling ($\Uparrow$) and bilinear upsampling ($\uparrow$). Combining the losses for each iteration leads to the scale losses (*LossX*) which are further aggregated to the final loss. While the entire sketch describes the loss computation for our 4-scale MS-RAFT+ approach, the dashed box applies to the reduced loss for our 3-scale MS-RAFT model

the previous flow estimate, hence requiring each intermediate flow to be accurate. This suggests a direct supervision of the flow estimate at each iteration and on each scale. The sketch of our multi-scale multi-iteration loss is illustrated in Fig. 7. All T_s flow estimates at scale s contribute to the computation of the loss term of that scale. The total loss is computed as the sum of the loss terms of all scales.

Note that similar to RAFT (Teed and Deng 2020), the loss considers the flow at the original resolution at each iteration of each scale. First, the flow is upsampled using the convex upsampler as discussed in Sect. 4.1. For the first three scales, where the resulting flow after convex upsampling is still smaller than the original resolution, the flow is further upsampled to that resolution via bilinear interpolation. At the finest scale, since the flow is already computed at the full resolution after convex upsampling, no bilinear upsampling is needed.

Our overall multi-scale multi-iteration loss is given by

$$\mathcal{L} = \sum_{s=1}^{S} \sum_{t=1}^{T_s} \gamma_{s,t} \cdot L_{s,t}, \tag{2}$$

where S and T_s denote the number of scales and iterations per scale, respectively. Furthermore, $L_{s,t}$ represents the loss at scale s and iteration t. For pre-training the network, $L_{s,t}$ then reads

$$L_{s,t} = \frac{1}{M} \sum_{m=1}^{M} \sum_{p=1}^{N_{\text{px}}} \frac{\|\mathbf{f}^{s,t}_{m,p} - \mathbf{f}^{\text{gt}}_{m,p}\|_{\epsilon}}{N_{\text{px}}}, \tag{3}$$

where M represents the batch size, N_{px} indicates the number of pixels in the image, and $\mathbf{f}^{s,t}_{m,p}$ denotes the estimated flow at scale s and iteration t in sample m in the batch at pixel p at the original resolution. Moreover, $\mathbf{f}^{\text{gt}}_{m,p}$ denotes the ground truth flow of the same sample at pixel p. Finally, $\|\cdot\|_{\epsilon} = \sqrt{\|\cdot\|_2^2 + \epsilon}$ stands for the regularised L_2-norm with $\epsilon = 1 \times 10^{-5}$.

To steer the influence of the loss at each iteration of the coarse-to-fine scheme, we extend RAFT's original weighting strategy to multiple scales. The corresponding weight at scale s and iteration t is given by

$$\gamma_{s,t} = \gamma^{T_{\text{tot}} - T_{s,t}}, \tag{4}$$

where the total number of iterations over all scales is computed by

$$T_{\text{tot}} = \sum_{s=1}^{S} T_s \tag{5}$$

and the current iteration number at scale s with regard to all scales reads

$$T_{s,t} = \sum_{k=1}^{s-1} T_k + t. \tag{6}$$

Setting $\gamma < 1$ in Equation (2) enforces exponentially increasing weights for later iterations as well as for finer scales.

Similar to PWC-Net (Sun et al. 2018), we use a different loss function for fine-tuning our method. However, in contrast to PWC-Net, which employs a pixel-wise robust loss function, we consider sample-wise robustness via

$$L_{s,t}^{\text{fine}} = \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{p=1}^{N_{\text{px}}} \frac{\|\mathbf{f}_{m,p}^{s,t} - \mathbf{f}_{m,p}^{\text{gt}}\|_{\epsilon}}{N_{\text{px}}} + \epsilon' \right)^{q} , \tag{7}$$

where we choose $q = 0.7$ and $\epsilon' = 0.01$.

Penalizing per-pixel deviations from the ground truth in a sub-quadratic manner via $\| \cdot \|_{\epsilon}$ already increases pixel-wise robustness by down-weighting outlier pixels (Huber 2004). In addition, our modification allows to down-weight the impact of an entire training sample in the batch, if the sample produces a large error, i.e., contains many outliers. In approaches such as FlowNet2.0 and PWC-Net (Sun et al. 2018; Ilg et al. 2017) difficult training samples were excluded explicitly from training. Our sample-wise loss can be hence interpreted as an implicit attenuation of difficult samples during fine-tuning. The usefulness of this strategy can be seen by our experimental evaluations in the next chapter.

5 Evaluation

In this section, we first detail on training and inference. Then, we investigate the accuracy of our MS-RAFT and MS-RAFT+ approach using the four most widely used benchmarks: KITTI 2015, MPI Sintel, Middlebury, and VIPER. In this context, we also discuss the performance of our MS-RAFT+ method in the Robust Vision Challenge 2022. Finally, we ablate architectural design choices and analyze different fine-tuning settings.

5.1 Training

In the following, we elaborate on the training of our two models, MS-RAFT and MS-RAFT+.

5.1.1 Detailed Settings

Our MS-RAFT and MS-RAFT+ models have 13.5M and 16 M parameters, respectively. The parameter overhead compared to the 5.3M parameters of RAFT is only due to the multi-scale U-Net-style feature extractor, as the update block and the learned upsampler are shared among scales. During training, we performed $(T_1, T_2, T_3) = (4, 6, 8)$ recurrent iterations for MS-RAFT and $(T_1, T_2, T_3, T_4) = (4, 5, 5, 6)$ iterations for MS-RAFT+, going from the coarsest to the finest scale. Increasing the number of iterations per scale could potentially improve results; however, this would make the training process more expensive, both time- and memory-wise.[3]

The image features of the coarsest to finest scale have 256, 128 and 96 channels for MS-RAFT, and 256, 128, 96 and 64 channels for MS-RAFT+, respectively. Context features have 256 channels for both models at all scales.

By default, we employ two hierarchical correlation levels on each coarse-to-fine scale for both of our models, where we look up the costs using a radius of four pixels. In MS-RAFT, this is done by directly constructing a two-level correlation pyramid, whereas in MS-RAFT+, we pool the second-frame image features once to obtain a hierarchical representation thereof, which is then used for the on-demand cost computation and sampling. We chose the same cost lookup range for both models, as both have their coarsest scale at $(h/16, w/16)$. Moreover, we chose this range based on the coarsest scale in such a way that, expectedly, the majority of match candidates lie inside its perceptive range.

To supervise the model during training, we used our multi-scale multi-iteration loss. We employed the L2 loss from Equation (3) for pre-training and the robust sample-wise loss from Equation (7) with $q = 0.7$ for fine-tuning. Regarding the weight of each loss at different iterations and scales, we set $\gamma = 0.8$ in Equation (2) for pre-training and $\gamma = 0.85$ for fine-tuning, similar to (Teed and Deng, 2020).

5.1.2 Training Phases

We trained both our models in three stages: First, we pre-trained the networks on Chairs (Dosovitskiy et al. 2015) and then on Flying Things3D (Mayer et al. 2016)—referred to as *Things* here, each for 100K iterations using batch sizes of 10 and 6, respectively, similar to RAFT (Teed and Deng 2020).

In the case of our MS-RAFT model we fine-tuned the network with the same mixture of datasets introduced in RAFT (Teed and Deng 2020) for 100K steps with batch size of 6. The mixed set includes 71% MPI Sintel (Butler et al. 2012), 13.5% KITTI 2015 (Menze and Geiger 2015), 13.5% Things, and 2% HD1K (Kondermann et al. 2016).

In the case of our MS-RAFT+ model, as we participated in the Robust Vision Challenge (RVC) 2022, we fine-tuned the network with focus on a more uniform distribution of datasets, including the VIPER dataset (Richter et al. 2017), as its benchmark results were also considered in this challenge. Therefore, we fine-tuned MS-RAFT+ for 100K iterations with batch size of 6 using a combination of datasets, where 30% of the training samples consist of MPI Sintel, 30% of VIPER (every 10th frame), 28% of KITTI 2015, 10% of Things, and 2% of HD1K.[4]

Note that even though the outcome of the Middlebury benchmark (Baker et al. 2011) was also taken into account

[3] This is due to longer sequential back-propagation steps.

[4] Similar to the proportions of MPI Sintel and VIPER, we used overall 30% of the samples for real-world KITTI-like sequences (28% KITTI 2015 + 2% HD1K).

in the RVC, we did not use any samples from its dataset for training as there are only eight frame pairs. In our pre-training we used the same augmentation strategies suggested by RAFT for pre-training, while for our fine-tuning on the mixed dataset, we employed RAFT's augmentations for fine-tuning on MPI Sintel.

Hardware Requirements

We used two Nvidia A100 GPUs with 40GB VRAM each to train our models. The overall training process took around six days for MS-RAFT+, and three days for MS-RAFT with the same hardware. The extended training time of the former was due to processing of the additional higher resolution and the on-demand computation of the matching cost, which memory-wise allowed us to use the fourth scale, i.e., the additional finer scale.

5.2 Inference

Recent RNN-based methods typically use more iterations to refine the flow at inference time than they use during training. Furthermore, they often use a different number of iterations for each dataset to achieve better results. For example, RAFT (Teed and Deng 2020) was trained with 12 recurrent iterations, but applied 32 and 24 for inference on MPI Sintel and KITTI 2015, respectively. For our MS-RAFT model, we similarly applied two different sets of iteration numbers for MPI Sintel and KITTI 2015 with $(T_1, T_2, T_3) = (10, 15, 20)$ and $(T_1, T_2, T_3) = (4, 8, 24)$, respectively.

In the case of MS-RAFT+, however, as we had to meet the requirements of the Robust Vision Challenge, we computed the optical flow using the same number of iterations for all the aforementioned benchmarks. To this end, we used $(T_1, T_2, T_3, T_4) = (4, 6, 5, 10)$ iterations, which we found to work best on the validation sets we employed for our fine-tuning ablations. More details on the validation sets can be found in Sect. 5.4.

Regarding the flow initialization in coarse-to-fine schemes (cf. Sect. 4.1), typically a zero flow is used as starting point at the coarsest scale (Sun et al. 2018). However, as shown in single-scale RAFT (Teed and Deng 2020), the estimation can benefit from the predictions of the previous time-step by forward-warping the corresponding result and using it as improved initialization for the current time step. This so-called warm-start strategy used by (Teed and Deng, 2020) considers the forward-warped predicted flow from the previous time-step in a recursive manner, i.e., the flow of the previous time step was also initialized by the forward-warped version of its previous pair's flow.

To make the sequence-dependency of each estimated flow shorter during inference, we applied a cold warm-start strategy, in which only the previous image pair is considered for initialization. For example, to compute the flow between frame I_3 and frame I_4, we initialize the flow by forward-warping the flow between frame I_2 and frame I_3, for which, in turn, the zero flow was used as initialization. In cases where the previous frame is not contained in the standard dataset files, or when the first image-pair in a sequence is considered, we use a zero-flow initialization—as in (Teed and Deng, 2020). Using a cold warm-start initialization makes our computed flow only dependent on at most three input frames, compared to the warm-start strategy of RAFT, where the output is dependent on all previous images in the sequence due to its recursive way of initialization.

5.3 Results

In the following we evaluate the performance of our 4-scale MS-RAFT+ approach. To this end, we investigate the accuracy of our method on the KITTI 2015, the MPI Sintel, the Middlebury, and the VIPER benchmarks, thereby also comparing its performance to the recent literature. In this context, we also analyze the visual quality of the computed flow fields. Moreover, we also compare the results of both our 3-scale MS-RAFT and our 4-scale MS-RAFT+ model. Finally, we will discuss the performance of our MS-RAFT+ method in the Robust Vision Challenge 2022. Please note that in this section the best and second best results are boldfaced and underlined, respectively.

5.3.1 KITTI 2015

Let us start by investigating the performance of our two models on the KITTI 2015 benchmark. To this end, we compare their results in Table 1 to the top-ten published optical flow methods. As can be observed, our MS-RAFT+ method achieves Rank 1 on both (Fl-all) and (Fl-all non-occluded) measures among all published optical flow methods, respectively.

Notably, our method scores a new state of the art in non-occluded regions, in which, it is even more accurate than the currently leading published scene flow method—CamLiFlow (Liu et al. 2022)—that achieves an Fl-all error of 2.40% compared to 2.18% of our approach. Moreover, our MS-RAFT+ approach shows substantial improvements over its underlying RAFT baseline, with the largest gain of 32.4% in the foreground of non-occluded regions. In addition, it improved over our 3-scale MS-RAFT model with a gain of 22.1% (Fl-all) within non-occluded regions. A comparison of qualitative results of our MS-RAFT+ approach and DIP (Zheng et al. 2022), FlowFormer (Huang et al. 2022), GMFlow (Xu et al. 2022), and RAFT (Teed and Deng 2020) is presented in Fig. 8. Evidently, our method allows to preserve fine structural details of the scene such as traffic signs and cars.

Table 1 Comparison to the literature for the *KITTI 2015* benchmark

	KITTI 2015 (Test)			KITTI 2015 NOC (Test)		
Method	bg	fg	all	bg	fg	all
MS_RAFT+ (ours)	**3.83**	5.71	**4.15**	**2.07**	**2.69**	**2.18**
DIP (Zheng et al., 2022)	3.86	5.96	4.21	2.31	3.00	2.43
RAFT-it (D. Sun et al., 2022)	4.11	**5.34**	4.31	2.68	2.77	2.70
RAFT-OCTC (Jeong et al., 2022)*	-	-	4.33	-	-	-
SeparableFlow (Zhang et al., 2021)	4.25	5.92	4.53	2.56	3.75	2.78
KPA-Flow (Luo et al., 2022)	4.17	6.77	4.60	2.59	3.83	2.82
FlowFormer (Huang et al., 2022)	4.37	6.18	4.68	2.54	3.38	2.69
RAFT-A (D. Sun et al., 2021)	4.54	5.99	4.78	3.01	3.17	3.04
CRAFT (Sui et al., 2022)	4.58	5.85	4.79	2.87	3.68	3.02
MS_RAFT (ours)	4.58	6.38	4.88	2.66	3.41	2.80
RAFT (Teed & Deng, 2020)	4.74	6.87	5.10	2.87	3.98	3.07
MS-RAFT over RAFT	3.4 %	7.1 %	4.3 %	7.3 %	14.3 %	8.8 %
MS-RAFT+ over RAFT	19.0 %	16.9 %	18.6 %	27.9 %	32.4 %	29.0 %
MS-RAFT+ over MS-RAFT	16.4 %	8.9 %	15.0 %	22.2 %	21.1 %	22.1 %

Results for the Top 10 published optical flow methods given in terms of the standard Fl outlier ratio in percent; cf. (Menze and Geiger, 2015). NOC = non-occluded pixels. bg = background. fg = foreground

* For the sake of reproducibility the value of the underlying peer-reviewed publication is listed. Please note that the benchmark shows improved results, which, however, are not documented (no description, no code, no settings)

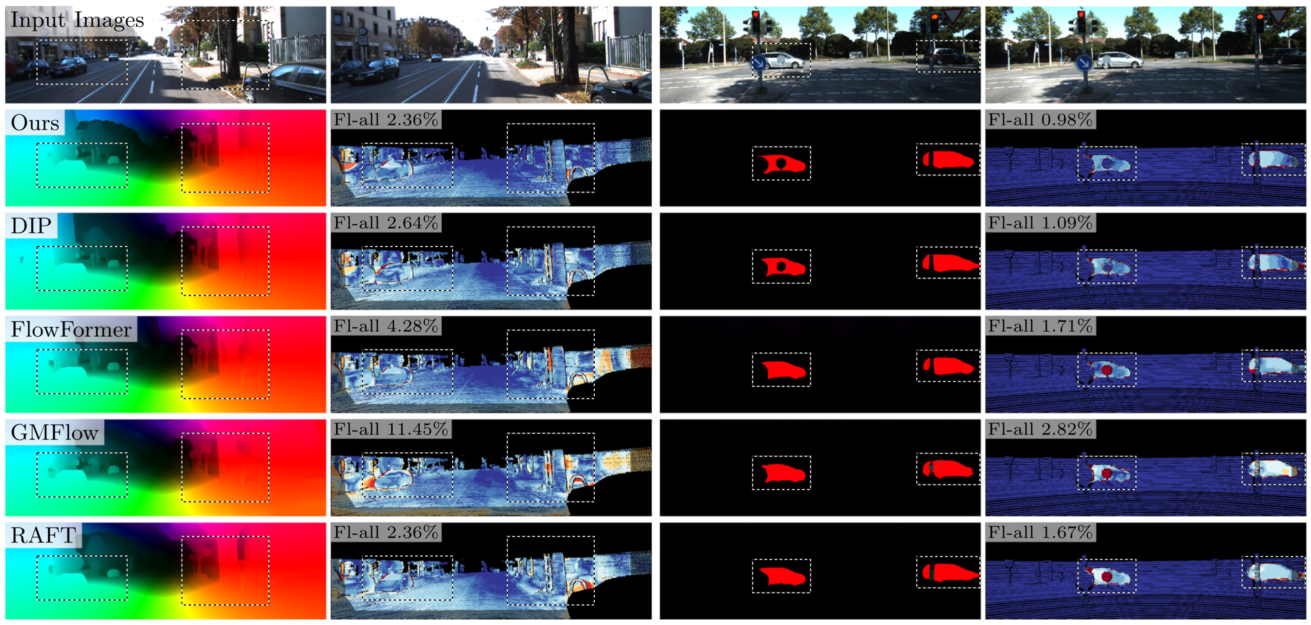

Fig. 8 Visual comparison to other methods from the literature for the *KITTI 2015* benchmark. **First row.** Two exemplary frame pairs. **Other rows.** Computed flow field and error map of the first and second frame pair. Note the differences in the marked locations. Best viewed electronically

5.3.2 MPI Sintel

Considering the MPI Sintel benchmark, results are shown in Table 2. Also here we compare our methods to the top ten published results from the literature. In this case, MS-RAFT+ achieves Rank 2 and Rank 9 on the clean and final pass (EPE-all), respectively. Our method performs again particularly well in non-occluded regions (in the clean pass). Moreover, it shows very good results for small displacements (both final and clean), where it achieves Rank 1 in the corresponding metrics. Further, compared to its underlying RAFT approach, our MS-RAFT+ model gains a 24.2% accuracy improvement for the EPE-all and 35.5% in the non-occluded regions as well as a substantial 50% in small displacements in the clean

Table 2 Comparison to the literature for the *MPI Sintel* benchmark

Method	Sintel Clean (Test) All	Mat	Unmat	s0-10	Sintel Final (Test) All	Mat	Unmat	s0-10
FlowFormer (Huang et al., 2022)	**1.20**	0.41	7.63	0.25	**2.12**	**0.99**	**11.37**	0.52
MS_RAFT+ (ours)	1.23	**0.40**	8.02	**0.16**	2.68	1.28	14.12	**0.42**
SKFlow (S. Sun et al., 2022)	1.30	0.57	**7.25**	0.28	2.26	1.14	11.42	0.58
MS_RAFT (ours)	1.37	0.48	8.68	0.22	2.67	1.19	14.71	0.47
GMA (Jiang et al., 2021)	1.39	0.58	7.96	0.33	2.47	1.24	12.50	0.57
GMFlowNet (Zhao et al., 2022)	1.39	0.52	8.49	0.31	2.65	1.27	13.88	0.70
RAFT-OCTC (Jeong et al., 2022)	1.42	0.54	8.57	0.30	2.57	1.24	13.44	0.58
GMA-FS (Im et al., 2022)	1.43	0.60	8.17	0.33	2.44	1.20	12.55	0.59
AGFlow (Luo et al., 2022)	1.43	0.56	8.54	0.32	2.47	1.22	12.64	0.56
DIP (Zheng et al., 2022)	1.44	0.52	8.92	0.34	2.83	1.28	15.49	0.57
RAFT (Teed & Deng, 2020)	1.61	0.62	9.65	0.34	2.86	1.41	14.68	0.63
MS-RAFT over RAFT	14.9 %	22.6 %	10.1 %	25.3 %	6.6 %	15.6 %	−0.2 %	25.4 %
MS-RAFT+ over RAFT	24.2 %	35.5 %	16.9 %	50.0 %	6.3 %	9.2 %	3.8 %	33.3 %
MS-RAFT+ over MS-RAFT	10.3 %	16.5 %	7.6 %	28.1 %	−0.4 %	−6.8 %	4.0 %	10.2 %

Results for the Top 10 published optical flow methods in terms of the standard EPE measure in pixels; cf. (Butler et al., 2012). Mat = matched. Unmat = unmatched. s0-10 = small displacements between 0 and 10 pixels

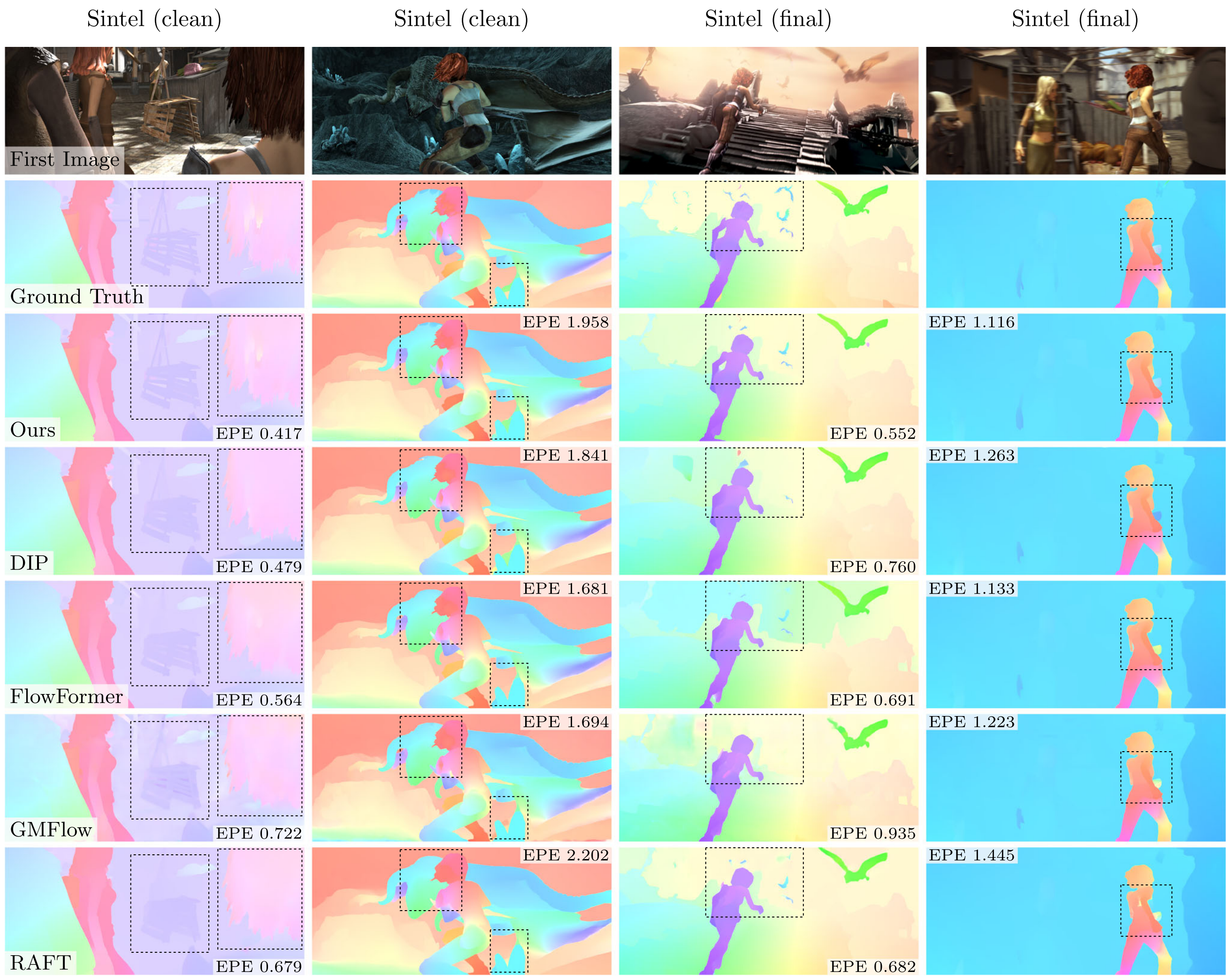

Fig. 9 Visual comparison to other methods from the literature for the *MPI Sintel* benchmark. **First row.** Four exemplary frames. **Other rows.** Computed flow fields. Note the differences in the marked locations. Best viewed electronically

Table 3 Comparison to the literature for the *Middlebury* benchmark

	Middlebury (Test)							
Method	Army	Meqn	Schef	Wood	Grov	Urbn	Yosem	Teddy
RAFT-it (D. Sun et al., 2022)	**0.07**	**0.15**	**0.17**	**0.05**	0.46	**0.09**	<u>0.06</u>	<u>0.28</u>
NNF-Local (Chen et al., 2013)	**0.07**	**0.15**	<u>0.18</u>	0.10	<u>0.41</u>	0.23	0.10	0.34
MS_RAFT+ (ours)	**0.07**	0.22	0.20	<u>0.07</u>	**0.39**	<u>0.11</u>	**0.05**	**0.27**
PMMST (Zhang et al., 2020)	0.09	0.18	0.21	0.10	0.51	0.24	0.11	0.37
OFLAF (Kim et al., 2013)	0.08	0.16	0.19	0.14	0.51	0.31	0.11	0.42
MDP-Flow2 (L. Xu et al., 2011)	0.08	**0.15**	0.20	0.15	0.63	0.26	0.11	0.38
NN-field (Chen et al., 2013)	0.08	0.17	0.19	0.09	0.41	0.52	0.13	0.35
CoT-AMFlow (Wang et al., 2020)	0.08	**0.15**	0.21	0.16	0.67	0.27	0.12	0.42
TC/T-Flow (Volz et al., 2011)	**0.07**	0.19	0.28	0.14	0.67	0.22	0.11	0.50
WLIF-Flow (Tu et al., 2016)	0.08	0.18	0.25	0.14	0.61	0.43	0.13	0.51

Results for the Top 10 published optical flow methods in terms of the standard AAE (=EPE) measure in pixels; cf. (Baker et al., 2011)
Meqn: Mequon, Schef: Schefflera, Wood: Wooden, Grov: Grove, Urbn: Urban, Yosem: Yosemite

pass. In the final pass, the improvements compared to RAFT are also large, but less than in the clean pass. Comparing the outcome of both our models, MS-RAFT+ improves over MS-RAFT by 10% EPE-all, 16.5% in the non-occluded regions and 28.1% for small displacements in the clean pass. However, in the case of the final pass, no substantial changes can be observed for EPE-all, while the results in the non-occluded regions actually degrade by 6.8%. Qualitative results shown in Fig. 9 demonstrate again that small structural details are well preserved by our MS-RAFT+ approach. Moreover, one can see the benefit of operating on 1/2 the resolution compared to DIP and GMFlow, which work at most on 1/4 the resolution, preserving even finer details comparably.

Please note that the coarse-to-fine strategy within our approach enables mainly a more accurate estimation at higher resolutions benefiting from useful initializations from coarser scales rather than the estimation of large displacements which are already handled reasonably well by the hierarchical non-local lookup of the matching costs. Hence, we mainly observe strong improvements for small displacements.

5.3.3 Middlebury

In the Middlebury benchmark, as can be seen in Table 3, our MS-RAFT+ method ranks third after (Sun et al., 2022) and (Chen et al., 2013). In this context, it achieves Rank 1 on the sequences Army, Grove, Yosemite, and Teddy, as well as Rank 2 on the sequences Wooden and Urban. Interestingly, even without considering the corresponding training data, the approach performs favourably against classical methods with hand-tuned models and parameters.

5.3.4 VIPER

Finally, we evaluated our MS-RAFT+ method on the VIPER benchmark. The results are shown in Table 4. As the comparison to the top five published methods of the literature shows, our MS-RAFT+ approach achieves Rank 1 overall. By clearly outperforming all other methods in all categories (day, sunset, rain, snow, night), it sets a new state-of-the-art for this benchmark. The resulting optical flow of our method for a few samples of this benchmark is visualized in Fig. 10. As can be observed, our method achieves excellent results in samples with different conditions and still produces very detailed and accurate results; see the different motion estimated in cars' wheels. For example in the third example from the left in the first row, in spite of reflections, different textures on the road and occlusions, our method produces highly accurate flow fields. Also in the first, second and fourth examples of the second row, regardless of shadows and combination of textured and texture-less regions our method achieves an excellent outcome.

Table 4 Comparison to the literature for the *VIPER* benchmark

	VIPER (Test)					
Method	Mean	Day	Sunset	Rain	Snow	Night
MS_RAFT+ (ours)	**78.6**	**84.3**	**80.6**	**71.3**	**80.1**	**76.6**
GMA (Jiang et al., 2021)	<u>74.3</u>	<u>79.0</u>	<u>77.0</u>	<u>66.4</u>	<u>77.0</u>	<u>71.9</u>
CRAFT (Sui et al., 2022)	73.8	78.7	76.5	66.1	76.5	71.1
IRR-PWC (Hur & Roth, 2019)	68.9	76.2	71.2	60.5	72.2	64.3
VCN (Yang & Ramanan, 2019)	60.9	71.0	62.2	53.9	61.4	56.0
PWC-Net (D. Sun et al., 2020)	59.9	70.6	61.7	51.8	60.7	54.9

Results for the Top 6 published optical flow methods in terms of the WAUC quality measure in percent; cf. (Richter et al., 2017)

5.3.5 Robust Vision Challenge

As mentioned before in this chapter, we participated in the ECCV 2022 Robust Vision Challenge (RVC) (Zendel et al. 2022) with our MS-RAFT+ method. The final ranking is shown in Table 5. Note that the RVC ranking also considers different measures in each individual benchmark. Achieving

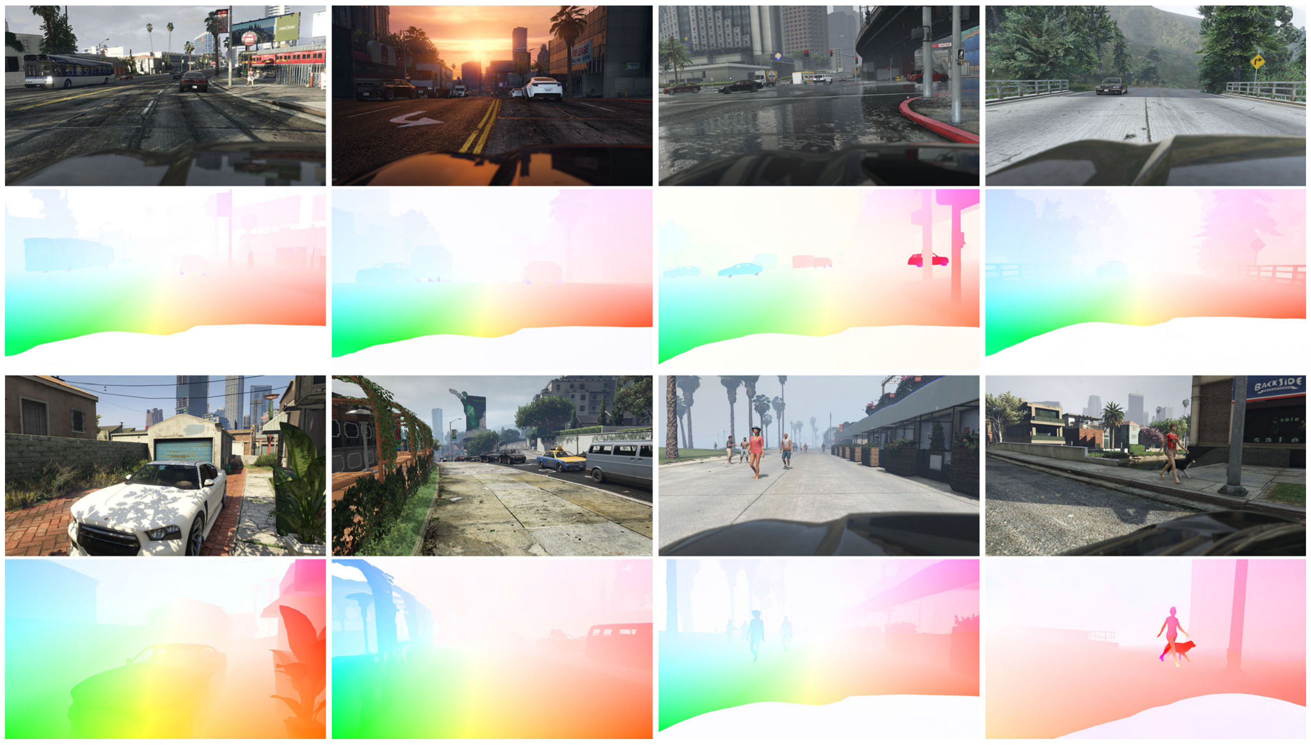

Fig. 10 Visual examples for the *VIPER* benchmark. **First and third row.** Four exemplary frames. **Second and fourth row.** Computed flow fields. Note the structural details. Best viewed electronically

Table 5 Robust Visions Challenge 2022

Rank	Method	Middle-bury	KITTI 2015	MPI Sintel	VIPER
1	**MS_RAFT+**	2	2	2	1
2	RAFT-it+	1	1	4	2
3	GMFlow	3	3	1	3
4	MCPFlow	5	4	3	5
5	RAFT-TF	4	5	5	4

All submitted methods have the suffix "_RVC" in their actual submission titles

Rank 2 for Middlebury, KITTI, and Sintel, as well as Rank 1 for VIPER, MS-RAFT+ shows a good generalization performance across all benchmarks. In the final leaderboard of the RVC, this results in Rank 1 among all participating methods (see Table 5) thus winning the Robust Vision Challenge 2022 for the optical flow task.

5.4 Ablations

In the following, we present ablations for design choices of our 3-scale MS-RAFT method, which we directly considered in our high-resolution MS-RAFT+ model. Regarding multi-scale concepts, where the experiments were feasible to run, we present additional experiments demonstrating that the selected strategies still yield best results in the 4-scale

Table 6 Coarsest scale vs. look-up levels

Resolution Scales	Iters	Look-Up Levels	Sintel (train) Clean	Final
$[\frac{1}{8}, \frac{1}{4}]$	[8, 10]	3	1.16	3.07
		4	1.14	2.64
		5	**1.11**	2.66
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}]$	[4, 6, 8]	2	<u>1.13</u>	<u>2.60</u>
		3	1.15	2.66
		4	1.14	2.70
$[\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}]$	[3, 3, 5, 7]	1	1.19	**2.53**
		2	1.16	2.67
		3	1.22	2.67

Table 7 Hidden-state initialization strategy

Hidden-state Initialization	Sintel (train) Clean	Final
Each scale	**1.13**	**2.60**
Coarsest scale	1.15	2.87
Mixed	<u>1.14</u>	<u>2.62</u>

Table 8 Single-scale vs. multi-scale

Resolution Scales	Iters	MS Feats	Sintel (train) Clean	Sintel (train) Final
Single-Scale Estimation				
$[\frac{1}{8}]$	12	-	1.40	2.67
	18	-	1.45	2.70
$[\frac{1}{4}]$	12	-	1.58	3.10
	18	-	1.52	3.08
$[\frac{1}{4}]$	18	✓	1.64	3.31
Multi-Scale Estimation				
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}]$	[4, 6, 8]	✓	<u>1.13</u>	<u>2.60</u>
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}]$	[4, 5, 5, 6]	✓	**0.99**	**2.56**

Table 9 U-Net-style vs. standard features

Resolution Scales	Features	Sintel (train) Clean	Sintel (train) Final
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}]$	U-Net-style	1.13	<u>2.60</u>
	Standard	1.11	2.68
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}]$	U-Net-style	**0.99**	**2.56**
	Standard	<u>1.03</u>	2.67

setting and their results are analogous with their 3-scale counterparts. All the aforementioned ablations consider the pre-trained model on Chairs and Things, each for 100K iterations with a batch size of 10 and 6, respectively. We evaluated our pre-trained models on MPI Sintel training data, using the warm-start strategy. For each architecture, we used the same number of recurrent iterations per scale during inference, so that the results of the same architecture are comparable. After ablating on the architectural aspects, we compare different dataset mixes for fine-tuning our method. Finally, we perform a comparison of different fine-tuning loss functions. Note that in each ablation, the selected setting is highlighted in the corresponding tables.

5.4.1 Basic Design Choices

In this section, we ablate on two aspects using our 3-scale MS-RAFT model. First we investigate the impact of the number of correlation pyramid levels and the resolution of the coarsest scale on the accuracy. We then compare three different strategies for re-initializing the hidden-state of the recurrent unit, used in estimating the flow.

Number of Correlation Pyramid Levels vs. Coarsest Scale Resolution

In the first ablation, we investigate the interplay between the number of correlation pyramid levels and the resolution of the coarsest scale, i.e., the starting scale. To this end, we considered a search range of four and compared different starting scales with a different number of pyramid lookup levels. The corresponding results can be found in Table 6. While the best results for the clean and final pass are associated with the two-scale setting using 5 pyramid levels, and the four-scale method using a single pyramid level, respectively, the best trade-off for both passes corresponds to the 3-scale setting using two pyramid levels. Note that, to make all three models relatively comparable, we used the same overall number of recurrent iterations over all scales. The number of iterations for the two-scale, three-scale and four-scale models are given by $8+10$, $4+6+8$ and $3+3+5+7$, respectively. In the three and four-scale variants, using more, i.e., additional coarser correlation pyramid levels yields an average degradation of accuracy. As the coarsest scale becomes smaller, one should not have a large search range, as many lookups might lie outside the domain of the corresponding features of the second image. Note that both MS-RAFT and MS-RAFT+ use the same coarsest resolution of $(h/16, w/16)$. Therefore, we keep using two correlation pyramid levels per scale also in their case.

Hidden-State Re-Initialization

As explained in Sect. 4, at each scale a part of the context features is used to re-initialize the hidden-state of the recurrent unit, while the hidden-state gets updated individually at each scale. In Table 7 we compare three different re-initialization strategies. The first row shows the scheme that we mentioned in Sect. 4, where a part of the current scale's context features is assigned to the hidden-state of the recurrent unit in the first iteration of that scale. The second row represents the result for passing the upsampled updated hidden-state of the coarsest scale. By doing so, the information learned over the iterations on the coarser scales is no longer forgotten, but at the same time, no new information from finer scales is introduced. A third viable option is combining the learned updated hidden-state with the finer context features, where the previous updated hidden-state is upsampled first and added to the context features of the current scale. As the results in Table 7 show, initializing the hidden-state per scale yields the best results, closely followed by the mixed setting. Hence for re-initializing the hidden-state of the recurrent unit in both our model, we used the first option, i.e., per-scale re-initialization.

5.4.2 Multi-Scale Design Choices

In this section, we ablate our multi-scale concepts. First, we ablate the multi-scale estimation scheme. We then compare the standard feature extractor, i.e., the extended encoder

Table 10 Multi-scale supervision

Resolution Scales	Loss		Sintel (train)	
	MS	MI	Clean	Final
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}]$	✓	✓	<u>1.13</u>	<u>2.60</u>
	✓	-	1.28	2.87
	-	✓	2.26	4.09
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}]$	✓	✓	**0.99**	**2.56**
	✓	-	1.24	2.89
	-	✓	1.66	3.42

MS = multi-scale loss. MI = multi-iteration loss

Table 11 Ending (finest) scale

Resolution Scales	Iters	Sintel (train)	
		Clean	Final
$[\frac{1}{16}, \frac{1}{8}]$	[8, 12]	1.54	2.96
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}]$	[4, 7, 9]	1.11	<u>2.60</u>
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}]$	[4, 5, 5, 6]	**0.99**	**2.56**
$[\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1]$	[4, 4, 4, 4, 4]	<u>1.04</u>	2.62

of RAFT in Sect. 4.2, and our U-Net-Style feature extractor. Afterwards, we compare our multi-scale multi-iteration loss to multi-scale single-iteration and single-scale multi-iteration settings. And finally, we investigate different ending scales, i.e., finest resolutions in a multi-scale setting.

Multi-Scale Estimation

Can the improvement of our approach only be due to high-resolution processing? Table 8 shows the evaluation results of different single-scale and both our multi-scale models. As can be seen from the results, operating on a single scale at $1/4$ the resolution compared to $1/8$, yields a degradation of the accuracy, even when the number of correlation pyramid levels is chosen to resemble the same lookup range as in RAFT. This behaviour can be seen for both models trained with 12 and 18 recurrent iterations. Let us now consider the integration of our advanced multi-scale features (see Sect. 4.2) in the single-scale settings to have a fairer comparison to our multi-scale models. However, using multi-scale features in a single scale estimation setting at $1/4$ resolution does not lead to improvements either. Only our multi-scale models that rely on a coarse-to-fine estimation yield improvements of the results.

Multi-Scale Feature Extraction

Table 9 shows a comparison between using our U-Net-Style feature extractor and a standard feature extractor that extends RAFT's feature encoder in a straightforward way; see the left-hand side of Fig. 6. From both sets of entries in the table, the advantage of using our U-Net-Style feature extractor becomes explicit—especially for the more challenging final pass.

Multi-Scale Loss

As discussed in Sect. 4.3, we supervise our network via a multi-scale multi-iteration loss. Table 10 compares our loss to two other variants for both 3-scale and 4-scale models: a single-scale multi-iteration loss and a multi-scale single-iteration loss. As explained in Sect. 4.3, in our multi-scale multi-iteration loss, we apply the loss in each iteration of each scale with increasing weights for later iterations and finer scales. In the case of using the single-scale multi-iteration loss, we apply the loss only to all the iterations of the finest scale, using increasing weights for later iterations,[5] while for multi-scale single-iteration cases, we consider the loss only for the last iteration of each scale employing increasing weights for finer scales.[6] Interestingly, applying the multi-scale single-iteration loss yields clearly better results over the single-scale multi-iteration loss, while it is still inferior compared to our multi-scale multi-iteration supervision.

Finest Scale

In Table 11 we investigated ending at different resolution scales. Importantly, all the models used the same sum of recurrent iterations over all scales during training. For the 2-scale to 5-scale cases, we used $8+12$, $4+7+9$, $4+5+5+6$ and $4+4+4+4+4$ recurrent iterations from the coarsest to the finest scale, respectively. Note that for all models, we applied a shared $\times 2$ learned convex upsampler to upsample the flow between scales for initializing the next finer scale and additionally once in the last scale. If the resolution of the final upsampled flow in the last scale was different than the original resolution, we bilinearly interpolate the flow to the original resolution. We used this procedure to have the most comparable architectures and most similar training of individual components among all cases in Table 11. As can be observed from the results, the 4-scale variant yields the highest accuracy. Therefore we chose $1/2$ the resolution as the finest resolution in our 4-scale extended method. Note that using different recurrent iterations per scale affects the outcome and typically adding more recurrent iterations—at least up to some extent—especially at the finer scales might improve the results. Using more recurrent iterations during training, however, makes the training process in general time- and memory-wise more expensive. Therefore we restricted our investigations to 20 recurrent iterations over all scales.

[5] We use the same weights as in our multi-scale multi-iteration loss, with the difference that all the loss terms except for the ones from the last finest scale are discarded.

[6] Similarly, we use the same weights in this case as our multi-scale multi-iteration loss at the final iteration per-scale.

Table 12 Composition of fine-tuning data

Training Data Split (in %)										
K	S	V	H	T	RFK	SF	used in	KITTI (split)	VIPER (validation)	Middlebury (train)
13.5	71.0	0.0	2.0	13.5	0.0	0.0	RAFT MS-RAFT	5.54	70.30	**0.186**
33.0	33.5	33.5	0.0	0.0	0.0	0.0	-	5.15	<u>81.15</u>	0.211
28.0	30.0	30.0	2.0	10.0	0.0	0.0	MS-RAFT+	**5.07**	**81.25**	0.201
20.0	30.0	30.0	0.0	10.0	10.0	0.0	-	5.53	81.08	<u>0.197</u>
26.0	28.0	28.0	2.0	8.0	0.0	8.0	-	<u>5.08</u>	81.05	0.206

K = KITTI 2015. S = MPI Sintel. V = VIPER. H = HD1K. T = FlyingThings. RFK = RealFlow-KITTI. SF = SlowFlow

Table 13 Fine-tuning loss

Loss	KITTI (split)	VIPER (validation)	Middlebury (train)
L1	5.68	79.94	**0.201**
L2 ($q = 1$)	5.30	80.66	0.205
L2 Rob ($q = 0.55$)	5.11	<u>81.06</u>	<u>0.204</u>
L2 Rob ($q = 0.7$)	<u>5.07</u>	**81.25**	**0.201**
L2 Rob ($q = 0.85$)	**5.06**	80.58	<u>0.204</u>

5.4.3 Fine-Tuning Setup

Previously, we ablated our architectural choices. In this section, we investigate how to compose a suitable mixed dataset for fine-tuning to perform well on all four datasets evaluated in the Robust Vision Challenge. Afterwards, we ablate the fine-tuning loss function.

Essentially, since we now investigate concepts regarding fine-tuning, where the MPI-Sintel dataset is included in the training, the same validation procedure as in the previous ablations, where a pre-trained model is validated on MPI-Sintel training data, is not valid anymore. Hence, in the following, we validate the fine-tuned models on the validation set of VIPER (every 10th frame), the training dataset of Middlebury (which is not used for training), and our own KITTI 2015 validation split making up 25% of the original KITTI 2015 training dataset.[7] In this context, we do not consider a split of MPI Sintel, as splitting this dataset into a representative train and validation subset is not trivial.

Please note that there are two different training scenarios when ablating the fine-tuning setup. When validating on VIPER and Middlebury, we considered the entire KITTI 2015 training data set for training. In this case we fine-tune the model with 100K iterations. However when validating on our KITTI 2015 split, we only train with the respective training subset. In this case, we only fine-tune for 75K iterations since this subset only contains 75% of the original training set.

Thereby we make sure that the same relative proportion of KITTI 2015 samples compared to the other datasets is used, in comparison with the case where the whole KITTI 2015 dataset is used. When evaluating our models, we computed the WAUC[8] (weighted area under the curve), the Fl error (percentage of outliers) and the EPE measure to validate the model on VIPER, KITTI 2015 and Middlebury, respectively.

Finally, due to time-constraints, instead of our 4-scale MS-RAFT+ model, we used the 3-scale MS-RAFT model to ablate the fine-tuning settings. Using the settings that were shown most effective in this model also showed excellent performance on our extended MS-RAFT+ approach.

Fine-Tuning Dataset

In Table 12, we compare the impact of using different proportions of different datasets on the outcome of the mentioned validation sets. Experimentation in this case is highly non-trivial, as introducing a proportion of a dataset means having less proportion of others. The first row represents the mixed data proportions used in fine-tuning RAFT (Teed and Deng 2020). Not surprisingly, using this data-mix yields the least accurate results on VIPER, as no VIPER data was used in this mixed-set, however it yields the best Middlebury results among all settings. In the second entry, we investigated a more uniform distribution of KITTI 2015, MPI Sintel and VIPER, while in the third entry we also used 10% Things and 2% HD1K, the latter yielding consistent improvements of the results. We used this setting to fine-tune our model for the Robust Vision Challenge. In the fourth entry, we investigated the impact of including less proportions of KITTI 2015 and introducing a percentage of RF-KITTI (Han et al. 2022) on the results. This setting leads to improvements only on the Middlebury results. Also using a few percent less of the other datasets and adding 8% of SlowFlow (Janai et al. 2017) did not improve the results.

[7] Our KITTI 2015 validation split consists of the following 50 samples from the KITTI 2015 (training) dataset: Sequences 0–29, Sequences 60–69 and Sequences 110–119.

[8] We used 100 bins for computing the WAUC measure. Please note that, in this case, higher values indicate better accuracy.

Table 14 Delineation of the impact of our fine-tuning setup (dataset mix, robust loss) and our multi-scale strategy

	Data mix	Robust loss	KITTI (split)	VIPER (validation)	Middlebury (train)
RAFT	orig	–	<u>6.51</u>	66.87	**0.281**
RAFT	ours	–	6.55	<u>78.56</u>	<u>0.285</u>
RAFT	ours	✓	**5.91**	**78.92**	0.296
RAFT	ours	✓	5.91	78.92	0.296
MS-RAFT	ours	✓	<u>5.07</u>	<u>81.25</u>	<u>0.201</u>
MS-RAFT+	ours	✓	**4.87**	**81.75**	**0.142**

Fine-Tuning Loss

In the following, we compare different loss functions for fine-tuning our model. In RAFT, the L1 loss was used as supervision during both pre-training and fine-tuning (cf. Sect. 3.2). Table 13 shows a comparison between the L1,[9] the L2 and the robust sample-wise L2 loss with different q values. For all experiments in the table we applied our mixed set from Table 12. Comparing the top entries, we can observe the advantage of the L2 loss for the VIPER and KITTI 2015 datasets. Regarding the robust sample-wise L2 loss, using the value $q = 0.85$ yields best results for the KITTI 2015 validation split with a small margin. In contrast, considering $q = 0.7$, gives the best results for Middlebury and VIPER datasets. We therefore used the robust sample-wise L2 loss with $q = 0.7$ to fine-tune our MS-RAFT+ model.

Discussion on Fine-Tuning

As experiments in Tables 12 and 13 show, both our mixed-training set and our robust sample-wise loss contribute in performing well across datasets. In Table 14 we show a comparison of RAFT fine-tuned with different settings, including the tuning on MPI Sintel as discussed in (Teed and Deng 2020) and fine-tuning using our mixed dataset and our robust sample-wise loss. Based on the experiments, except for Middlebury which is not used for fine-tuning, using our settings yields clear improvements. While the use of our mixed dataset enhances the results for VIPER, exploiting our robust sample-wise loss yields significant improvements for KITTI. Note that the former is not surprising, because VIPER was not included in the original mixed fine-tuning datasets of RAFT. Finally, we report the validation results of our MS-RAFT(+) models. Comparing the validation results of our multi-scale models to the single-scale RAFT approach shows the significant improvements for all of the validation sets.

6 Limitations

The proposed approach has also some limitations. On the one hand, the training and inference time of MS-RAFT+ is comparably large due to the on-demand computation of matching costs required to reduce the memory footprint at high resolutions and due to operating on high resolutions themselves. For instance, computing the optical flow for Sintel-sized images with 436 × 1024 pixels using the mentioned recurrent iterations-per-scale takes 0.53 s using an Nvidia A100 GPU. On the other hand, our model does not perform any form of occlusion handling (see e.g., Zhao et al. 2020; Jiang et al. 2021). This, however, is not a general problem, since such concepts are orthogonal to the multi-scale ideas of our approach and hence can be integrated to further improve its performance.

[9] For this experiment we used the L1 loss for both pre-training and fine-tuning the model, as in RAFT.

7 Conclusion

In this paper we have shown that multi-scale concepts are still useful when it comes to improving both the accuracy and the robustness of recent recurrent optical flow approaches. To this end, we have presented MS-RAFT+, a RAFT-based high-resolution multi-scale recurrent neural network that combines four hierarchical concepts in a single estimation framework. It complements (i) RAFT's original correlation pyramid that considers non-local cost information in the matching process by (ii) a coarse-to-fine estimation strategy that exploits valuable initializations from coarser resolutions, (iii) a multi-scale feature extractor that introduces higher-level information from coarser scales, as well as (iv) a multi-scale multi-iterations loss with sample-wise robustification that allows to downweight particularly difficult samples during training. In this way, we obtained a method that not only achieves state-of-the-art results on several benchmarks, but also allows to obtain these results with a single parameter setting, clearly demonstrating its robustness. In this context, also the benefits of the high resolution estimation became apparent. In contrast to other approaches our method shows much more structural details and performs favourably in matched regions and for small displacements, while still being sufficiently robust due to its multi-scale nature.

Acknowledgements The authors thank the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Project-ID 251654672—TRR 161 (B04) for funding and the state of Baden-Württemberg for their support through bwHPC. Furthermore, the authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Azin Jahedi.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability Our trained model and code are available at https://github.com/cv-stuttgart/MS_RAFT_plus. The training and evaluation data that support the findings of this study are available from Flying Chairs (Dosovitskiy et al. 2015), https://lmb.informatik.uni-freiburg.de/resources/datasets/FlyingChairs.en.html, Flying Things3D (Mayer et al. 2016), https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html/ MPI Sintel (Butler et al. 2012): http://sintel.is.tue.mpg.de/, KITTI 2015 (Menze and Geiger 2015), https://www.cvlibs.net/datasets/kitti/, RF-KITTI (Han et al. 2022), https://github.com/megvii-research/RealFlow, SlowFlow (Janai et al. 2017), https://www.cvlibs.net/projects/slow_flow/, HD1K (Kondermann et al. 2016), http://hci-benchmark.iwr.uni-heidelberg.de/ Middlebury (Baker et al. 2011), https://vision.middlebury.edu/flow/, VIPER (Richter et al. 2017), https://playing-for-benchmarks.org/.

References

Ali, A., Jalil, A., Niu, J., Zhao, X., Rathore, S., Ahmed, J., & Aksam Iftikhar, M. (2016). Visual object tracking–Classical and contemporary approaches. *Frontiers of Computer Science, 10*(1), 167–188.

Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision (IJCV), 2*(3), 283–310.

Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., & Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV), 92*(1), 1–31.

Barnes, C., Shechtman, E., Finkelstein, A., & Goldman, D. B. (2009). PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG), 28*(3), 24:1-24:11.

Black, M. J., & Anandan, P. (1991). Robust dynamic motion estimation over time. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 296–302).

Brox, T., Bregler, C., & Malik, J. (2009). Large displacement optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (*CVPR*) (pp. 41–48).

Brox, T., Bruhn, A., Papenberg, N., & Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *Proceedomg of the European conference on computer vision* (*ECCV*) (pp. 25–36).

Butler, D. J., Wulff, J., Stanley, G. B., & Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Proceeding of the European conference on computer vision* (*ECCV*) (pp. 611–625).

Chao, H., Gu, Y., & Napolitano, M. (2013). A survey of optical flow techniques for UAV navigation applications. In *Proceeding of the international conference on unmanned aircraft systems* (*ICUAS*) (pp. 710–716).

Chen, Z., Jin, H., Lin, Z., Cohen, S., & Wu, Y. (2013). Large displacement optical flow from nearest neighbor fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*).

Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., & Brox, T. (2015). FlowNet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision* (*ICCV*) (pp. 2758–2766).

Enkelmann, W. (1988). Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. *Computer Vision, Graphics, and Image Processing (CVGIP), 43*(2), 150–177.

Giachetti, A., Campani, M., & Torre, V. (1998). The use of optical flow for road navigation. *IEEE Transactions on Robotics and Automation (TRA), 14*(1), 34–48.

Han, Y., Luo, K., Luo, A., Liu, J., Fan, H., Luo, G., & Liu, S. (2022). Realflow: EM-based realistic optical flow dataset generation from videos. In *Proceedings of the European conference on computer vision* (*ECCV*) (pp. 288–305).

Hofinger, M., Bulò, S. R., Porzi, L., Knapitsch, A., Pock, T., & Kontschieder, P. (2020). Improving optical flow on a pyramid level. In *Proceedings of the European conference on computer vision* (*ECCV*) (pp. 770–786).

Hu, Y., Song, R., & Li, Y. (2016). Efficient coarse-tofine PatchMatch for large displacement optical flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5704–5712).

Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K. C., Qin, H., Dai, J., & Li, H. (2022). FlowFormer: A transformer architecture for optical flow. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 668–685).

Huber, P. (2004). *Robust statistics*. New York: Wiley.

Hui, T.-W., & Loy, C. C. (2020). Lite- FlowNet3: Resolving correspondence ambiguity for more accurate optical flow estimation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 169–184).

Hui, T.-W., Tang, X., & Loy, C. C. (2018). Lite- FlowNet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 8981–8989).

Hui, T.-W., Tang, X., & Loy, C. C. (2021). A lightweight optical flow CNN-revisiting data fidelity and regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 43*(8), 2555–2569.

Hur, J., & Roth, S. (2019). Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 5747–5756).

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 1647–1655).

Im, W., Lee, S., & Yoon, S. (2022). Semi-supervised learning of optical flow by flow supervisor. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 302–318). Springer.

Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Brock, A., Shelhamer, E., H'enaff, O., Botvinick,

M. M., Zisserman, A., Vinyals, O., & Carreira, J. (2022). Perceiver IO: A general architecture for structured inputs & outputs. In *Proceedings of the international conference on learning representations (ICLR)*.

Jahedi, A., Mehl, L., Rivinius, M., & Bruhn, A. (2022). Multi-Scale RAFT: Combining hierarchical concepts for learning-based optical flow estimation. *IEEE international conference on image processing (ICIP)* (pp. 1236–1240).

Janai, J., Güney, F., Behl, A., & Geiger, A. (2020). *Computer vision for autonomous vehicles: Problems, datasets and state of the art* (Vol. 12).

Janai, J., Güney, F., Wulff, J., Black, M. J., & Geiger, A. (2017). Slow flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1406–1416).

Jeong, J., Lin, J. M., Porikli, F., & Kwak, N. (2022). Imposing consistency for optical flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 3171–3181).

Jiang, S., Campbell, D., Lu, Y., Li, H., & Hartley, R. (2021). Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 9772– 9781).

Kajo, I., Malik, A. S., & Kamel, N. (2015). Motion estimation of crowd flow using optical flow techniques: A review. In *Proceedings of the international conference on signal processing and communication systems (ICSPCS)* (pp. 1–9).

Kim, T., Lee, H., & Lee, K. (2013). Optical flow via locally adaptive fusion of complementary data costs. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 3344–3351).

Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., & Jahne, B. (2016). The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (CVPR-W)* (pp. 19–28).

Lai, W.-S., Huang, J.-B., Wang, O., Shechtman, E., Yumer, E., & Yang, M.-H. (2018). Learning blind video temporal consistency. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 179–195).

Li, J., Wang, P., Xiong, P., Cai, T., Yan, Z., Yang, L., Liu, J., Fan, H., & Liu, S. (2022). Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 16263–16272).

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 936–944).

Liu, H., Lu, T., Xu, Y., Liu, J., Li, W., & Chen, L. (2022). CamLiFlow: Bidirectional camera-LiDAR fusion for joint optical flow and scene flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 5791–5801).

Long, L., & Lang, J. (2022). Detail preserving residual feature pyramid modules for optical flow. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV)* (pp. 2100–2108).

Lu, Y., Valmadre, J., Wang, H., Kannala, J., Harandi, M., & Torr, P. H. S. (2020). Devon: Deformable volume network for learning optical flow. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision (WACV)* (pp. 2694–2702).

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 674–679).

Luo, A., Yang, F., Li, X., & Liu, S. (2022). Learning optical flow with kernel patch attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 8896–8905).

Luo, A., Yang, F., Luo, K., Li, X., Fan, H., & Liu, S. (2022). Learning optical flow with adaptive graph reasoning. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2021). Multiple object tracking: A literature review. *Artificial Intelligence (AI), 293*, 103448.

Mahfouf, Z., Merouani, H. F., Bouchrika, I., & Harrati, N. (2018). Investigating the use of motion-based features from optical flow for gait recognition. *Neurocomputing (NC), 283*, 140–149.

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 4040-4048).

Menze, M., & Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 3061–3070).

Philipp, M., Bacher, N., Saur, S., Mathis-Ullrich, F., & Bruhn, A. (2022). From chairs to brains: Customizing optical flow for surgical activity localization. In *IEEE international symposium on biomedical imaging (ISBI): Proceedings*

Ranjan, A., & Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 2720–2729).

Ranjan, A., Hoffmann, D. T., Tzionas, D., Tang, S., Romero, J., & Black, M. J. (2020). Learning multi-human optical flow. *International Journal of Computer Vision (IJCV), 128*(4), 873–890.

Richter, S. R., Hayder, Z., & Koltun, V. (2017). Playing for benchmarks. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 2232–2241).

Rishav, Schuster, R., Battrawy, R., Wasenmüller, O., & Stricker, D. (2021). ResFPN: Residual skip connections in multi-resolution feature pyramid networks for accurate dense pixel matching. In *Proceedings of the international conference on pattern recognition (ICPR)* (pp. 180–187).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the international conference on medical image computing and computer-assisted intervention (MICCAI)* (pp. 234–241).

Saxena, R., Schuster, R., Wasenmuller, O., & Stricker, D. (2019). PWOC-3D: Deep occlusion-aware end-to-end scene flow estimation. In *Proceedings of the IEEE intelligent vehicles symposium (IV)* (pp. 324–331).

Sevilla-Lara, L., Liao, Y., Güney, F., Jampani, V., Geiger, A., & Black, M. J. (2019). On the integration of optical flow and action recognition. In *Proceedings of the german conference on pattern recognition (GCPR)* (pp. 281–297).

Shrivastava, A., Sukthankar, R., Malik, J., & Gupta, A. (2017). Beyond skip connections: Topdown modulation for object detection arXiv:1612.06851

Singh, J. P., Jain, S., Arora, S., & Singh, U. P. (2021). A survey of behavioral biometric gait recognition: Current success and future perspectives. *Archives of Computational Methods in Engineering (ACME), 28*(1), 107–148.

Sreenu, G., & Saleem Durai, M. A. (2019). Intelligent video surveillance: A review through deep learning techniques for crowd analysis. *Journal of Big Data, 6*(1), 48.

Sui, X., Li, S., Geng, X., Wu, Y., Xu, X., Liu, Y., Goh, R. & Zhu, H. (2022). CRAFT: Crossattentional flow transformer for robust optical flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 17581–17590).

Sun, D., Herrmann, C., Reda, F. A., Rubinstein, M., Fleet, D. J., & Freeman, W. T. (2022). Disentangling architecture and training for optical flow. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 165–182).

Sun, D., Vlasic, D., Herrmann, C., Jampani, V., Krainin, M., Chang, H., Zabih, R., Freeman, W. T. & Liu, C. (2021). AutoFlow: learning a better training set for optical flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 10093–10102).

Sun, D., Yang, X., Liu, M.-Y., & Kautz, J. (2018). PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 8934–8943).

Sun, D., Yang, X., Liu, M.-Y., & Kautz, J. (2020). Models matter, so does training: An empirical study of CNNs for optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 42*(6), 1408–1423.

Sun, S., Chen, Y., Zhu, Y., Gou, G., & Li, G. (2022). Learning optical flow with super kernels. In *Proceedings of the conference on neural information processing systems (NeurIPS)*.

Teed, Z., & Deng, J. (2020). RAFT: Recurrent allpairs field transforms for optical flow. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 402–419).

Tu, Z., Li, H., Xie, W., Liu, Y., Zhang, S., Li, B., & Yuan, J. (2022). Optical flow for video super-resolution: A survey. *Artificial Intelligence Review, 55*(8), 6505–6546.

Tu, Z., Poppe, R., & Veltkamp, R. C. (2016). Weighted local intensity fusion method for variational optical flow estimation. *Pattern Recognition (PR), 50*, 223–232.

Volz, S., Bruhn, A., Valgaerts, L., & Zimmer, H. (2011). Modeling temporal coherence for optical flow. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 1116–1123).

Wan, Z., Mao, Y., & Dai, Y. (2020). PRAFlow RVC: Pyramid recurrent all-pairs field transforms for optical flow estimation in Robust Vision Challenge 2020. arXiv:2009.06360 [cs].

Wang, H., Fan, R., & Liu, M. (2020). CoTAMFlow: Adaptive modulation network with co-teaching strategy for unsupervised optical flow estimation. In *Proceedings of the conference on robot learning (CoRL)* (pp. 143–155).

Xu, H., Yang, J., Cai, J., Zhang, J., & Tong, X. (2021). High-resolution optical flow from 1D attention and correlation. In *Proceedings of the IEEE/CVF international conference on computer vision* (*ICCV*) (pp. 10498– 10507).

Xu, H., Zhang, J., Cai, J., Rezatofighi, H., & Tao, D. (2022). GMFlow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 8111–8120).

Xu, L., Jia, J., & Matsushita, Y. (2011). Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 34*(9), 1744–1757.

Yang, G., & Ramanan, D. (2019). Volumetric correspondence networks for optical flow. In *Proceedings of conference on neural information processing systems (NeuRIPS)*.

Yao, G., Lei, T., & Zhong, J. (2019). A review of convolutional-neural-network-based action recognition. *Pattern Recognition Letters (PRL), 118*, 14–22.

Yin, Z., Darrell, T., & Yu, F. (2019). Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 6037–6046).

Yousif, K., Bab-Hadiashar, A., & Hoseinnezhad, R. (2015). An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intelligent Industrial Systems, 1*(4), 289–311.

Zendel, O., Dai, A., Fernandez, X. P., Geiger, A., Koltun, V., Kontschieder, P., ... Wulff, J. (2022). *ECCV 2022 robust vision challenge*. (http://www.robustvision.net/)

Zhang, F., Woodford, O. J., Prisacariu, V. A., & Torr, P. H. S. (2021). Separable Flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 10807–10817).

Zhang, F., Xu, S., & Zhang, X. (2020). High accuracy correspondence field estimation via MST based patch matching. *Multimedia Tools and Applications, 79*, 13291–13309.

Zhao, S., Sheng, Y., Dong, Y., Chang, E., & Xu, Y. (2020). Mask-Flownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 6278–6287).

Zhao, S., Zhao, L., Zhang, Z., Zhou, E., & Metaxas, D. N. (2022). Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 17571–17580).

Zheng, Z., Nie, N., Ling, Z., Xiong, P., Liu, J., Wang, H., & Li, J. (2022). DIP: Deep inverse patchmatch for high-resolution optical flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (*CVPR*) (pp. 8925–8934).