Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation

Junhwa Hur   Stefan Roth
Department of Computer Science, TU Darmstadt

Abstract

Deep learning approaches to optical flow estimation have seen rapid progress over the recent years. One common trait of many networks is that they refine an initial flow estimate either through multiple stages or across the levels of a coarse-to-fine representation. While leading to more accurate results, the downside of this is an increased number of parameters. Taking inspiration from both classical energy minimization approaches as well as residual networks, we propose an iterative residual refinement *(IRR) scheme based on* weight sharing *that can be combined with several backbone networks. It reduces the number of parameters, improves the accuracy, or even achieves both. Moreover, we show that integrating occlusion prediction and bi-directional flow estimation into our IRR scheme can further boost the accuracy. Our full network achieves state-of-the-art results for both optical flow and occlusion estimation across several standard datasets.*

1. Introduction

Akin to many areas of computer vision, deep learning has had a significant impact on optical flow estimation. But in contrast to, *e.g.*, object detection [19] or human pose estimation [55], the accuracy of deep learning-based flow methods on public benchmarks [10, 17, 41] had initially not surpassed that of classical approaches. Still, the efficient test-time inference has led to their widespread adoption as a sub-module in applications requiring to process temporal information, including video object segmentation [13], video recognition [15, 43, 64], and video style transfer [11].

FlowNet [14] pioneered the use of convolutional neural networks (CNNs) for estimating optical flow and relied on a – by now standard – encoder-decoder architecture with skip connections, similar to semantic segmentation [36], among others. Since the flow accuracy remained behind that of classical methods based on energy minimization, later work has focused on designing more powerful CNN architectures for optical flow. FlowNet2 [26] remedied the accuracy limitations of FlowNet and started to outperform classical approaches. Its main principle is to stack multiple FlowNet-
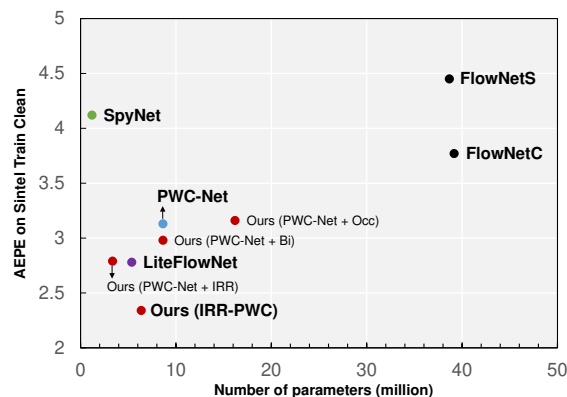

Figure 1. **Accuracy / network size tradeoff of CNNs for optical flow:** Combining our iterative residual refinement (IRR), as well as bi-directional (Bi) and occlusion estimation (Occ) with PWC-Net [52] in comparison to previous work. Our full model (IRR-PWC), combining all three components, yields significant accuracy gains over [52] while having many fewer parameters.

family networks [14], such that later stages effectively refine the output from the previous ones. However, one of the side effects of this stacking is the linearly and strongly increasing number of parameters, being a burden for the adoption in other applications. Also, stacked networks require training the stages sequentially rather than jointly, resulting in a complex training procedure in practice.

More recently, SpyNet [45], PWC-Net [52], and Lite-FlowNet [24] proposed lightweight networks that still achieve competitive accuracy (*cf*. Fig. 1). SpyNet adopts coarse-to-fine estimation in the network design, a well-known principle in classical approaches. It residually updates the flow across the levels of a spatial pyramid with individual trainable weights and demonstrates better accuracy than FlowNet but with far fewer model parameters. Lite-FlowNet and PWC-Net further combine the coarse-to-fine strategy with multiple ideas from both classical methods and recent deep learning approaches. Particularly PWC-Net outperformed all published methods on the common public benchmarks [10, 17, 41].

Interestingly, many recent deep learning approaches for flow [24, 26, 45, 52] have a common structure: From a rough first flow estimate, later modules or networks re-

(a) FlowNet2-like [26] stack of FlowNet networks.

(b) Iterative residual refinement version of *(a)*.

(c) A rolled version of *(b)*.

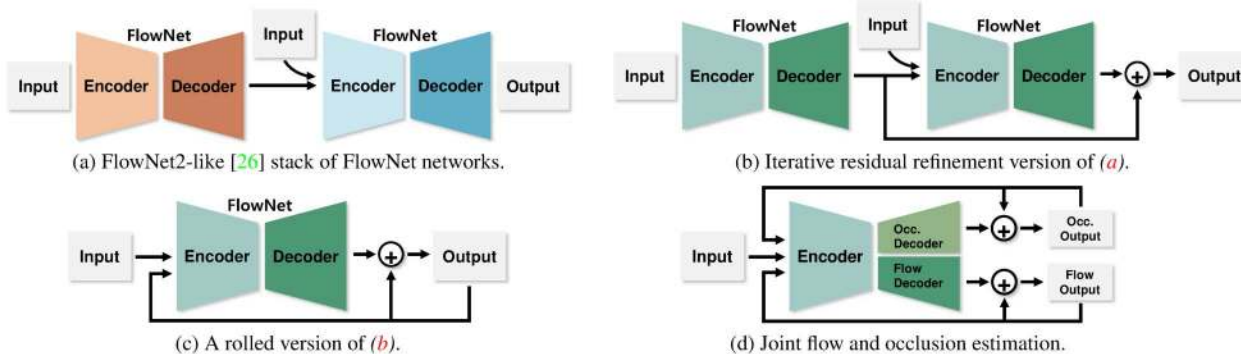(d) Joint flow and occlusion estimation.

Figure 2. **From a standard network stack to our iterative residual refinement scheme with joint optical flow and occlusion estimation:** The stacked version of FlowNet *(a)* can be converted into an iterative residual refinement model *(b)* with the half number of parameters. Note that modules with the same color share their weights. We can re-interpret *(b)* as a rolled version *(c)*, making it more immediate to include occlusion estimation *(d)* for further improving the accuracy.

fine the previous estimates across pyramid levels or through multiple chained networks. As illustrated in Fig. 2a, the later modules or networks have their own trainable weights, since each module assumes a particular functionality at the respective spatial resolution or conditioned on the output of the preceding modules. The downside is that this significantly increases the number of required model parameters.

In this paper, we take the inspiration from classical energy minimization-based optical flow approaches several steps further. Energy-based methods iteratively estimate the flow based on a consistent underlying energy with a single set of parameters [8, 9, 51]. Hence we ask: *Can we iteratively refine flow with a deep network based on a single, shared set of weights?* Moreover, energy-based methods have benefited from bi-directional estimation and occlusion reasoning [2, 25, 50, 60]. We thus ask: *Can deep learning approaches to optical flow similarly benefit from bi-directional estimation with occlusion reasoning?*

We address these questions and make a number of contributions: *(i)* We first propose an iterative residual refinement (IRR) scheme that takes the output from a previous iteration as input and iteratively refines it by only using a *single* network block with *shared weights*. *(ii)* We demonstrate the applicability to two popular networks, FlowNet [14] (Fig. 2c) and PWC-Net [52] (Fig. 4). For FlowNet, we can significantly increase the accuracy without adding parameters; for PWC-Net, we can reduce the number of parameters while even improving the accuracy (Fig. 1). *(iii)* Next, we demonstrate the integration with occlusion estimation (Fig. 2d). *(iv)* We further extend the scheme to bi-directional flow estimation, which turns out to be only beneficial when combined with occlusion estimation. Unlike previous work [27], our scheme enables the flow accuracy to benefit from joint occlusion estimation. *(v)* We finally propose lightweight bilateral filtering and occlusion upsampling layers for refined motion and occlusion boundaries.

Applying our proposed scheme to two backbone net-

works, FlowNet and PWC-Net, yields significant improvements in flow accuracy of 18.5% and 17.7%, respectively, across multiple datasets. In case of PWC-Net, we achieve this accuracy gain using 26.4% *fewer* parameters. Note that occlusion estimation and bi-directional flow are additional outcomes as by-products of this improvement.

2. Related Work

Optical flow with CNNs. Starting with FlowNet [14], various deep network architectures for optical flow have been proposed, *e.g.*, FlowNet2 [26], SpyNet [45], PWC-Net [52], and LiteFlowNet [24]. They are based on an autoencoder design, allow for supervised end-to-end training, and enable fast inference during testing time. To alleviate the need for training data with ground truth in a specific domain, unsupervised [1, 40, 47, 58, 63, 66] and semi-supervised [31, 65] alternatives have also been developed.

Other than such end-to-end approaches, CNNs can also serve to extract learned feature descriptors, which are combined with classical optimizers or well-designed correspondence search methods to find matches between extracted features [4, 6, 16, 20, 59, 61]. Such optimization-based approaches can yield less blurry results than typical CNN decoders, but not all are end-to-end trainable and their runtime in the testing phase is significantly longer.

Here, we investigate how to improve generic autoencoder-based architectures by adapting an iterative residual scheme that is widely applicable.

Optical flow and occlusion. Occlusion has been regarded as an important cue for estimating more accurate optical flow. Because occluded pixels do not have correspondences in the other frame, several approaches [5, 12, 16, 23, 32] aim to filter out these outliers to minimize their ill effects and apply post-processing to refine the estimates [22, 33, 48, 67].

Other methods [2, 7, 25, 28, 50, 57, 60] model occlusions in a joint energy and utilize them as additional evi-
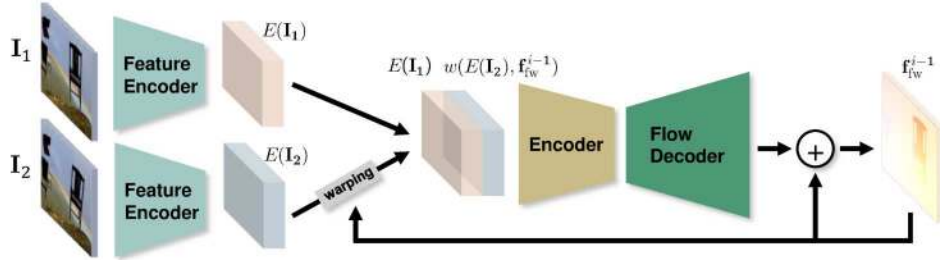
Figure 3. **Our Iterative Residual Refinement (IRR) version of FlowNetS** [14]. The model iteratively estimates residual flow from the previous output. Note that we apply warping after several encoder layers, see text for details.

dence for flow through recursive joint estimation. Occlusion estimates can enable *(i)* a more accurate matching cost [50, 57, 60], *(ii)* bi-directional consistency [25, 28], or *(iii)* uniqueness constraints for pixel-level matching [2, 25, 54].

Recently, [58] proposed an unsupervised deep network that jointly estimates flow and occlusion. Occlusions are explicitly detected from the inverse of disocclusion [25]; the per-pixel loss is disabled on occluded pixels. [27] proposed a supervised network for jointly estimating optical flow and occlusion, as well as depth and motion boundaries. [30, 42] integrate occlusion estimation into a PWC-Net backbone based on temporal propagation in longer sequences; [30] is based on unsupervised learning. Our work also directly learns to estimate occlusion using ground-truth supervision signals, but requires only two frames and unlike [27] enables to improve the flow using the estimated occlusions.

Iterative and residual refinement. Despite of the immense learning capacity of deep networks, early CNN approaches to optical flow did not outperform classical methods [14]. Often, the CNN decoder yielded blurry, thus less accurate regression results. In order to overcome this and motivated by classical coarse-to-fine refinement [8, 9, 51], SpyNet [45] and PWC-Net [52] residually update the flow along a pyramid structure. FlowNet2 [26] instead stacks multiple networks to refine the previous estimates, though this linearly increases the network size. DeMoN [56] uses a combined strategy of stacking and iteratively using one network, also requiring more parameters than one single baseline network. LiteFlowNet [24] cascades extra convolution layers for refining the outputs and regularizes the outputs based on a feature-driven local convolution, which adaptively defines the convolution weights based on the estimated outputs. Related approaches have also been proposed in the stereo literature [18, 34, 44], successfully improving the accuracy but still increasing the network size.

In contrast, we propose a generic scheme that repetitively uses one baseline network to yield better accuracy without increasing the network size. For certain networks, we even reduce the size by removing repetitive modules while still enabling competitive or even improved accuracy.

3. Approach

3.1. Core concepts & base networks

**Iterative residual refinement (IRR) with shared weights.** The basic problem setup is to estimate (forward) optical flow $\mathbf{f}_{\text{fw}}$ from the reference frame $\mathbf{I}_1$ to the target frame $\mathbf{I}_2$. The main concept of our IRR scheme is to make a model learn to residually refine its previous estimate by iteratively *re-using* the same network block with *shared weights*. We pursue two scenarios: *(i)* We increase the accuracy without adding parameters or complicating the training procedure, by iteratively re-using a single network to keep refining its previous estimate; or *(ii)* we aim toward a more compact model by substituting multiple network blocks assuming the same basic functionality with only a single block.

IRR with FlowNet. Addressing the first scenario, we propose an iterative residual refinement version of FlowNetS [14], *cf.* Fig. 3 for an overview. Our IRR version iteratively estimates residual flow with multiple iterations using one single FlowNetS; the final result is the sum of residual flows from all iteration steps. We use one shared encoder E for feature extraction from each input image $\mathbf{I}_1$ and $\mathbf{I}_2$, similar to FlowNetC, and concatenate the two feature maps after warping the second feature map based on the estimated flow $\mathbf{f}_{\text{fw}}^{i-1}$ from the previous iteration $i-1$. Then we input the concatenated feature maps to the decoder D to estimate the residual (forward) flow at iteration i:

$$\mathbf{f}_{\text{fw}}^i = D\Big(E(\mathbf{I}_1), w\big(E(\mathbf{I}_2), \mathbf{f}_{\text{fw}}^{i-1}\big)\Big) + \mathbf{f}_{\text{fw}}^{i-1}, \qquad (1)$$

where $w(\cdot, \cdot)$ is a bilinear interpolation function for backward warping [29]. Here, warping the second feature map is crucial as it yields a suitable input for estimating the appropriate *residual* flow. This yields much improved accuracy while re-using the same network with only slight modifications and not requiring additional training stages.

IRR with PWC-Net. Based on the classical coarse-to-fine principle, PWC-Net [52] and SpyNet [45] both use multiple repetitive modules for the same purpose but with *separate weights*. Fig. 4a shows a 3-level PWC-Net (for ease of visualization, originally 7-level) that incrementally updates
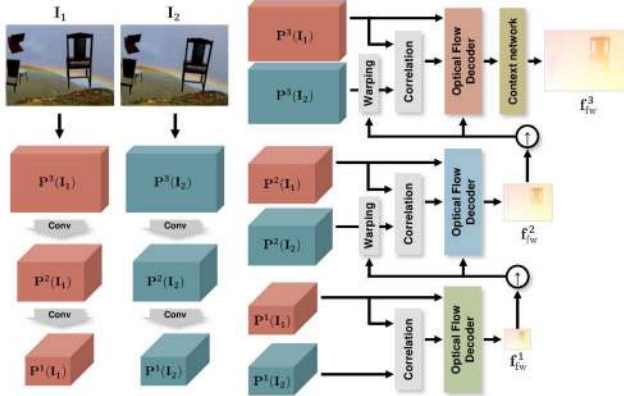
(a) Original PWC-Net [52].
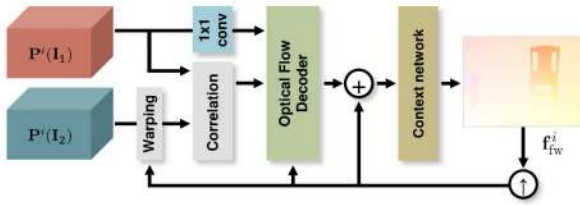
(b) Our IRR version of PWC-Net.

Figure 4. **Our IRR version of PWC-Net**, which uses only one single shared decoder over the pyramid levels, see text for details.

the estimation across the pyramid levels with individual decoders for each level. Adopting our IRR scheme here to address the second scenario, we can substitute the multiple decoders with only one shared decoder that iteratively refines the output over all the pyramid levels, cf. Fig. 4b. We set the number of iterations equal to the number of pyramid levels, keeping the original pipeline but with fewer parameters and a more compact representation:

$$\mathbf{f}_{\text{fw}}^i = D\Big(\mathbf{P}^i(\mathbf{I}_1), c\big(\mathbf{P}^i(\mathbf{I}_1), w(\mathbf{P}^i(\mathbf{I}_2), \hat{\mathbf{f}}_{\text{fw}}^{i-1})\big), \hat{\mathbf{f}}_{\text{fw}}^{i-1}\Big) + \hat{\mathbf{f}}_{\text{fw}}^{i-1} \tag{2a}$$

with

$$\hat{\mathbf{f}}_{\text{fw}}^{i-1} = 2 \cdot \uparrow(\mathbf{f}_{\text{fw}}^{i-1}), \tag{2b}$$

where $\mathbf{P}^i$ is the feature map at pyramid level i, $c(\cdot, \cdot)$ calculates a cost volume, and $\uparrow$ performs $2\times$ bilinear upsampling to twice the resolution of the previous flow field. As the dimension increases, we also scale the flow magnitude accordingly (Eq. 2b).

One important change from the original PWC-Net [52], which estimates flow for each level on the original scale, is that we estimate flow for each level at its native spatial resolution. This enables us to use only one shared decoder and yet make it possible to handle different resolutions across all levels. When calculating the loss, we revert back to the original scale to use the same loss function.

In addition, we add a 1×1 convolution layer after the input feature map $\mathbf{P}^i(\mathbf{I}_1)$ to make the number of feature maps input to the decoder D be equal across the pyramid

levels. This enables us to use one single shared decoder with a fixed number of input channels across the pyramid.

Occlusion estimation. It is widely reported that jointly localizing occlusions and estimating optical flow can benefit each other [2, 7, 25, 28, 50, 57, 60]. Toward leveraging this in the setting of CNNs, we attach an additional decoder estimating occlusion $\mathbf{o}_1^i$ in the first frame at the end of the encoder, in parallel to the flow decoder as shown in Fig. 2d, similar to [30, 42]. The occlusion decoder has the same configuration as the flow decoder, but the number of output channels is 1 (instead of 2 for flow). The input to the occlusion decoder is the same as to the flow decoder.

3.2. Joint optical flow and occlusion estimation

Iteratively re-using residual subnetworks and adding occlusion decoders are independent, easily combined together, and adaptable to many types of optical flow base networks. Beyond simply adopting these two concepts, we additionally propose several ideas to improve the accuracy further in a joint estimation setup: *(i)* bi-directional estimation, *(ii)* bilateral refinement of flow and occlusion, and *(iii)* an occlusion upsampling layer.

Bi-directional estimation. Based on the basic IRR setup for joint flow and occlusion estimation in Fig. 2d, we first perform bi-directional flow and occlusion estimation by simply switching the order of the input feature maps for the decoder [40, 58]. This yields backward flow $\mathbf{f}_{\text{bw}}^i$ and occlusion $\mathbf{o}_2^i$ in the second frame. Note that bi-directional estimation requires no extra convolutional weights as it again re-uses the same shared decoders. As we shall see, estimating both forward and backward flow together yields at most minor accuracy improvements itself, but we find that exploiting forward-backward consistency is crucial for estimating more accurate occlusions.

Bilateral refinement of flow and occlusion. Blurry estimates, particularly near motion boundaries, have recently been identified as a main limitation of standard optical flow decoders in CNNs. To address this, bilateral filters or local attention maps [21, 24] have been proposed as viable solutions. We also adopt this idea in our setup, extend it to refine optical flow *and* occlusion using bilateral filters, but with *weight sharing* across all iteration steps.

Similar to Hui *et al.* [24], we construct learned bilateral filters individualized to each pixel and apply them to each flow component u, v and the occlusion separately:

$$\tilde{f}_{\text{fw},u}^i(x,y) = g_{\text{fw}}(x,y) * f_{\text{fw},u}^i(x,y) \tag{3a}$$

$$\tilde{f}_{\text{fw},v}^i(x,y) = g_{\text{fw}}(x,y) * f_{\text{fw},v}^i(x,y) \tag{3b}$$

$$\tilde{o}_1^i(x,y) = g_o(x,y) * o_1^i(x,y), \tag{3c}$$

where, *e.g.*, $\tilde{f}_{\text{fw},u}^i(x,y)$ is the filtered horizontal flow at (x,y), $g_{\text{fw}}(x,y)$ is the $w \times w$ learned bilateral filter kernel
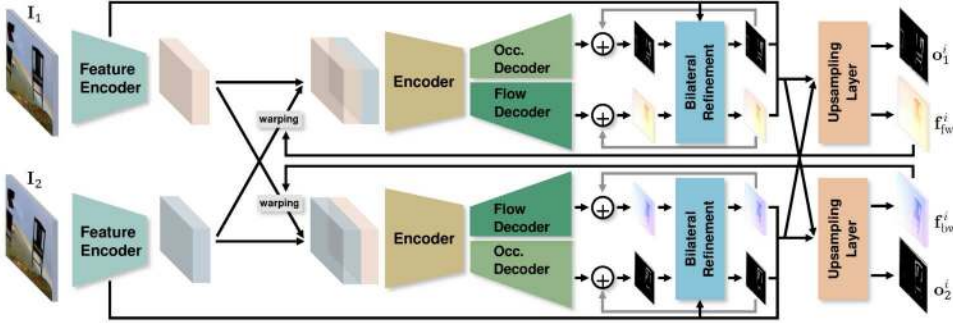
Figure 5. **Joint optical flow and occlusion estimation: bi-directional estimation, bilateral refinement, and upsampling layer** (in the FlowNet setting): We estimate flow in both temporal directions and occlusion maps in both frames by switching the order of inputs to the decoder. Bilateral refinement and the upsampling layer further improve the accuracy of flow and occlusion. Modules with the same color share their weights. *C.f.* supplemental material for the corresponding PWC-Net variant.

(a) Ground-truth occlusion map.

(b) Reconstructed occlusion map from a quarter resolution of (a).

Figure 6. **Oracle study** showing the limitation of outputting low-resolution occlusion maps.

for flow at (x, y), and $f_{\mathrm{fw},u}^i(x, y)$ is the $w \times w$ patch of the horizontal flow centered at (x, y). Note that we construct the kernels for flow and occlusion separately as motion and occlusion boundaries are not necessarily aligned.

For constructing the bilateral filter for the flow, we follow the strategy of Hui *et al.* [24], and for occlusion we input occlusion estimates, a feature map, and a warped feature map from the other temporal direction. One important difference to [24] is that we do not need separate learnable convolutional weights for every iteration step or every pyramid level. Our IRR design enables re-using the same weights for constructing the bilateral filters for all iteration steps or pyramid levels. In case of adapting to PWC-Net, our bilateral refinement adds only 0.69M parameters, which is $2.4\times$ less than the scheme of [24], adding 1.66M parameters.

Occlusion upsampling layer. One common trait of FlowNet [14] and PWC-Net [52] is that the output resolution of flow from the CNN is a quarter ($\frac{1}{4}H \times \frac{1}{4}W$) of the input resolution ($H \times W$), which is then bilinearly upscaled to the input resolution. The reasons for not directly estimating at full resolution are a marginal accuracy improvement and the GPU computation and memory overhead.

Yet for estimating occlusion, there is a significant accuracy loss when estimating only at a quarter resolution. On the Sintel dataset, we conduct an oracle study by downscaling the ground-truth occlusion maps to a quarter size and then upscaling them back. The F-score of the reconstructed occlusion maps was 0.777, suggesting a significant accuracy limitation. As seen in Fig. 6, quarter-resolution occlusion maps cannot really represent fine occlusions, through which the major loss in F-score occurs. This strongly emphasizes the importance of estimating at full resolution.

To estimate more accurate occlusion at higher resolution, we attach an upsampling layer at the end of network, *cf.* Fig. 5, to upscale optical flow and occlusion together back to the input resolution. Fig. 7 illustrates our upsampling
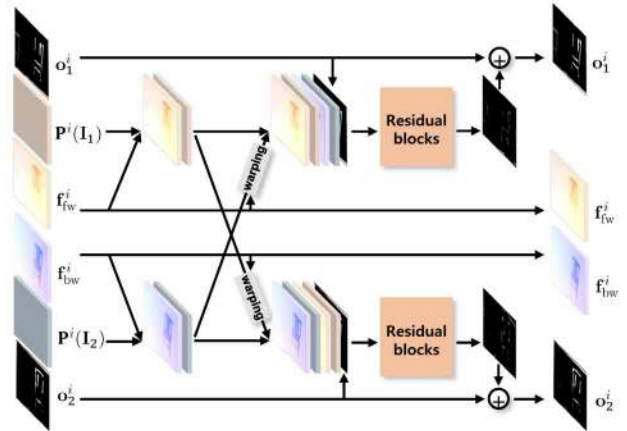

Figure 7. **Occlusion upsampling layer**: Inputs are bi-linearly upsampled flow and upsampled occlusion using nearest neighbor. Residual blocks then improve the occlusion accuracy using residual occlusion updates at the full output resolution.

layer. For optical flow, we found bilinear upsampling to be sufficient. For occlusion, we first perform nearest-neighbor upsampling, which is fed into a CNN module to estimate the residual occlusion on the upsampled occlusion map. The CNN module consists of three residual blocks [35], which receive flow, a feature map from the encoder, warped flow, and a warped feature map from the other temporal direction. Putting the warped feature map and flow from the other direction enables exploiting the classical forward-backward consistency for estimating the occlusion. We provide further details in the supplemental material.

Initialization. To bootstrap our iterative estimation, we input zero as initial optical flow (*i.e.* $\mathbf{f}_{\mathrm{fw}}^0$ and $\mathbf{f}_{\mathrm{bw}}^0$) and occlusion (*i.e.* $\mathbf{o}_1^0$ and $\mathbf{o}_2^0$) into the first stage. Note that 0 indicates non-occluded (visible) and 1 indicates occluded.

Training loss. Let N be the total number of steps in our iterative setting. Then we predict a set of forward optical

flow maps $\mathbf{f}_{\text{fw}}^i$, backward optical flow $\mathbf{f}_{\text{bw}}^i$, occlusion maps in the first image $\mathbf{o}_1^i$ and in the second image $\mathbf{o}_2^i$ for each iteration step, where $i = 1, \ldots, N$. Forward and backward optical flow are supervised using the $\mathcal{L}_{2,1}$ norm as

$$l_{\text{flow}}^i = \tfrac{1}{2} \sum \left(\|\mathbf{f}_{\text{fw}}^i - \mathbf{f}_{\text{fw,GT}}\|_2 + \|\mathbf{f}_{\text{bw}}^i - \mathbf{f}_{\text{bw,GT}}\|_2 \right), \quad (4)$$

whereas for the supervision of the two occlusion maps we use a weighted binary cross-entropy

$$l_{\text{occ}}^i = -\tfrac{1}{2} \sum \left(w_1^i o_1^i \log o_{1,\text{GT}} + \bar{w}_1^i (1 - o_1^i) \log (1 - o_{1,\text{GT}}) \right.$$
$$\left. + w_2^i o_2^i \log o_{2,\text{GT}} + \bar{w}_2^i (1 - o_2^i) \log (1 - o_{2,\text{GT}}) \right).$$
$$(5)$$

Here, we apply the weights $w_1^i = \frac{H \cdot W}{\sum o_1^i + \sum o_{1,\text{GT}}}$ and $\bar{w}_1^i = \frac{H \cdot W}{\sum (1 - o_1^i) + \sum (1 - o_{1,\text{GT}})}$ to take into account the number of predictions and true labels.

Our final loss is the weighted sum of the two losses above, taken over all iteration steps using the same multi-scale weights α_s as in the original papers. In case of FlowNet [14], the final loss becomes

$$l_{\text{FlowNet}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{s=s_0}^{S} \alpha_s (l_{\text{flow}}^{i,s} + \lambda \cdot l_{\text{occ}}^{i,s}), \qquad (6)$$

where s denotes the scale index given in Fig. 3 of [14]. In case of PWC-Net [52], the number of scales is equal to the number of iterations, hence the final loss is

$$l_{\text{PWC-Net}} = \frac{1}{N} \sum_{i=1}^{N} \alpha_i (l_{\text{flow}}^i + \lambda \cdot l_{\text{occ}}^i). \qquad (7)$$

λ weighs the flow against the occlusion loss. In every iteration, we calculate the λ that makes the loss of the flow and the occlusion be equal. We empirically found that this strategy yields better accuracy than just using a fixed trade-off.

4. Experiments

4.1. FlyingChairsOcc dataset

Lacking a suitable dataset, we create our own dataset for the supervision of bi-directional flow and the two occlusion maps, with ground truth for forward flow, backward flow, and occlusion maps at the first and second frame. To build the dataset, we follow the exact protocol of the FlyingChairs dataset [14]. We refer to this dataset as *FlyingChairsOcc*.

We crawl 964 background images with a resolution of 1024×768 from Flickr and Google using the keywords *cityscape*, *street*, and *mountain*. As foreground objects, we use 809 chair images rendered from CAD models with varying views and angles [3]. Then we follow the exact protocol of [14] for generating image pairs, including the

number of foreground objects, object size, and random parameters for generating the motion of each object. As the motion is parametrized by a 3×3 matrix, it is easy to calculate not only backward ground-truth flow but also occlusion maps by conducting visibility checks. The number of images in the training and validation sets are the same as in FlyingChairs (*i.e.* 22232 and 640, respectively).

4.2. Implementation details

Training details. We follow the training settings of FlowNet respective PWC-Net for a fair comparison. We use the same geometric and photometric augmentations with additive Gaussian noise as described in [26]. After applying the geometric augmentation on the occlusion ground truth, we additionally check for pixels moving outside of the image boundary (*i.e.* out-of-bound pixels) and set them as occluded. Note that no multi-stage training is needed.

We first train the proposed model on our FlyingChairs-Occ dataset with learning rate schedule S_{short} (instead of S_{long}), described in [26]. Next, we fine-tune on the Flying-Things3D-subset dataset [39], which contains much larger displacements; we use half the S_{fine} learning rate schedule [26]. We empirically found that using shorter schedules was enough as our model converged faster. We finally fine-tune on different public benchmark datasets, including Sintel [10] and KITTI [17], following the fine-tuning protocol of [53]. We use a smaller minibatch size of 4, as our model implicitly increases the batch size by performing iterative bi-directional estimation with a single model.

Lacking other ground truth, we only use the forward flow and the occlusion map for the first frame for supervision on Sintel; for KITTI we only use the forward flow. Importantly, our model is still trainable when ground truth is available only for one direction (*e.g.*, forward flow with occlusion map at the first frame), since both temporal directions share the same "unidirectional" decoder.

4.3. Ablation study

To see the effectiveness of each proposed component, we conduct an ablation study by training our model in multiple settings. All models are trained on the FlyingChair-sOcc dataset with the S_{short} schedule and tested on multiple datasets to assess generalization across datasets. We use a minibatch size of 4 when either bi-directional estimation or iterative residual refinement is on, or the original minibatch size of 8, otherwise. For a simpler ablation study, we use two iteration steps when applying IRR on FlowNet [14].

Table 1 assesses the optical flow in terms of the average end-point error (EPE) and occlusion estimation with the average F1-score, if applicable for the respective configuration. In contrast to findings in recent work [27], estimating occlusion together yields a gradual improvement of the flow of up to 5% on the training domain, and an even bigger im-

Figure 8. **Qualitative examples from the ablation study on PWC-Net**: *(left to right)* overlapped input images, ground-truth flow, the original PWC-Net [52], our PWC-Net with IRR, our PWC-Net with Bi-Occ-IRR, and our full model (*i.e.* IRR-PWC).

	Bi	Occ	IRR	Chairs Full	ChairsOcc Validation	Sintel Clean Training	Sintel Final Training	Rel. Param.
FlowNet [14]				2.39	2.27	4.35	5.44	0 %
	✓			2.43	2.30	4.40	5.53	0 %
		✓		2.29	2.18 (0.690)	4.26 (0.521)	5.51 (0.493)	+38.5%
			✓	2.36	2.22	3.77	5.00	0 %
	✓	✓		2.31	2.20 (0.691)	4.21 (0.515)	5.46 (0.488)	+38.5%
	✓		✓	2.14	2.00	3.45	4.96	0 %
		✓	✓	2.22	2.10 (0.689)	3.56 (0.507)	5.03 (0.486)	+38.5%
	✓	✓	✓	2.05	1.91 (0.699)	3.40 (0.528)	5.08 (0.502)	+38.5%
	✓	✓	✓+	**1.92**	**1.77 (0.736)**	**3.32 (0.596)**	**4.92 (0.560)**	+40.7%
PWC-Net [52]				2.03	1.89	3.13	4.41	0 %
	✓			2.06	1.87	2.98	4.14	0 %
		✓		1.94	1.79 (0.706)	3.16 (0.616)	4.35 (0.581)	+87.4%
			✓	2.01	1.83	2.79	4.10	−61.2%
	✓	✓		1.99	1.82 (0.696)	3.01 (0.618)	4.39 (0.581)	+87.4%
	✓		✓	2.08	1.90	2.80	4.13	−61.2%
		✓	✓	1.91	1.73 (0.700)	2.64 (0.630)	4.09 (0.593)	−34.7%
	✓	✓	✓	1.98	1.81 (0.698)	2.69 (0.633)	4.03 (0.598)	−34.7%
	✓	✓	✓+	**1.67**	**1.48 (0.757)**	**2.34 (0.677)**	**3.95 (0.624)**	−26.4%

Table 1. **Ablation study of our design choices on the two baseline models.** The numbers indicate the average end-point error (EPE) for optical flow (the lower the better) and the average F1-score for occlusion in parentheses, where available (the higher the better). **Bi**: Bi-directional estimation, **Occ**: Joint occlusion estimation, **IRR**: Iterative residual refinement, **IRR+**: Iterative residual refinement including bilateral refinement and occlusion upsampling layer. The final column reports the relative changes on the number of parameters comparing to the vanilla baseline.

Method	Chairs Full	ChairsOcc Validation	Sintel Clean Training	Sintel Final Training	Rel. Param.
No refinement	1.98	1.81 (0.698)	2.69 (0.633)	4.03 (0.598)	0 %
Ours	**1.66**	**1.45 (0.735)**	**2.32 (0.648)**	3.90 **(0.602)**	+12.3%
LiteFlowNet's [24]	1.74	1.58 (0.688)	2.34 (0.596)	**3.86** (0.543)	+29.5%

Table 2. Comparison of our bilateral refinement layer against that of LiteFlowNet [24].

Method	Chairs Full	ChairsOcc Validation	Sintel Clean Training	Sintel Final Training	Rel. Param.
No upsampling	**1.66**	**1.45 (0.735)**	**2.32 (0.648)**	**3.90** (0.602)	0 %
Ours	1.67	1.48 **(0.757)**	2.34 **(0.677)**	3.95 **(0.624)**	+0.49%
[26, 27]	2.18	2.01 (0.712)	2.90 (0.624)	4.37 (0.577)	+9.21%

Table 3. Comparison of our occlusion upsampling layer and the refinement network from FlowNet2 [26, 27].

provement across different datasets when combined on top of bi-directional estimation (Bi) or IRR. We believe this to mainly stem from using a separate occlusion decoder instead of a joint decoder [27]. Bi-directional estimation by itself yields at most a marginal improvement on flow, but it is important for the input of the occlusion upsampling layer, which brings very large benefits on occlusion estimation. Iterative residual refinement yields consistent improvements in flow accuracy on the training domain, and perhaps surprisingly *a much better generalization across datasets*, with up to 10% improvement in EPE. We presume that this better generalization comes from training a single decoder to handle feature maps from all iteration steps or pyramid levels, which encourages generalization even across datasets. The benefits of using IRR become even clearer when combined with other components. For example, FlowNet with Bi, Occ, and IRR demonstrates up to 20% improvement in EPE on Sintel Clean compared to only using Bi and Occ. Additionally, the bilateral refinement and the upsampling

layer significantly improve the accuracy of both flow and occlusion with a small overhead of only 0.83M parameters. For PWC-Net, we obtain a significant accuracy boost of 17.7% on average over the baseline, while reducing the number of parameters by 26.4%. We name the full versions of the models including all modules **IRR-FlowNet** and **IRR-PWC**. Fig. 8 highlights the improvement of the flow from our proposed components with qualitative examples. Please note the completeness and sharp boundaries.

Bilateral refinement. We compare our bilateral refinement layer with the refinement layer of LiteFlowNet [24] based on a PWC-Net with Bi, Occ, and IRR components enabled. Table 2 shows that the benefit of our design choice (*i.e.* sharing weights) holds for bilateral refinement as well, yielding better accuracy for flow and particularly for occlusion, with $2.5\times$ fewer parameters than that of [24].

Occlusion upsampling layer. Similar to our upsampling layer, [27] uses a refinement network from FlowNet2 [26] to upsample the intermediate quarter-resolution outcome back to the original resolution. We compare our upsampling layer with the refinement network from [26, 27], adding it to our network based on a PWC-Net backbone with Bi, Occ, IRR, and the bilateral refinement layer enabled. Table 3 shows the clear benefits of using our upsampling layer, yielding significant gains in both tasks while requiring fewer parameters. The refinement network from FlowNet2 [26] actually degrades the accuracy of flow estimation. We presume this may stem from differences in

Number of iterations or stacking stages	1	2	3	4	5
IRR on a single FlowNetS	**4.358**	**3.545**	**3.325**	**3.303**	**3.302**
Stacking multiple FlowNetS	4.445	3.553	3.377	3.391	3.517

Table 4. $n\times$ IRR *vs.* $n\times$ stacking: EPE on Sintel Clean.

Method	Training		Test		Parameters
	Clean	Final	Clean	Final	
ContinualFlow_ROB[†§] [42]	–	–	3.34	**4.53**	14.6 M
MFF[§] [46]	–	–	3.42	4.57	N/A
IRR-PWC (Ours)	(1.92)	(2.51)	3.84	4.58	6.36M
PWC-Net+[†] [53]	(1.71)	(2.34)	3.45	4.60	8.75M
ProFlow[§] [37]	–	–	2.82	5.02	–
PWC-Net-ft-final [53]	(2.02)	(2.08)	4.39	5.04	8.75M
DCFlow [61]	–	–	3.54	5.12	–
FlowFieldsCNN [6]	–	–	3.78	5.36	5.00M
MR-Flow [59]	1.83	3.59	**2.53**	5.38	–
LiteFlowNet [24]	(1.35)	(1.78)	4.54	5.38	5.37M
S2F-IF [62]	–	–	3.50	5.42	–
SfM-PM [38]	–	–	2.91	5.47	–
FlowFields++ [49]	–	–	2.94	5.49	–
FlowNet2 [26]	(2.02)	(3.14)	3.96	6.02	162.5 M

Table 5. **MPI Sintel Flow**: Average end-point error (EPE) and number of CNN parameters. [§]using more than 2 frames, [†]using additional datasets (KITTI and HD1k) for better accuracy.

training. FlowNet2's refinement layer may require piece-wise training, while our model is trained all at once.

Different IRR steps on FlowNet. For FlowNet, we can freely choose the number of IRR steps as we iteratively refine previous estimates by re-using a single network. We try different numbers of IRR steps on vanilla FlowNetS [14] (*i.e.* without Bi or Occ) and compare with stacking multiple FlowNetS networks. All networks are trained on FlyingChairsOcc with the S_{short} schedule, minibatch size of 8, and tested on Sintel Clean. As shown in Table 4, the accuracy keeps improving with more IRR steps and stably settles at more than 4 steps. In contrast, stacking multiple FlowNetS networks overfits on the training data after 3 steps, and is consistently outperformed by IRR with the same number of stages. This clearly demonstrates the advantage of our IRR scheme over stacking: *better accuracy without linearly increasing the number of parameters*.

4.4. Optical flow benchmarks

We test the accuracy of our IRR-PWC on the public Sintel [10] and KITTI [17, 41] benchmarks. When fine-tuning, we use the robust training loss as in [24, 52, 53] for flow, and standard binary cross-entropy for occlusion. On Sintel Final, our IRR-PWC achieves a new state of the art among 2-frame methods. Comparing to the PWC-Net baseline (*i.e.* PWC-Net-ft-final) trained in the identical setting, our contributions improve the flow accuracy by 9.18% on Final and 12.36% on Clean, while using 26.4% fewer parameters. On KITTI 2015, our IRR-PWC again outperforms all published 2-frame methods, improving over the baseline PWC-Net.

When fine-tuning on benchmarks, our important obser-

Method	Training		Test
	AEPE	Fl-all	Fl-All
MFF[§] [46]	–	–	**7.17%**
IRR-PWC (Ours)	(1.63)	(5.32%)	7.65%
PWC-Net+ [53]	(1.45)	(7.59%)	7.72%
LiteFlowNet [24]	(1.62)	(5.58%)	9.38%
PWC-Net [52]	(2.16)	(9.80%)	9.60%
ContinualFlow_ROB[†§] [42]	–	–	10.03%
MirrorFlow [25]	–	9.98%	10.29%
FlowNet2 [26]	(2.30)	(8.61%)	10.41%

Table 6. **KITTI Optical Flow 2015**: Average end-point error (EPE) and outlier rates (Fl-Noc and Fl-all).

Method	Type	Sintel Training	
		Clean	Final
IRR-PWC (Ours)	supervised	**0.712**	**0.669**
FlowNet-CSSR [27]	supervised	0.703	0.654
OccAwareFlow [58]	unsupervised	0.54	0.48
Back2FutureFlow [30]	unsupervised	0.49	0.44
MirrorFlow [25]	estimated	0.390	–

Table 7. Occlusion estimation results on Sintel Training.

vations are that our model *(i)* converges much faster than the baseline and *(ii)* overfits to the training split less, demonstrating much better accuracy on the test set despite slightly higher error on training split. This highlights the benefit of our IRR scheme: better generalization *even on the training domain* as well as across datasets.

4.5. Occlusion estimation

We finally evaluate the accuracy of occlusion estimation on the Sintel training set as no public benchmarks are available for the task. Table 7 shows the comparison with state-of-the-art algorithms. Supervised methods are trained on FlyingChairs and FlyingThings3D; unsupervised methods are trained on Sintel without the use of ground truth. We achieve state-of-the-art accuracy with far fewer parameters (6.00M instead of 110M) and much simpler training schedules than the previous state of the art [27].

5. Conclusion

We proposed an iterative residual refinement (IRR) scheme based on weight sharing for generic optical flow networks, with additional components for bi-directional estimation and occlusion estimation. Applying our scheme on top of two representative flow networks, FlowNet and PWC-Net, significantly improves flow accuracy with a better generalization while even reducing the number of parameters in case of PWC-Net. We also show that our design choice of jointly estimating occlusion together with flow brings accuracy improvements on both domains, setting the state of the art on public benchmark datasets. We believe that our powerful IRR scheme can be combined with other baseline networks and can form the basis of other follow-up approaches, including multi-frame methods.

References

[1] Aria Ahmadi and Ioannis Patras. Unsupervised convolutional neural networks for motion estimation. In *ICIP*, pages 1629–1633, 2016. 2

[2] Luis Álvarez, Rachid Deriche, Théodore Papadopoulo, and Javier Sánchez Pérez. Symmetrical dense optical flow estimation with occlusions detection. *Int. J. Comput. Vision*, 75(3):371–385, 2007. 2, 3, 4

[3] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, pages 3762–3769, 2014. 6

[4] Min Bai, Wenjie Luo, Kaustav Kundu, and Raquel Urtasun. Exploiting semantic information and deep matching for optical flow. In *ECCV*, volume 6, pages 154–170, 2016. 2

[5] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow Fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*, pages 4015–4023, 2015. 2

[6] Christian Bailer, Kiran Varanasi, and Didier Stricker. CNN-based patch matching for optical flow with thresholded hinge embedding loss. In *CVPR*, pages 2710–2719, 2017. 2, 8

[7] Coloma Ballester, Lluís Garrido, Vanel Lazcano, and Vicent Caselles. A TV-L1 optical flow method with occlusion detection. In *DAGM*, pages 31–40, 2012. 2, 4

[8] Michael J. Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Comput. Vis. Image Und.*, 63(1):75–104, 1996. 2, 3

[9] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, volume 4, pages 25–36, 2004. 2, 3

[10] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, volume 6, pages 611–625. 2012. 1, 6, 8

[11] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *ICCV*, pages 1114–1123, 2017. 1

[12] Qifeng Chen and Vladlen Koltun. Full Flow: Optical flow estimation by global optimization over regular grids. In *CVPR*, pages 4706–4714, 2016. 2

[13] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. SegFlow: Joint learning for video object segmentation and optical flow. In *ICCV*, pages 686–695, 2017. 1

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 1, 2, 3, 5, 6, 7, 8

[15] Raghudeep Gadde, Varun Jampani, and Peter V. Gehler. Semantic video CNNs through representation warping. In *ICCV*, pages 4463–4472, 2017. 1

[16] David Gadot and Lior Wolf. PatchBatch: A batch augmented loss for optical flow. In *CVPR*, pages 4236–4245, 2016. 2

[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 1, 6, 8

[18] Spyros Gidaris and Nikos Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *CVPR*, pages 7187–7196, 2017. 3

[19] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE T. Pattern Anal. Mach. Intell.*, 38(1):142–158, Jan. 2016. 1

[20] Fatma Güney and Andreas Geiger. Deep discrete flow. In *ACCV*, volume 4, pages 207–224, 2016. 2

[21] Adam W. Harley, Konstantinos G. Derpanis, and Iasonas Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *ICCV*, pages 5048–5057, 2017. 4

[22] Yinlin Hu, Yunsong Li, and Rui Song. Robust interpolation of correspondences for large displacement optical flow. In *CVPR*, pages 4791–4799, 2017. 2

[23] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine PatchMatch for large displacement optical flow. In *CVPR*, pages 5704–5712, 2016. 2

[24] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Lite-FlowNet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, pages 8981–8989, 2018. 1, 2, 3, 4, 5, 7, 8

[25] Junhwa Hur and Stefan Roth. MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation. In *ICCV*, pages 312–321, 2017. 2, 3, 4, 8

[26] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, pages 1647–1655, 2017. 1, 2, 3, 6, 7, 8

[27] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *ECCV*, volume 12, pages 626–643. 2018. 2, 3, 6, 7, 8

[28] Serdar Ince and Janusz Konrad. Occlusion-aware optical flow estimation. *IEEE T. Image Process.*, 17(8):1443–1451, Aug. 2008. 2, 3, 4

[29] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*2015*, pages 2017–2025. 3

[30] Joel Janai, Fatma Güney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, volume 16, pages 713–731. 2018. 3, 4, 8

[31] Wei-Sheng Lai, Jia-Bin Huang, and Ming-Hsuan Yang. Semi-supervised learning for optical flow with generative adversarial networks. In *NIPS*2017*, pages 354–364. 2

[32] Yu Li, Dongbo Min, Michael S. Brown, Minh N. Do, and Jiangbo Lu. SPM-BP: Sped-up PatchMatch belief propagation for continuous MRFs. In *ICCV*, pages 4006–4014, 2015. 2

[33] Yu Li, Dongbo Min, Minh N. Do, and Jiangbo Lu. Fast guided global interpolation for depth and motion. In *ECCV*, volume 3, pages 717–733, 2016. 2

[34] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *CVPR*, pages 2811–2820, 2018. 3

[35] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, pages 1132–1140, 2017. 5

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 1

[37] Daniel Maurer and Andrés Bruhn. ProFlow: Learning to predict optical flow. In *BMVC*, 2018. 8

[38] Daniel Maurer, Nico Marniok, Bastian Goldluecke, and Andrés Bruhn. Structure-from-Motion-Aware PatchMatch for adaptive optical flow estimation. In *ECCV*, volume 8, pages 575–592. 2018. 8

[39] Nikolaus Mayer, Eddy Ilg, Philip Häusser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, pages 4040–4048, 2016. 6

[40] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, pages 7251–7259, 2018. 2, 4

[41] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015. 1, 8

[42] Michal Neoral, Jan Šochman, and Jiří Matas. Continual occlusions and optical flow estimation. In *ACCV*, 2018. 3, 4, 8

[43] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *CVPR*, pages 6819–6828, 2018. 1

[44] Jiahao Pang, Wenxiu Sun, Jimmy SJ. Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV Workshops*, pages 878–886, 2017. 3

[45] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, pages 2720–2729, 2017. 1, 2, 3

[46] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B. Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *WACV*, pages 2077–2086, 2019. 8

[47] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, pages 1495–1501, 2017. 2

[48] Jérôme Revaud, Philippe Weinzaepfel, Zaïd Harchaoui, and Cordelia Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *ICCV*, pages 1164–1172, 2015. 2

[49] René Schuster, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. FlowFields++: Accurate optical flow correspondences meet robust interpolation. In *ICIP*, pages 1463–1467, 2018. 8

[50] Deqing Sun, Ce Liu, and Hanspeter Pfister. Local layering for joint motion estimation and occlusion detection. In *CVPR*, pages 1098–1105, 2014. 2, 3, 4

[51] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *Int. J. Comput. Vision*, 106(2):115–137, Jan. 2014. 2, 3

[52] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[53] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of CNNs for optical flow estimation. *IEEE T. Pattern Anal. Mach. Intell.*, 2019, to appear. 6, 8

[54] Jian Sun, Yin Li, Sing Bing Kang, and Heung-Yeung Shum. Symmetric stereo matching for occlusion handling. In *CVPR*, pages 399–406, 2005. 3

[55] Jonathan J. Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*2014*, pages 1799–1807. 1

[56] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *CVPR*, pages 5622–5631, 2017. 3

[57] Markus Unger, Manuel Werlberger, Thomas Pock, and Horst Bischof. Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *CVPR*, pages 1878–1885, 2012. 2, 3, 4

[58] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *CVPR*, pages 4884–4893, 2018. 2, 3, 4, 8

[59] Jonas Wulff, Laura Sevilla-Lara, and Michael J. Black. Optical flow in mostly rigid scenes. In *CVPR*, pages 6911–6920, 2017. 2, 8

[60] Jiangjian Xiao, Hui Cheng, Harpreet S. Sawhney, Cen Rao, and Michael A. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *ECCV*, volume 1, pages 211–224, 2006. 2, 3, 4

[61] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *CVPR*, pages 5807–5815, 2017. 2, 8

[62] Yanchao Yang and Stefano Soatto. S2F: Slow-to-fast interpolator flow. In *CVPR*, pages 3767–3776, 2017. 8

[63] Jason J. Yu, Adam W. Harley, and Konstantinos G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV Workshops*, volume 3, pages 3–10, 2016. 2

[64] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *CVPR*, pages 4141–4150, 2017. 1

[65] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G. Hauptmann. Guided optical flow learning. In *CVPR 2017 Workshops*, 2017. 2

[66] Yi Zhu and Shawn D. Newsam. DenseNet for dense flow. In *ICIP*, pages 790–794, 2017. 2

[67] Shay Zweig and Lior Wolf. InterpoNet, a brain inspired neural network for optical flow dense interpolation. In *CVPR*, pages 6363–6372, 2017. 2