Normalized Convolution Upsampling for Refined Optical Flow Estimation

Anonymous WACV submission

Paper ID 861

Abstract

Optical flow is a regression task where convolutional neural networks (CNNs) have led to major breakthroughs. However, this comes at major computational demands due to the use of cost-volumes and pyramidal representations. This was mitigated by producing flow predictions at quarter the resolution, which are upsampled using bilinear interpolation during test time. Consequently, fine details are usually lost and post-processing is needed to restore them. We propose to use joint upsampling approaches to make the full-resolution flow available for coarse-to-fine optical flow CNNs during training. Experiments show that this leads to significant improvements in the estimated flow even if the vanilla bilinear interpolation was used. Moreover, we propose an efficient upsampler (NCUP) based on the normalized convolutional neural networks that outperforms all existing approaches in comparison when trained with PWC-Net. Finally, we test our upsampler with a recurrent optical flow CNN, i.e. *RAFT, and we achieve state-of-the-art results on Sintel dataset with 6.3% error reduction and 7.5% fewer parameters (see Figure 1). Our approach is also superior to the standard RAFT on the KITTI dataset and shows better generalization capabilities across datasets.*

1. Introduction

Computer vision encompasses a broad range of regression tasks where the goal is to produce numerical output given a visual input. Some of these tasks, *e.g.* depth prediction and optical flow, even require pixel-wise output, which makes theses tasks more challenging. Convolutional neural networks (CNNs) have lead to major breakthroughs in these regression tasks by exploiting deep representations of data. A common design for these regression CNNs is coarse-to-fine where a low-resolution prediction is produced and then progressively upsampled and refined to the full-resolution. This usually requires a large GPU memory, especially at finer stages as the spatial dimensionality grows. Therefore, the scale of these networks has been throttled by the availability of computational resources, which has been mostly mitigated either by limiting the depth of the networks or reducing the resolution of the data.

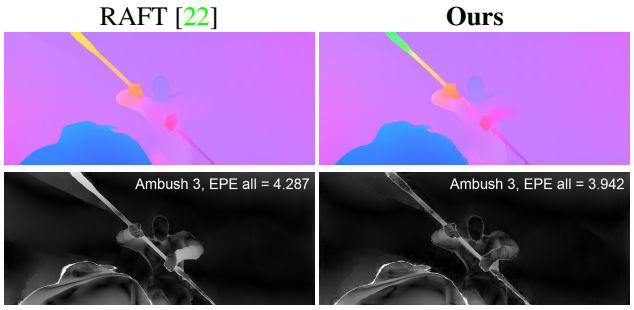
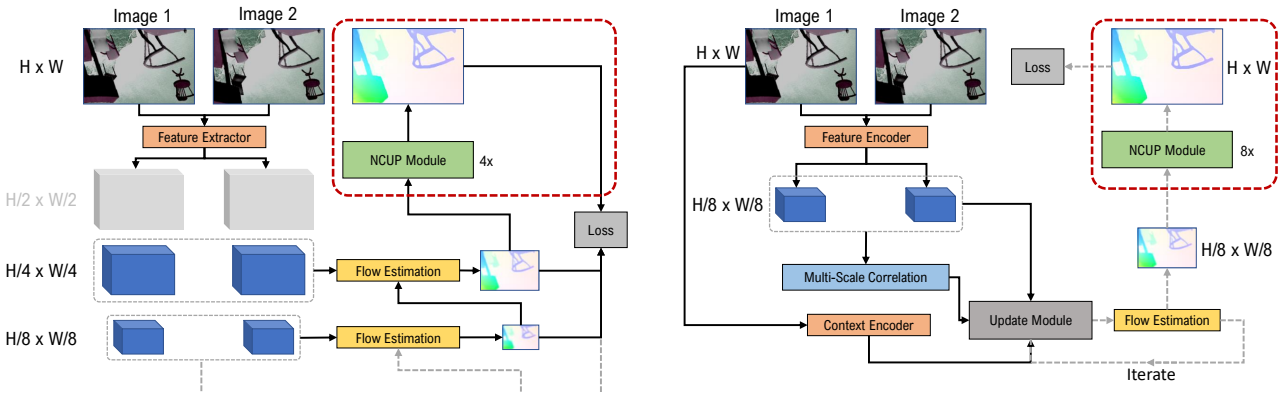


Figure 1: An example from the Sintel [4] test set that shows the flow improvement achieved by our proposed upsampler NCUP incorporated with RAFT [22].

As an example, the early work on CNN-based depth estimation in [6] employed an encoder/decoder network where the training datasets were downsampled to half the resolution to fit into the available GPU memory. Similarly, the prevalent optical flow estimation network, FlowNet [10], trains on a quarter of the full resolution and uses bilinear interpolation to restore the full-resolution during test time. This practice has been preserved in subsequent optical flow CNNs, particularly with the increased complexity of these networks and the emergence of the computationally expensive cost+volumes and pyramidal representations [10, 20, 14]. Nonetheless, pyramid levels with full and half the resolution were not utilized as they would not fit on the available GPU memory. Unfortunately, operating on a fraction of the full-resolution leads to loss of fine details, which might be crucial in certain tasks.

To alleviate these shortcomings of coarse-to-fine approaches, several joint image upsampling approaches have been applied as post-processing to the output from optical flow and depth estimation networks [16, 19, 24]. These approaches substitute the bilinear interpolation and they utilize RGB images as guidance to perform adaptive upsampling for the predicted flow that preserves edges and fine details. The key idea of theses approaches is to use a guid-



(a) Coarse-to-fine (b) Recurrent

Figure 2: An illustration for how we train our proposed normalized convolution upsampler (NCUP) with coarse-to-fine and recurrent optical flow networks. In coarse-to-fine CNNs, *e.g.* PWCNet [20] in **(a)**, the flow is estimated at different levels of a pyramid of features. However, pyramid levels with full and half the resolution are not utilized as it is not feasible to fit them in GPU memory. We propose to upsample the flow to the full-resolution during training using our proposed approach leading to refined flow predictions. In recurrent CNNs, *e.g.* RAFT [22] in **(b)**, the full-resolution flow needs to be available after each iteration. We replace the convex combination upsampler in RAFT, with our more compact upsampler NCUP and we achieve state-of-the-art results using fewer parameters.

ance modality, *e.g.* RGB images, to guide the upsampling of a target modality, *e.g.* flow fields or depth values. However, these approaches act as post-processing stage and are trained separately from the network of the original task, omitting potential benefits from training them end-to-end. Therefore, we propose to train these joint upsampling approaches within the coarse-to-fine optical flow CNNs, *e.g.* FlowNet, PWCNet, in an end-to-end fashion to allow optical flow networks to exploit the fine details during training. Moreover, we propose a novel joint upsampling approach (NCUP) based on the normalized convolutional neural networks [8, 9] that is more efficient (2k parameters) and outperforms other joint upsampling approaches in comparison on the task of end-to-end optical flow upsampling. An illustration for the proposed setup is shown in Figure 2a.

Another category of optical flow networks that emerged recently is based on recurrent networks [13, 22], where the predicted flow is iteratively refined. This requires the availability of the flow in full-resolution at the end of each iteration. The bilinear interpolation was used for this purpose in [13], while a learnable convex combination upsampler was used in [22]. However, this convex upsampler performs the upsampling with a scaling factor of 8 in a single-shot with a limited kernel support of 3×3. Moreover, it has a large number of parameters which encompasses approximately 10% of the entire network. We replace this convex combination module with our efficient upsampler that performs the upsampling at multi-scales, leading to state-of-the-art results on Sintel dataset [4], better results on the KITTI dataset [18], better generalization capabilities, and using 5 times fewer parameters. Figure 2b shows an illustration for setup of recurrent networks, where we replace the upsampling module with our proposed upsampler.

Our Contributions can be summarized as follows:

- We propose to include the full-resolution flow during the training of coarse-to-fine optical flow CNNs through several upsampling approaches, and we show experimentally that the estimated flow is improved.
- We propose a joint upsampling approach (NCUP) that is light-weight and outperforms all approaches in comparison, when trained end-to-end with PWCNet [20].
- When we replace the convex combination module in RAFT [22] with our upsampler, we achieve state-of-the-art results on Sintel [4] dataset, and perform better on the KITTI [18] dataset.
- We show that our upsampling approach has better generalization capabilities than the convex combination module, when trained on FlyingThings3D [17] and evaluated on Sintel [4] and KITTI [18].

2. Related Work

CNN-based Optical Flow Deep learning recently surfaced as a plausible substitute for the classical optimization-based optical flow approaches [25, 1, 11]. CNNs can be trained to directly predict optical flow given two images avoiding explicitly designing an optimization objective

manually in classical approaches. FlowNet [10] introduced the first CNN for optical flow estimation that is trained end-to-end in a coarse-to-fine fashion. Subsequent approaches followed the same coarse-to-fine scheme where FlowNet2 [14] proposed a stacked version of FlowNet, PWCNet [20] introduced a pyramidal variation, and LiteFlowNet [12] designed a light-weight cascaded network at each pyramid level. VCN [26] proposed several improvements for matching cost-volumes to expand their reciptive field and they added support for multi-dimensional similarities.

Recently, several recurrent approaches were proposed where the flow is iteratively refined similar to the optimization-based approaches. A preliminary flow prediction is produced at the first iteration and it is refined for a number of iterations. IRR [13] proposed to use either FlowNetS [10] or PWCNet [20] as a recurrent unit that iteratively estimates the residual flow from the previous iteration. However, the number of iterations was limited either by the size of the network in FlowNet, or the number of pyramid levels in PWCNet. RAFT [22] introduced a light-weight recurrent unit that is coupled with a GRU cell [5] as an update operator. This cell allowed performing more iterations and led to more refined flow predictions at a relatively low computational cost.

Joint Image Upsampling The notion of joint (guided) image upsampling is to use a guidance image to steer the upsampling of another target image, where both the guidance and the target images could be from the same or different modalities. Several classical approaches were proposed that are based on variations of the bilateral filtering [27, 2]. Li *et al*. [16] proposed a CNN-based architecture for joint image filtering that can be applied to joint upsampling. They employed two sub-networks for target and guidance features extraction followed by a fusion block. Wu *et al*. [24] proposed a trainable guided filtering network that was applied to clone the behavior of several vision tasks. Su *et al*. [19] proposed pixel-adaptive convolutions that modifies the convolution filter with a spatially varying kernel. Wannenwetsch *et al*. [23] extended the pixel-adaptive convolutions to incorporate pixel-wise confidences.

Optical Flow Upsampling For coarse-to-fine networks, FlowNet [10] suggested the use of an iterative variational approach [3] to produce the full-resolution flow during test time. However, this approach is computationally expensive and is not possible to train jointly with the network. For recurrent networks, the full-resolution flow is required during the training at the end of each iteration. IRR [13] attempted a residual upsampling block, but found to be futile with optical flow and they used the bilinear interpolation. RAFT [22] produces the flow in 1/8 of the full-resolution and employed a convex combination upsampler to construct the full-resolution flow. However, their upsampler has a limited receptive field and has a large number of parameters.

For coarse-to-fine networks, we propose to employ trainable joint upsampling approaches to upsample the flow during training. Moreover, we propose a joint upsampling approach (NCUP) based on the normalized convolutional neural networks [8, 9] that is more efficient and performs better than other approaches in comparison on optical flow upsampling. Different to other joint upsampling approaches, our upsampler estimates the guidance on the low-resolution data instead of the full-resolution ones, which leads to a smaller guidance sub-network. For recurrent networks, *i.e*. RAFT [22], we replace the convex module with our proposed upsampler, which performs the upsampling at multi-scales and has 5 times fewer parameters. This modification leads to state-of-the-art results on Sintel [4] dataset with 6.3% error reduction, while using 7.5% less number of parameters. We also perform better on the KITTI dataset [18] than the standard RAFT. Moreover, our approach shows better generalization capabilities when trained on FlyingThings [17] and tested on Sintel [4] and [18] datasets.

3. Approach

In joint image upsampling task, it is desired to train a network θ to upsample a low-resolution input $\mathbf{I}_{\text{LR}}$ to a high-resolution output $\mathbf{I}_{\text{HR}}$, guided by some high-resolution guidance data $\mathbf{g}_{\text{HR}}$; $\theta : \mathbf{I}_{\text{LR}} \rightarrow \mathbf{I}_{\text{HR}}|\mathbf{g}_{\text{HR}}$. The guidance data is typically the RGB image, but can be of any modality or even intermediate feature representations from a CNN. In this section, we briefly describe the normalized convolutional neural networks [8] followed by our proposed Normalized Convolution Upsampler (NCUP).

3.1. Normalized Convolutional Neural Networks

Eldesokey *et al*. [8, 9] proposed the normalized convolution layer, which is a sparsity-aware convolution operator that was used to interpolate a sparse depth map on an irregular grid. More formally, they learn an interpolation function $\theta : \tilde{\mathbf{I}}_{\text{HR}} \rightarrow \mathbf{I}_{\text{HR}}|\tau(\tilde{\mathbf{I}}_{\text{HR}})$, where $\tilde{\mathbf{I}}_{\text{HR}}$ is a sparse high-resolution input with missing pixels, and $\tau()$ is a thresholding operator that produces ones at pixels where data is present and zeros otherwise. They recently proposed to replace the thresholding operator τ with a CNN Φ that predicts pixel-wise weights from the sparse input $\theta : \tilde{\mathbf{I}}_{\text{HR}} \rightarrow \mathbf{I}_{\text{HR}}|\Phi(\tilde{\mathbf{I}}_{\text{HR}})$ in a self-supervised manner [7]. The high-resolution output $\mathbf{I}_{\text{HR}}$ is predicted by a cascade of L normalized convolution layers, where the output for layer $l \in \{1 \dots L\}$, is calculated as:

$$\mathbf{I}^{l}_{\text{HR}}(\mathbf{x}) = \frac{\sum_{\mathbf{m}\in\mathbb{R}^2} \mathbf{I}^{l-1}_{\text{HR}}(\mathbf{x}-\mathbf{m})\, \mathbf{w}^{l-1}(\mathbf{x}-\mathbf{m})\, \mathbf{a}^{l}(\mathbf{m})}{\sum_{\mathbf{m}\in\mathbb{R}^2} \mathbf{w}^{l-1}(\mathbf{x}-\mathbf{m})\, \mathbf{a}^{l}(\mathbf{m})}\,, \tag{1}$$

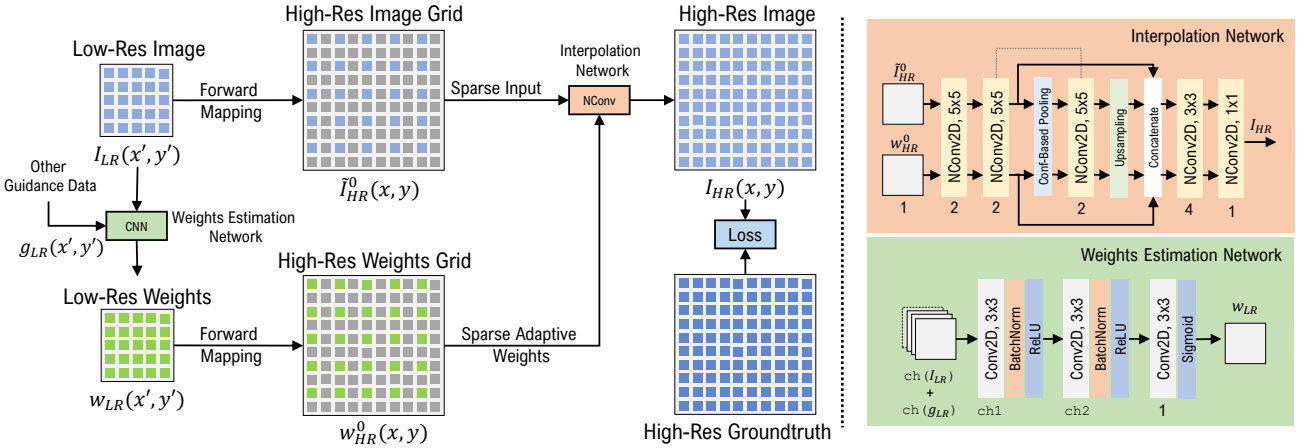


Figure 3: An illustration of our proposed joint upsampling approach (NCUP). First, The low-resolution image is mapped to the high-resolution grid using forward mapping. Pixel-wise weights for the low-resolution image are produced by a *weights estimation network the* (green block) which takes the low-resolution image and other auxiliary data as input. The weights are mapped to the high-resolution grid in a similar fashion using forward mapping. Next, an *interpolation network* that encompasses a cascade of normalized convolution layers (the orange block) receives the high-resolution grid as well as the weights and produce the high-resolution image. Note that the notation `ch()` denotes number of channels.

where $\mathbf{x}, \mathbf{m}$ are the spatial coordinates of the image, $\mathbf{I}^0_{\text{HR}} = \tilde{\mathbf{I}}_{\text{HR}}$, $\mathbf{w}^0 = \Phi(\tilde{\mathbf{I}}_{\text{HR}})$, and $\mathbf{a}^l$ is the interpolation kernel at layer l. The weights are propagated between layers as:

$$\mathbf{w}^l(\mathbf{x}) = \frac{\sum_{\mathbf{m}\in\mathbb{R}^2} \mathbf{w}^{l-1}(\mathbf{x}-\mathbf{m})\, \mathbf{a}^l(\mathbf{m})}{\sum_{\mathbf{m}\in\mathbb{R}^2} \mathbf{a}^l(\mathbf{m})} \quad , \qquad (2)$$

At the final layer L, the high-resolution output is produced $\mathbf{I}_{\text{HR}} = \mathbf{I}^L_{\text{HR}}$.

3.2. Formulating Upsampling as a Sparse Problem

Typically, the standard interpolation operations, *e.g.* bilinear, bicubic, employ backward mapping to ensure that each pixel in the output is assigned a value. Contrarily, if forward mapping is used, a sparse grid is formed in the output. Fortunately, normalized convolution layers were demonstrated to perform well with irregular sparse grids, *e.g.* depth completion, sparse optical flow, and, consequently, can be used to interpolate regular sparse grids.

Given a low-resolution input image $\mathbf{I}_{\text{LR}}$, a high-resolution sparse grid $\tilde{\mathbf{I}}_{\text{HR}}$ can be constructed using forward mapping. The forward mapping from the low-resolution grid coordinates (x', y') to the high-resolution grid (x, y) for a scaling factor s can be realized as:

$$(x, y) = (\text{round}(s \cdot x'), \text{round}(s \cdot y')) \; \forall \; (x', y') \qquad (3)$$

Note that the high-resolution grid is regular when $s \in \mathbb{N}$.

The initial pixel-wise weights $\mathbf{w}^0$ required for the normalized convolution network can be estimated using a weights estimation network Φ similar to [7]. But different from [7] and other existing joint upsampling approaches, we estimate the pixel-wise weights from the low-resolution guidance image, not the high-resolution one. Predicting weights for the low-resolution image contributes to making the weights estimation network much smaller and shallower, leading to a more efficient upsampling. For instance, the entire upsampling network that we use with coarse-to-fine optical flow networks, *e.g.* FlowNet and PWCNet, has only 2k parameters (see Figure 3 where `ch1`=16 and `ch2`=8), while being able to outperform other approaches with at least one order of magnitude more parameters.

Another difference from [7] is that we employ other modalities, *e.g.* RGB input image, intermediate CNN features, as guidance for the weights estimation network similar to the existing joint upsampling approaches [16, 19]; $\Phi([\mathbf{I}_{\text{LR}}, \mathbf{g}_{\text{LR}}])$. This allows exploiting other modalities to adapt the weights based on the context. The output from the weights estimation network is also transformed to the high-resolution grid using the forward mapping.

Essentially, we train an upsampling network $\theta : \mathbf{I}_{\text{LR}} \rightarrow \mathbf{I}_{\text{HR}}|\Phi([\mathbf{I}_{\text{LR}}, \mathbf{g}_{\text{LR}}])$, where the sparse high-resolution grid $\tilde{\mathbf{I}}_{\text{HR}}$ is an intermediate stage generated by applying forward mapping to the the low-resolution input. The pixel-wise weights are predicted using a CNN from the low-resolution input and any other guidance data. The weights are similarly mapped to the high-resolution grid using forward mapping. Finally, a cascade of normalized convolution layers is applied to interpolate the missing values in the sparse high-resolution grid. An illustration of the whole pipeline is shown in Figure 3.

3.3. Weights Estimation Network

Since the weights are estimated for the low-resolution input, the receptive field of the weights estimation network can be quite small. Therefore, we use two convolution layers with a 3×3 filters followed by Batch Normalization and ReLU activation. The number of channels per layer is determined based on the guidance data that is used. When RGB images are used, we use 16 and 8 channels for the two convolution layers, while we use 64 and 32 channels when intermediate CNN features are used as guidance (`ch1` and `ch2` values in Figure 3). A last convolution layer with a 1×1 filter is applied to produce the same number of channels as the low-resolution input $\mathbf{I}_{\text{LR}}$. Finally, a Sigmoid activation is applied to produce valid non-negative weights. Other function with a non-negative co-domain can be used, *e.g*. Softplus, but the Sigmoid function was found to achieve the best results. The estimated weights are transformed to the high-resolution grid using forward mapping as well.

3.4. Interpolation Network

We use a U-Net shaped normalized convolution network as described in [8]. However, we perform downsampling only once, *i.e*. we use two scales instead of four in [8], since the sparsity in our case is significantly lower than the LiDAR depth completion problem they were solving. This leads to a smaller network with 224 parameters instead of 480 parameters in [8]. The interpolation network receives the high-resolution image grid $\tilde{\mathbf{I}}_{\text{HR}}$ and the weights grid $\mathbf{w}^0$ as an input. The weights are propagated and updated within the interpolation network until the final dense output $\mathbf{I}_{\text{HR}}$ is produced at the final layer.

3.5. Optical Flow Upsampling

Optical flow is represented as two channels for vertical and horizontal flow field. We process the two channels jointly within the weights estimation network, *i.e*. $\texttt{ch}(I_{\text{LR}}) = 2$ in Figure 3. However, for the interpolation network, the two channels are processed separately and then concatenated. In coarse-to-fine optical flow estimation networks, *e.g*. FlowNet [10] and PWCNet [20], the flow is produced at quarter the resolution. We attach the upsampling module to the optical flow estimation network to upsample the flow from $H/4 \times W/4$ to $H \times W$

Typically, the multi-scale loss is employed in coarse-to-fine networks:

$$\sum_{p \in P} \alpha_p \left| \mathbf{f}^p - \mathbf{f}^p_{\text{GT}} \right|^2 \ , \tag{4}$$

where $\mathbf{f}^p$ is the flow estimation at pyramid level p in PWCNet or resolution p in FlowNet, where $P = \{3, 4, 5, 6, 7\}$, and $\mathbf{f}^p_{\text{GT}}$ is the corresponding downsampled groundtruth. The choice of α_p's were empirically determined as $\{0.32, 0.08, 0.02, 0.01, 0.005\}$. Note that that $p = 1, p = 2$, where not considered during training as explained earlier. We consider another level/scale in the loss for the full-resolution flow, *i.e*. we set $P = \{1, 3, 4, 5, 6, 7\}$, and we found empirically that the best performance is obtained when $\alpha_1 = 0.02$ for most methods. This denotes that the flow is upsampled by a factor of 4 from quarter the resolution to the full-resolution. For the recurrent network RAFT, we use their proposed loss [22].

4. Experiments

In this section, we evaluate our proposed joint upsampling approach with *two types* of optical flow estimation CNNs: coarse-to-fine and recurrent networks.

4.1. Joint Upsampling for Coarse-to-fine Networks

We choose two of the most popular coarse-to-fine optical flow CNNs, *i.e*. FlowNet [10] and PWCNet [20]. Different joint upsampling approaches are attached to the two networks and they are trained end-to-end as illustrated in Figure 2a. The joint upsampling approaches that we compare against are DJIF [16], PAC [19], the convex combination from RAFT [22] which we refer to as ConvComb, and the bilinear interpolation. We train only on the FlyingChairs [10] as its spatial resolution is smaller than its counterparts allowing training memory-demanding joint upsampling approaches. For instance, PWCNet trained with PAC fully occupy a 32 GB V100 GPU when trained of FlyingChairs with a batch size of 3. We use the official PyTorch implementations provided by the corresponding authors.

Experimental Setup We initialize FlowNetS and PWCNet using pretrained models on the FlyingChairs dataset, while the joint upsampling approaches are initialized randomly. We train each network for 60 epochs with an initial learning rate of 0.0001 that is halved at epochs $\{20, 30, 40, 50, 55\}$. Since we can only fit a batch size of 3 for PAC on a 32GB V100 GPU, we use a batch size of 4 for all other approaches for a fair comparison. We use data augmentation as described in [13].

Quantitative Results Table 1 summarizes the results for coarse-to-fine networks. All upsampling approaches lead to performance gains demonstrating the advantage from making the full-resolution flow available for coarse-to-fine networks during training. On PWCNet, our upsampler achieves the best improvement over the baseline despite having at least one order of magnitude lower parameters than its counterparts, while other approaches performs comparably well. On FlowNetS, our upsampler performs second best with a small margin to PAC. We believe that the larger model of PAC allows it to refine the poor predictions from FlowNetS slightly better than our upsampler.

Qualitative Results A qualitative example for different approaches on the FlyingChairs dataset is shown if Figure 4. All upsampling approaches make edges and details more

	Baseline	Bilinear	DJIF [16]	PAC [19]	ConvComb [22]	**Ours**
PWCNet [20]	1.69	1.58	1.51	*1.50*	1.52	**1.46**
FlowNetS [10]	2.53	2.23	2.16	**2.11**	2.16	*2.13*
Relative Params.	+0k	+0k	+56k	+183k	*+44k*	**+2k**

Table 1: Summary of the results when two coarse-to-fine optical flow networks are trained end-to-end with joint upsampling approaches. The stated results are the Average End-Point Error (AEPE) on the FlyingChairs [10] test set. The best results are shown in **Bold** and the second best in *Italics*.

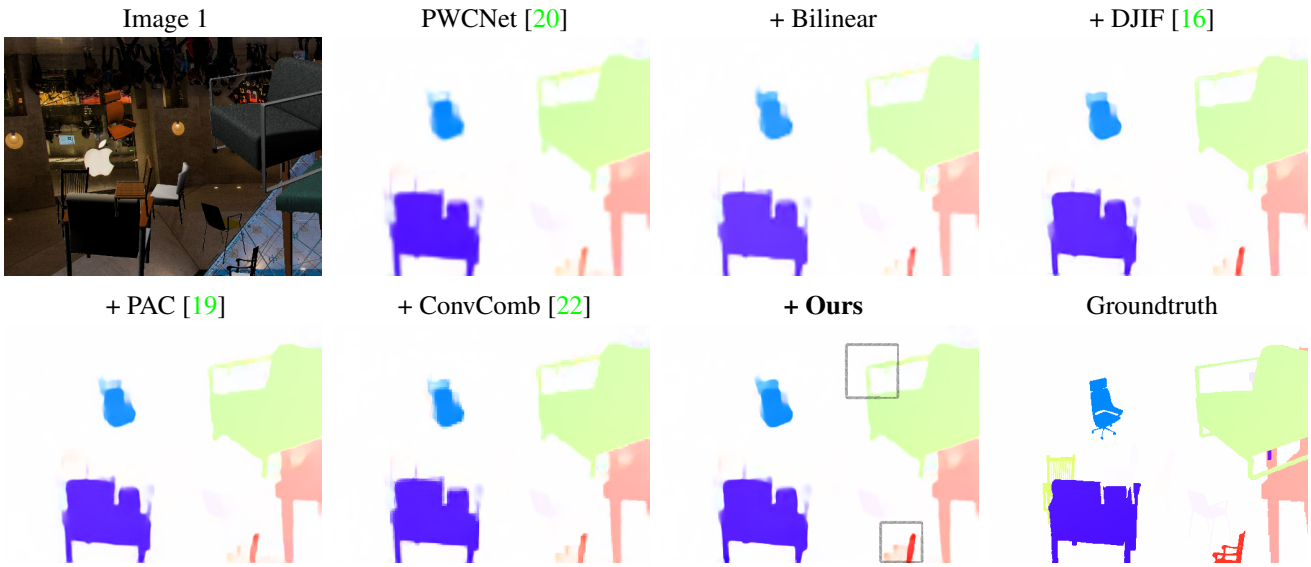


Figure 4: A qualitative example from the FlyingChairs [10] dataset when PWCNet [20] is trained end-to-end using different joint upsampling approaches. Better viewed on a computer display.

sharp and defined compared to the standard PWCNet as a result of making the full-resolution flow available during training. Nonetheless, PAC and our upsampler tend to produce sharpest results amongst all. However, our upsampler does a better job preserving small objects in some situations such as the red chair at the bottom of the scene.

4.2. Joint Upsampling for Recurrent Networks

We test our proposed upsampler as a substitute for the convex combination upsampler in the recurrent optical flow approach RAFT [22]. The convex upsampler which has $\sim 500k$ parameters is removed and replaced with our upsampler constituting $\sim 100k$ parameters. We use the output from the GRU cell, which has 128 channels as guidance data as they suggested in addition to the low-resolution flow. For efficiency, we upsample the flow from 1/8 to 1/4 the full-resolution and then use our upsampler for restoring the full-resolution.

Experimental Setup We initialize the network using the pretrained weights provided by the authors [22]. We use the same training hyperparameters as described in [22] except for the weight decay that we set to 0.00005 and we only train for 50k iterations. For Sintel [4], we do not include FlyingThings3D and HD1k during finetuning.

Benchmark Comparison Table 2 shows the results for Sintel and KITTI benchmarks. On the KITTI benchmark, we outperform the standard RAFT with a slight margin despite having 7.5% fewer parameters. On the Sintel benchmark, we outperform the standard RAFT with a 6.3% error reduction on the challenging final pass, while the error is slightly increased by 1.8% on the clean pass. We believe that this performance boost on the final pass is caused by multi-scale interpolation scheme employed by our upsampler that can eliminate large faulty regions in the predicted flow.

Generalization Results To examine the generalization capabilities of our upsampler, we train it on FlyingChairs followed by FlyingThings3D and evaluate it on the training set of Sintel and KITTI. Table 2 shows that our upsampler outperforms the standard RAFT on clean pass of Sintel and KITTI, while it performs slightly worse on the final

Training Dataset	Method	Sintel (Train)		KITTI (Train)		Sintel (Test)		KITTI
		Clean	*Final*	*AEPE*	*Fl-All*	*Clean*	*Final*	(Test)
C+T	PWCNet [20]	2.55	3.93	10.35	33.7	-	-	-
	LiteFlowNet [12]	2.48	4.04	10.39	28.5	-	-	-
	VCN [26]	2.21	3.67	8.36	25.1	-	-	-
	MaskFlowNet [28]	2.25	3.61	-	23.1	-	-	-
	FlowNet2 [14]	2.02	3.54	10.08	30.0	3.96	6.02	-
	RAFT-Small [22]	2.21	3.35	7.51	26.9	-	-	-
	RAFT [22]	*1.43*	**2.71**	*5.04*	*17.4*	-	-	-
	RAFT+NCUP	**1.39**	*2.74*	**4.83**	**17.3**	-	-	-
C+T+S/K	FlowNet2 [14]	(1.45)	(2.01)	(2.30)	(6.80)	4.16	5.74	11.48
	IRR-PWC [13]	(1.92)	(2.51)	(1.63)	(5.30)	3.84	4.58	7.65
	RAFT [22]	(0.77)	(1.20)	(0.64)	(1.50)	2.08	3.41	*5.27*
	RAFT+NCUP†	-	-	(0.59)	(1.32)	-	-	**5.25**
C+T+S+K+H	PWCNet+ [21]	(1.71)	(2.34)	(1.50)	(5.30)	3.45	4.60	7.27
	VCN [26]	(1.66)	(2.24)	(1.16)	(4.10)	2.81	4.40	6.30
	MaskFlowNet [28]	-	-	-	-	2.52	4.17	6.10
	RAFT* [22]	(0.77)	(1.27)	-	-	**1.61**	*2.86*	-
	RAFT+NCUP*	(0.70)	(1.13)	-	-	*1.64*	**2.68**	-

Table 2: Summary for quantitative results when using our upsampler NCUP with RAFT [22]. The best results are shown in **Bold** and the second best in *Italics*. Different datasets are indicated as following: FlyingChairs [10] → C, FlyingThings3D [17] → T, Sintel [4] → S, KITTI-Flow 2015 [18] → K, HD1K [15] → H, and S/K indicates that either Sintel or KITTI were used for training. Results between brackets are training set score and hence not comparable. Note that we did not use FlyingThings3D and HD1K during finetuning for Sintel. † The results for Sintel were not included due to the limited number of runs allowed at the benchmark. * Indicates that warm starts [22] were used.

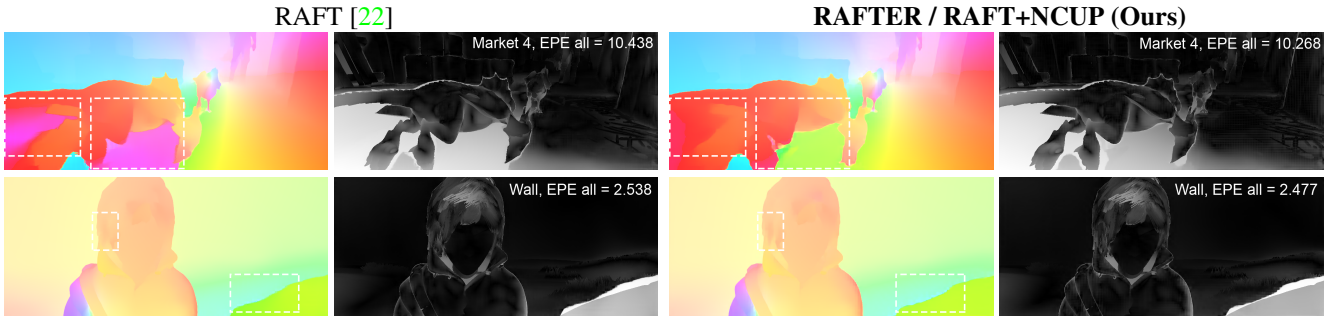


Figure 5: Qualitative examples from the Sintel [4] test set.

pass of Sintel. We believe that the slight degradation on the final pass is due to training the clean and the final pass of FlyingThings3D together without a weighted sampling. However, the large improvement on KITTI significantly indicates that our upsampler posses better generalization.

Qualitative Results Figure 5 shows some qualitative results from the Sintel [4] test set. The use of our upsampler leads to better flow estimations compared to the standard RAFT. The first row shows an example where a large region of faulty flow prediction (the purple region under the dragon) produced by the standard RAFT that is corrected when our proposed upsampler was used. The second row shows another example where the flow is improved at fine details such as the hair. These results clearly demonstrates

Model	AEPE
PWCNet+NCUP (Baseline)	**1.46**
Weights Estimation Network	
Final activation is SoftPlus	1.48
Estimate from High-Res	1.75
Low-Res not used as guidance	1.52
Interpolation Network	
Two downsampling instead of one	1.49
Max instead of Conf. pooling	1.48
Loss Function	
$\alpha_1 = 0.002$	1.48
$\alpha_1 = 0.2$	**1.46**

Table 3: Ablation results on the FlyingChairs [10] test set. The baseline is PWCNet [20] trained with our upsampler NCUP.

the impact of upsampling on the quality of the flow.

4.3. Ablation Study

We conduct an ablation study to justify specific design choices in our proposed approach. Experiments are reported for PWCNet+NCUP on the FlyingChairs [10] test set. Table 3 summarizes the average end-point-error scores for different experiments.

Weights Estimation Network: We replace the final activation with SoftPlus function instead of Sigmoid to get the estimate weights in the range of $[0, \infty[$ instead of $[0, 1]$ produced by the Sigmoid function. The network converges faster when using the SoftPlus function, however the AEPE score is slightly worse. We also attempt to feed the full-resolution guidance data to the weights estimation networks similar to other joint upsampling approaches. The kernel size of the first two convolution layers was increased to 5×5 for a larger receptive field. The results are significantly worse, which is probably because a larger network is needed to exploit the interesting information in the full-resolution data. Finally, we omit the low-resolution flow from being used with guidance data. The results shows that using the low-resolution flow with guidance data contributes significantly to the results.

Interpolation Network: We experiment with two downsamplings instead of one, which indicates that the interpolation is performed at three scales instead of two. The results show that the the best results are achieved when using only one downsampling. We also test the standard max pooling for downsampling instead of the confidence-based pooling proposed in [8]. The results show that the confidence-based pooling perform slightly better than max pooling.

The Loss Function: We experiment with one order of magnitude higher and lower factor α_1 in (4). The results indicates that the choice of $\alpha_1 = 0.02$ and $\alpha_1 = 0.2$ lead to the best results. So, we choose $\alpha_1 = 0.02$ since it works the best for the majority of methods in comparison, but the value of α_1 can be tuned further for our approach.

Figure 6: An example of the predicted weights from NCUP when used with RAFT [22].

4.4. What does our Upsampler Learn?

Figure 6 shows an example of the predicted weights within our upsampler when used with RAFT on the Sintel dataset in comparison to the bilinear interpolation. The estimated weights essentially highlight edges and fine details with low-weight regions separating them. The width of these regions defines to what extent each object is extrapolated and ensures the separability between objects. Based on the design of the interpolation network, the width of these regions is adapted accordingly. On the other hand, solid regions, *e.g.* the girl's face, with no texture are assigned uniform weights acting as averaging. This adaptive behavior shows a great potential for using our upsampling with other regression tasks, where the weights estimation network would learn the upsampling pattern that minimizes the reconstruction error.

5. Conclusion

We introduced an efficient upsampling approach based on the normalized convolutional networks that we incorporated in training coarse-to-fine and recurrent optical flow CNNs. In coarse-to-fine networks, *e.g.* PWCNet, the full-resolution flow was produced by our upsampler during the training leading to the fines flow estimations compared to other joint upsampling approaches in comparison, while having at least one order of magnitude fewer parameters. When trained with the recurrent optical flow network RAFT, it achieved state-of-the-art results on the Sintel dataset, and achieved lower error on the KITTI dataset, while having 400k less parameters. Additionally, our approach showed better generalization capabilities compared to the standard RAFT.

References

[1] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 4015–4023, 2015. 2

[2] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer, 2016. 3

[3] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2010. 3

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 1, 2, 3, 6, 7

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 3

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 1

[7] Abdelrahman Eldesokey, Michael Felsberg, Karl Holmquist, and Michael Persson. Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12014–12023, 2020. 3, 4

[8] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Propagating confidences through cnns for sparse data regression. In *The British Machine Vision Conference (BMVC), Northumbria University, Newcastle upon Tyne, England, UK, 3-6 September, 2018*, 2018. 2, 3, 5, 8

[9] Abdelrahman Eldesokey, Michael Felsberg, and Fahad Shahbaz Khan. Confidence propagation through cnns for guided sparse depth regression. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 2, 3

[10] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015. 1, 3, 5, 6, 7, 8

[11] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981. 2

[12] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8981–8989, 2018. 3, 7

[13] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5754–5763, 2019. 2, 3, 5, 7

[14] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 1, 3, 7

[15] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–28, 2016. 7

[16] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Joint image filtering with deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1909–1923, 2019. 1, 3, 4, 5, 6

[17] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. 2, 3, 7

[18] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018. 2, 3, 7

[19] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11166–11175, 2019. 1, 3, 4, 5, 6

[20] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. 1, 2, 3, 5, 6, 7, 8

[21] 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 transactions on pattern analysis and machine intelligence*, 42(6):1408–1423, 2019. 7

[22] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020. 1, 2, 3, 5, 6, 7, 8

[23] Anne S. Wannenwetsch and Stefan Roth. Probabilistic pixel-adaptive refinement networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[24] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1838–1847, 2018. 1, 3

[25] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1289–1297, 2017. 2

[26] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Advances in neural information processing systems*, pages 794–805, 2019. 3, 7

[27] Qingxiong Yang, Ruigang Yang, James Davis, and David Nistér. Spatial-depth super resolution for range images. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 3

[28] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020. 7