ASFlow: Unsupervised Optical Flow Learning with Adaptive Pyramid Sampling

Shuaicheng Liu, *Member, IEEE,* Kunming Luo, Ao Luo, *Member, IEEE,* Chuan Wang, *Member, IEEE,* Fanman Meng, *Member, IEEE,* and Bing Zeng, *Fellow, IEEE*

***Abstract*—We present an unsupervised optical flow estimation method by proposing an adaptive pyramid sampling in the deep pyramid network. Specifically, in the pyramid downsampling, we propose a Content-Aware Pooling (CAP) module, which promotes local feature gathering by avoiding cross region pooling, so that the learned features become more representative. In the pyramid upsampling, we propose an Adaptive Flow Upsampling (AFU) module, where cross edge interpolation can be avoided, producing sharp motion boundaries. Equipped with these two modules, our method achieves the best performance for unsupervised optical flow estimation on multiple leading benchmarks, including MPI-Sintel, KITTI 2012 and KITTI 2015. Particularly, we achieve EPE=1.5 on KITTI 2012 and F1=9.67% KITTI 2015, which outperform the previous state-of-the-art methods by 16.7% and 13.1%, respectively.**

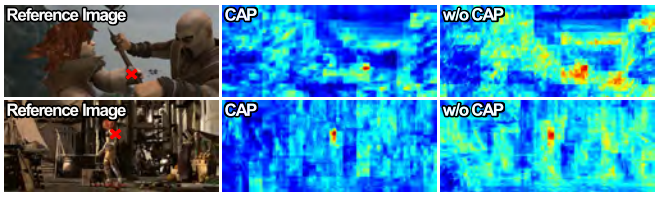
***Index Terms*—Unsupervised Learning, Optical Flow, Pyramid Upsampling, Pyramid Downsampling.**

I. Introduction

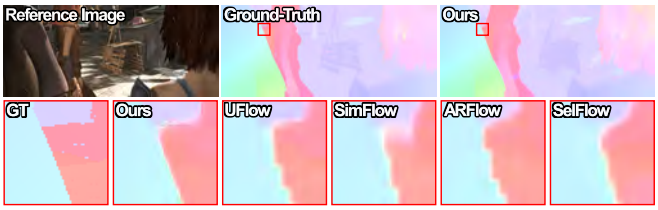
OPTICAL flow estimation is a long-lasting research topic since proposed by Horn and Schunck [1]. It is a fundamental technique for many computer vision applications [2]–[6]. Early methods optimize the pre-defined energy functions with various assumptions [7]–[9] and constraints [10], [11]. The learning-based optical flow methods [12]–[14] become more popular than the traditional variational-based counterparts due to their leading performances in benchmark evaluations and real-time inference speed.

The DNN-based methods can be classified into supervised [15]–[19] and unsupervised [20]–[24] approaches. The training of supervised methods requires ground-truth flow labels, which are hard to obtain. As a result, these models are primarily trained on large-scale synthetic datasets [15], [25], because obtaining ground-truth annotations for real-world scenarios are prohibitively expensive. Consequently, the supervised methods may suffer from domain transfer problems, where the synthesized images are different from the real ones.

In unsupervised methods, ground-truth annotations are not necessary. The photometric loss is optimized by warping one image to the other with predicted optical flows. Without the label guidance, occlusions and motion boundaries need special attention during the unsupervised training process [24], [26].



(a) Feature similarity map with and w/o our content-aware pooling (CAP).



(b) An example from Sintel Clean benchmark.

Fig. 1. Some examples from Sintel Clean benchmark. (a) With our proposed CAP, the learned features are more representative. (b) Compared with previous unsupervised methods, UFlow [24], SimFlow [27], ARFlow [23], and SelFlow [28], our approach produces sharper and more accurate results at motion boundaries.

The pyramid structure is popular in optical flow learning, where global and local motions can be estimated in a coarse-to-fine manner. We notice that there are two components that should be improved in the pyramid structure [18], [19]. One is related to the pyramid downsampling and the other is the upsampling.

In the process of pyramid downsampling, the network adopts striding in convolution (SIC) or the pooling to decrease the feature sizes. However, the striding or pooling is fixed with a rectangular size, which may not be optimal for the feature information gathering. Considering that a rectangle may span different image regions, where multiple irrelevant values are forced to gather together, picking one of them may not be optimal, yielding less representative values. On the other side, in the pyramid upsampling, the flows are interpolated from coarse to fine. However, such an interpolation may cross image edges, resulting in blur effects in the estimated flows. Even worse, such errors will be propagated and aggregated when the scale becomes finer.

Based on the above observations, we propose an Adaptive Pyramid Sampling approach to upgrade the pyramid network structure, including a *Content-Aware Pooling* (**CAP**) module for

Manuscript received July 12, 2021; revised October 4, 2021 and November 3, 2021; accepted November 12, 2021. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61872067, 62031009, 61871087, and 61720106004. (Corresponding author: Shuaicheng Liu)

Shuaicheng Liu, Fanman Meng and Bing Zeng are with School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, China. (e-mail: liushuaicheng@uestc.edu.cn).

Kunming Luo, Ao Luo, and Chuan Wang are with Megvii Research Chengdu, Chengdu, Sichuan, 610095, China.

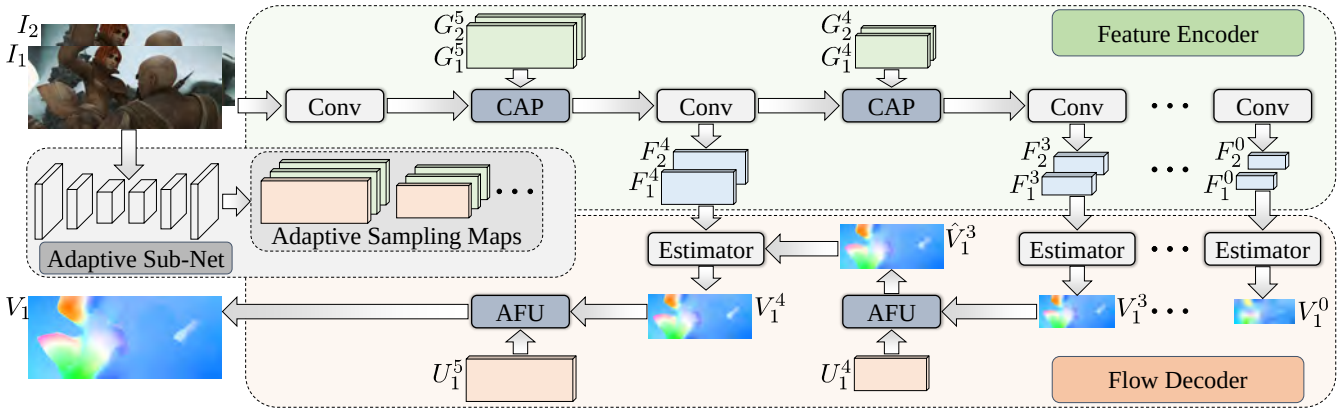


Fig. 2. Illustration of our network, where 'Conv' represents a convolutional block that contains two convolution layers with kernel size 3 and stride 1, 'Estimator' denotes the conventional optical flow estimator, 'CAP' is the proposed Content-Aware Pooling module, and 'AFU' is the proposed Adaptive Flow Upsampling module.

the pyramid downsampling and an *Adaptive Flow Upsampling* (**AFU**) module for the pyramid upsampling. The CAP can automatically group image features, such that similar features can be gathered locally before the downsampling. With our CAP, the learned features become more representative, so as to promote the overall performance. On the other side, the AFU module interpolates the flows adaptively, where cross-edge interpolation can be avoided, leading to sharper flows at motion boundaries. Specifically, in the AFU, we propose a *sampling regularization loss* to constrain the learned adaptive sampling maps, where the upsampled flow fields can better fit the object boundaries.

Fig. 1 provides some visualization results on Sintel Clean dataset. Specifically, Fig. 1a shows some feature similarity maps. We extract features from source and target images. We choose one feature vector at a position (marked in the red cross) from the source image and calculate its similarity with all features at the target image. We plot the similarity as a heat map, where high similarity values are depicted in red. As seen, with our CAP, the feature at the 'red cross' is quite different from the features at the other places. In contrast, without our CAP, features at many different places also have high similarity values. Therefore, our model can learn more representative features with the proposed content-aware pooling. Fig. 1b shows our predicted optical flow compared with other unsupervised methods. As can be seen, with the help of AFU, the interpolation can produce sharp motion boundaries. Equipped with CAP and AFU, the classical pyramid network has been upgraded, producing leading performance both quantitatively and qualitatively when evaluated on the flow benchmarks [25], [29], [30]. To sum up, our main contributions include:

- We propose a Content-Aware Pooling (CAP) module for the pyramid downsampling. The CAP can assemble similar features locally, improving the capability of feature representation substantially.
- We propose an Adaptive Flow Upsampling (AFU) module for the pyramid upsampling, where the blurs caused by cross-edge interpolation can be avoided, yielding sharper motion boundaries.
- We achieve superior performance over the state-of-the-art unsupervised methods, evaluated on multiple leading benchmarks.

II. Related Work

A. Supervised Deep Optical Flow

Supervised methods require the annotated ground-truth flow labels to train the network [17], [31]. FlowNet [15] was first proposed by training a fully convolutional network on the flying chairs dataset [15]. FlowNet2 [32] then proposed to improve the performance by iteratively stacking networks. In order to cover large displacements, SPyNet [16] proposed to estimate optical flow based on the coarse-to-fine manner. Then, PWC-Net [18], [33] designed an efficient pyramid network to learn the motion from coarse to fine, where cost volumes are calculated after the feature warping process at each pyramid level. Based on the PWC-Net structure, IRR-PWC [19] proposed an iterative residual refinement scheme to make the optical flow estimator shared across pyramid levels, resulting in an efficient and lightweight network architecture.

Recently, it is also proved that optical flow estimation can be improved by extracting better correspondence information. VCN [34] proposed to build 4D volumetric correspondence by volumetric encoder-decoder layers, multi-channel cost volumes and separable volumetric filters. MaskFlowNet [35] proposed an asymmetric occlusion-aware feature matching module to filter out useless areas during building the cost volume. DICL-Flow [36] proposed a displacement-invariant matching cost to decouple the 2D displacements and learn the matching costs at each 2D displacement hypothesis independently. LiteFlowNet3 [31] proposed a cost volume modulation module and a flow field deformation module which improved the performance of optical flow estimation. More recently, RAFT [12] proposed to build multi-scale 4D correlation volume for all pairs of pixels and estimate optical flow iteratively using the recurrent network, yielding state-of-the-art performance on multiple benchmarks. However, the applications of these supervised optical flow methods are limited because their performance depends heavily on the training data, which is hard to obtain in real applications.

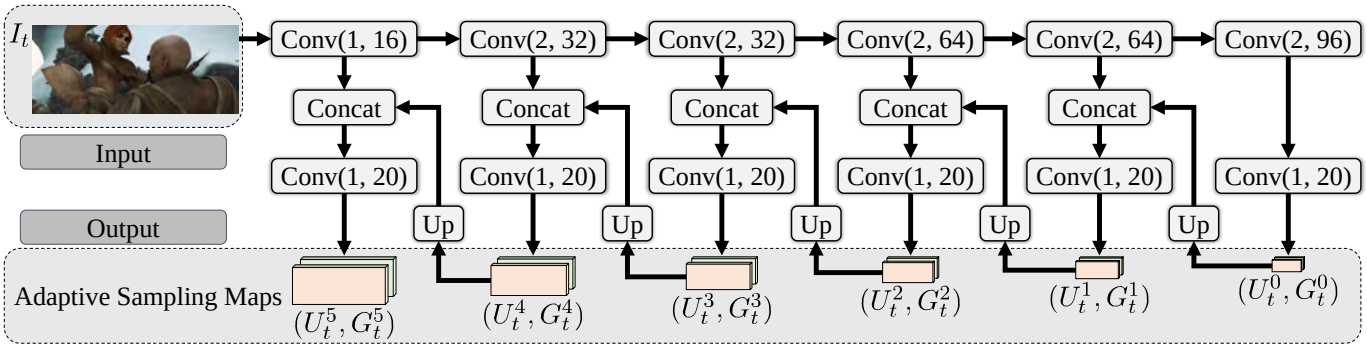


Fig. 3. The detailed structure of our Adaptive Sub-Net. I_t is the input image, U_t^i and G_t^i ($i \in \{0, 1, ..., 5\}$) are adaptive sampling maps. 'Up' denotes the bilinear upsampling operation and 'Concat' is the feature concatenate operation. 'Conv(n,m)' indicates the typical convolutional layer whose kernel size is 3×3, stride is n and output channel is m.

B. Unsupervised Deep Optical Flow

Unsupervised methods can learn optical flow networks using only image sequences. The basic formulation of unsupervised optical flow learning [37]–[39] is to train networks by minimizing the photometric loss between two input images: warping one to the another with predicted flow vectors. In this way, there is no need for ground-truth optical flow labels. However, the training becomes more difficult than supervised methods.

One core problem of unsupervised optical flow learning is the occlusion problem in the photometric loss. UnFlow [20] proposed to exclude occlusion regions from photometric loss by bidirectional flow estimation and forward-backward occlusion check. OAFlow [40] used range-map occlusion check [41] to handle occlusion. Back2Future [42] proposed to learn an occlusion estimation network for occlusion handling. DDFlow [21] and SelFlow [28] proposed to learn optical flow in occlusion regions by artificially generating occlusions during the data distillation learning process. STFlow [43] introduce a self-taught framework that used the traditional flow interpolation method [44] to improve the self-estimated flow to provide better pseudo labels for self-supervised training.

Another key point of unsupervised optical flow learning is how to learn accurate matching information from the image alignment objective. From this point of view, UnFlow [20] proposed to use census transform to make the photometric loss robust for illumination changes. OAFlow [40] proposed to calculate photometric loss by an enlarged search warping, which can facilitate the learning of large motion. Back2Future [42] proposed to improve unsupervised optical flow learning by multi-frame formulation. EpiFlow [22] introduced the low-rank constraint and the sub-space constraint into the unsupervised objective function. Stereo and depth information is also introduced to improve optical flow learning [45]–[49]. Recently, NLFlow [50] proposed a CNN-based non-local term to remove noise and blur around motion boundaries. PatchFlow [51] proposed to replace pixel-based warping with patch-based warping to improve the photometric loss. More recently, ARFlow [23] proposed the augmentation regularization loss to learn optical flow from self-supervision and augmentations. SimFlow introduced the deep feature similarity constraint to improve the photometric loss [27]. UFlow [24] proposed to learn optical flow by a unified framework that systematically analyzes different existing unsupervised components, which achieved state-of-the-art performance. Although previous methods have largely improved the performance of unsupervised optical flow learning, the downsampling and upsampling problems in the pyramid process have not been addressed. In this paper, we propose to use our content-aware pooling (CAP) module and adaptive flow upsampling (AFU) module to improve performance.

C. Image-Guided Upsampling

Our method is also related to edge-aware interpolation and upsampling, such as joint bilateral upsampling [52] and guided image filtering [53]. These methods introduced the image content information into the upsampling interpolation process. However, the image content information may be noisy, which is harmful to the upsampling results. Apart from the traditional methods, CNN-based approaches have also been proposed to extract robust guidance features or guidance filters for upsampling [54]–[56]. However, learning a CNN-based upsampling module in the unsupervised setting is difficult for previous methods, because the guidance information from unsupervised objective functions is noisy and even unreliable. In this paper, we propose to learn our AFU module in the unsupervised setting and compare it with previous components to demonstrate its effectiveness.

III. Algorithm

In this section, we first provide an overview of the network architecture of our method in Sec. III-A. Then we introduce the proposed Content-Aware Pooling (CAP) module in Sec. III-B and Adaptive Flow Upsampling (AFU) module in Sec. III-C. Finally, we describe the loss functions used for unsupervised training in Sec. III-D.

A. Network Architecture

The pipeline of the proposed network is illustrated in Fig. 2. It takes two frames I_1 and I_2 as inputs and produces an optical flow field V_1 that describes the motion of each pixel in I_1 towards I_2. The whole network contains three parts: an adaptive sub-net, a siamese feature encoder and a flow decoder.

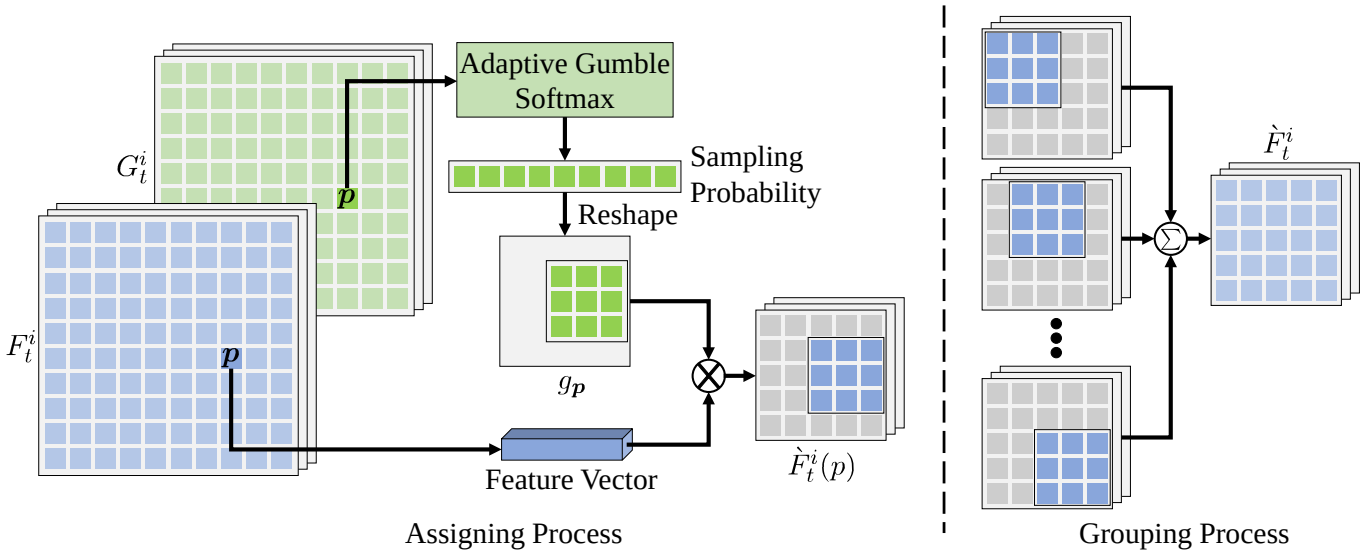


Fig. 4. Illustration of our Content-Aware Pooling module. The left is the assigning process: for each feature vector in high-resolution feature F_t^i, we assign it to the corresponding neighbor position in low-resolution feature according to its sampling probability kernel $g_{\boldsymbol{p}}$ produced by adaptive gumbel softmax and reshape operation. The right is the grouping process: we generate the low-resolution feature $\grave{F}_t^i$ by accumulating all the assigning maps $\grave{F}_t^i(\boldsymbol{p})$, as in Eq. 6.

First, we use the adaptive sub-net to extract multi-scale adaptive sampling maps, which will be used later in the CAP module and the AFU module:

$$\left\{G_1^i, G_2^i, U_1^i\right\} = \mathcal{A}(I_1, I_2), \quad i \in \{0, 1, ..., N\} \tag{1}$$

where $\mathcal{A}$ is our adaptive sub-net, i is the index of each scale and the small number represents the coarse scale, G_1^i, G_2^i and U_1^i are adaptive sampling maps. In our implementation, the adaptive sub-net is designed as a simple U-Net structure, which is shown in Fig. 3.

Second, in the siamese feature encoder, we extract multi-scale feature pairs from the input images to cover both global and local information, which is formulated as:

$$\grave{F}_t^i = \mathcal{G}(F_t^i, G_t^i), \tag{2}$$

$$F_t^{i-1} = \mathcal{C}^{i-1}(\grave{F}_t^i) \tag{3}$$

where $t \in \{1, 2\}$ is the index of the input images, $\mathcal{G}$ represents the proposed CAP module, $\grave{F}_t^i$ is the downsampled feature of F_t^i, and $\mathcal{C}^i$ is a convolution layer. Note that the architecture of the siamese feature encoder is the same as IRR-PWC [19], where 6 convolutional blocks are used to build the feature pyramid. In each convolutional block, there are only two convolutional layers: one with stride $= 2$ to downscale the feature map and another with stride $= 1$ to extract high level information.

After the feature encoding process, we estimate flow fields by the flow decoder formulated as follows:

$$\hat{V}_1^{i-1} = \mathcal{U}(V_1^{i-1}, U_1^i), \tag{4}$$

$$V_1^i = \mathcal{D}(F_1^i, F_2^i, \hat{V}_1^{i-1}), \tag{5}$$

where $\mathcal{U}$ represents our AFU module, $\hat{V}_1^{i-1}$ is the upsampled flow from $i-1$ scale and $\mathcal{E}$ is a flow estimator. Specifically, the flow estimator $\mathcal{D}$ is designed following the recent work UFlow [24], which contains feature warping, correlation layer, cost volume normalization, a dense convolution block and a dilated convolution block.

In the feature encoding process, convolution layers with stride $= 2$ are used to downscale feature maps. However, the regular downsampling method based on sliding windows may fuse features from different objects, reducing the matching accuracy of pair-wise correlation estimation. To tackle this issue, we propose a CAP module to automatically group similar features in the downsampling process, referred to as content-aware pooling. Besides, we notice that the commonly used bilinear upsampling may introduce interpolation errors and blur artifacts during the decoding process. Thus, the AFU module is proposed to ease this problem by adaptively interpolating flow fields with learnable weights. The details of these two modules are presented in Sec. III-B and Sec. III-C.

B. Content-Aware Pooling

As mentioned above, the CAP module is proposed to automatically group similar features in the pooling process. The input is a high-resolution feature map F_t^i with the size of $H \times W \times c$, and an adaptive sampling map G_t^i with the size of $H \times W \times 10$, in which 9 channels are used as sampling scores $\overline{G}_t^i$ and the remaining channel is used as the control parameter τ in our adaptive gumbel softmax. The output is a downsampled feature map $\grave{F}_t^i$ with size of $\frac{H}{r} \times \frac{W}{r} \times c$, where c denotes the channel number and r is the sampling rate, typically set to 2 in the feature encoding process.

The detailed structure of our CAP is shown in Fig. 4. We divide the CAP into two processes: the assigning process and the grouping process. In the assigning process, we first generate a sampling probability kernel $g_{\boldsymbol{p}}$ from $G_t^i(\boldsymbol{p})$ using adaptive gumbel softmax. Given a feature vector $F_t^i(\boldsymbol{p})$ at spatial position $\boldsymbol{p}$, the sampling probability kernel $g_{\boldsymbol{p}}$ indicates the probability of $F_i^t(\boldsymbol{p})$ contributing to the neighboring region of its corresponding position in the low-resolution feature

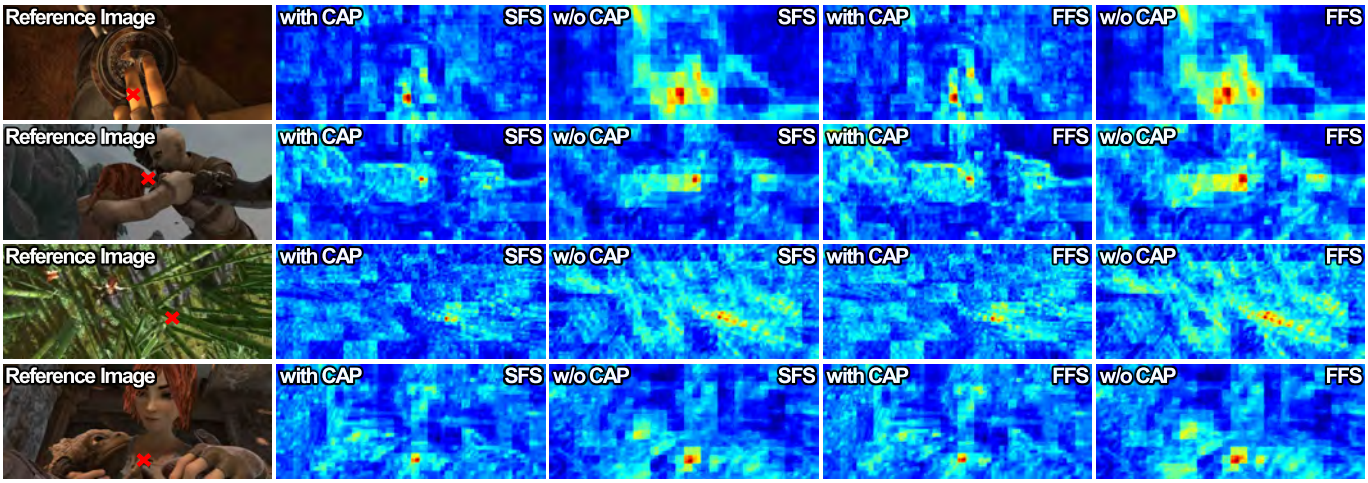


Fig. 5. The feature matching visualizations of our CAP module vs. conventional striding in convolution. We extract features from the source and the target images. We pick a feature vector from the source feature map (red cross) and compute cosine differences with other places in the source feature map (SFS), and with all features at the target feature map (FFS). More details are provided in Fig. 6. Red represents high similarity score and blue represents low similarity score. Features by SIC are likely to be similar to other places, while features by CAP are only similar to themselves.

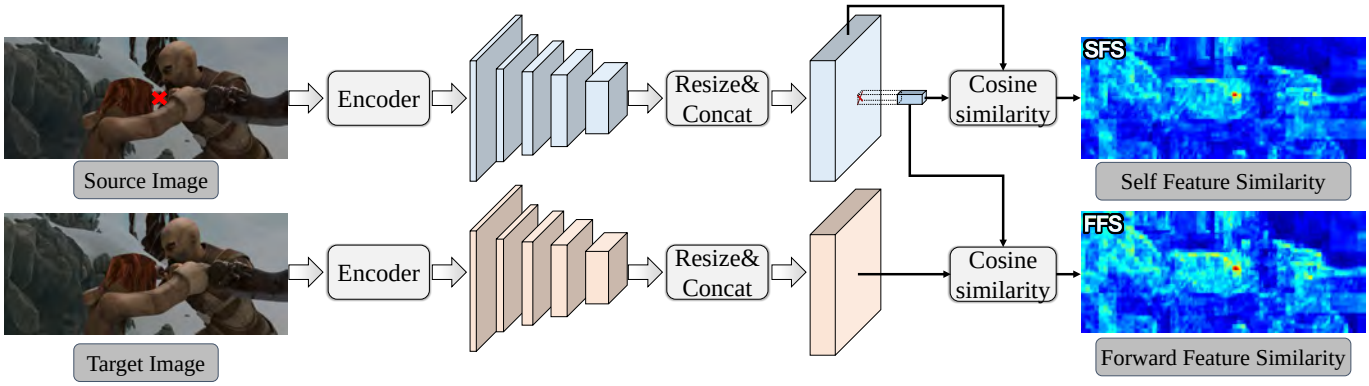


Fig. 6. Illustration of how we compute feature similarity maps for feature matching visualization. We first use the encoder to extract multi-scale feature maps from the source and target images. Then we resize the multi-scale feature maps into full size and concatenate them together. We select a feature vector in the source feature map marked by the red cross. The self feature similarity (SFS) and forward feature similarity (FFS) are calculated by computing cosine similarity between the selected feature vector and all features in the source and target feature map, respectively.

$\grave{F}_t^i$. Then we compute the assign map $\grave{F}_t^i(\boldsymbol{p})$ by a multiply operation to assign the feature vector $F_t^i(\boldsymbol{p})$ to each coordinate in low resolution according to the sampling probability kernel $g_{\boldsymbol{p}}$. Finally, in the grouping process, we generate the low-resolution feature $\grave{F}_t^i$ by accumulating all the assign maps. Thus similar feature vectors in high-resolution feature maps are grouped together according to the sampling probability kernels. In summary, our CAP module can be formulated as follows:

$$\grave{F}_t^i = \sum_{\boldsymbol{p}} F_t^i(\boldsymbol{p}) \bigotimes g_{\boldsymbol{p}}, \tag{6}$$

where $\bigotimes$ is the multiply operation with the broadcasting mechanism.

During the assigning process, in order to avoid feature grouping across different regions, we use adaptive gumbel softmax [57], [58] to suppress small probabilities when producing sampling probability kernel $g_{\boldsymbol{p}}$. In previous works, gumbel softmax is used to produce a sharp and near-differentiable mapping function with straight-through gradient estimation. Here, we follow the design in PRNet [58], where a sub-network is used to predict the temperature of the gumbel softmax correspondence. We first split the adaptive sampling map G_t^i as a sampling score $\overline{G}_t^i(j, \boldsymbol{p})$ and a control parameter $\tau(\boldsymbol{p})$ to control the distribution tendency of sampling kernels, where j is channel index and $\boldsymbol{p}$ is spatial coordinate. Note that when $\tau(\boldsymbol{p})$ gets smaller, the sampling probability will become sharper. In summary, the adaptive gumbel softmax can be formulated as follows:

$$x(j, \boldsymbol{p}) = \frac{\overline{G}_t^i(j, \boldsymbol{p})}{\text{sigmoid}(\tau(\boldsymbol{p})) + \rho}, \tag{7}$$

$$g_{\boldsymbol{p}}(j) = \frac{exp(x(j, \boldsymbol{p}))}{\sum_k^9 exp(x(k, \boldsymbol{p}))}, \tag{8}$$

where ρ is a constant to avoid zero denominator and $x(j, \boldsymbol{p})$ is the transformed sampling score.

Fig. 5 provides some visualizations of content-aware pooling results by comparing our CAP module with conventional striding in convolution (SIC). We train two networks with

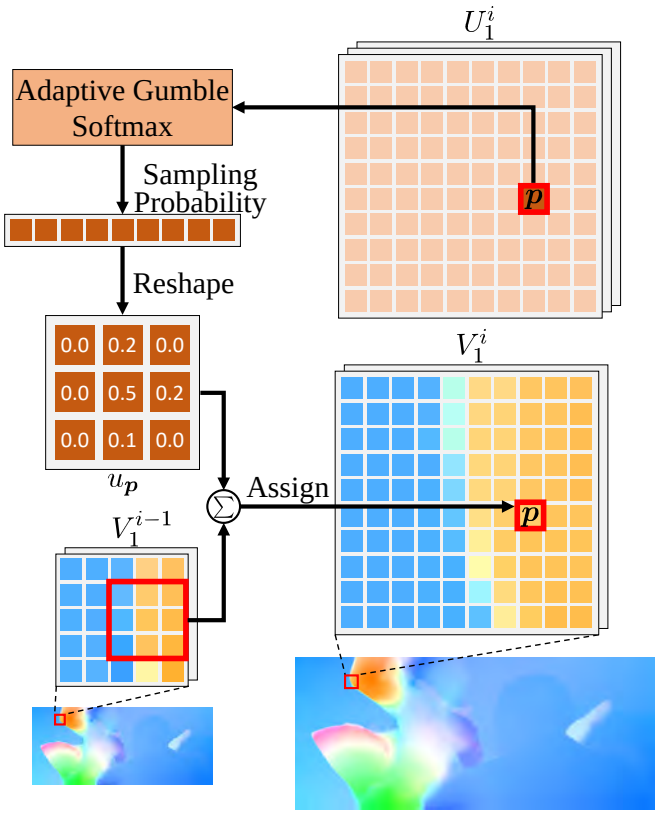


Fig. 7. Illustration of our Adaptive Flow Upsampling module. The flow vector in high-resolution flow field $V_1^i(\boldsymbol{p})$ is generated by sampling and fusion according to its sampling kernel $u_{\boldsymbol{p}}$.

different encoder structures in the same training setting. Then we visualize the feature similarity results as illustrated in Fig. 6. We first interpolate pyramid features into the image size and concatenate them together. Then feature vector in I_1 located by the red cross is selected to calculate cosine similarity with features of I_1 and I_2, which is the self feature similarity (SFS) map and the forward feature similarity (FFS) map, respectively. The SFS map reveals the discriminative ability of the encoded features and the FFS map reveals the matching ability between feature pairs. From Fig. 5, we can see that features extracted by SIC method are likely to be similar with neighbor objects, which may produce ambiguity matching information for optical flow estimation. Unlike the SIC method, our CAP module can generate features that are only similar to their corresponding feature vectors.

C. Adaptive Flow Upsampling

The conventional bilinear upsampling method may interpolate flow vectors across object boundaries leading to blur artifacts and errors during the flow decoding process. To solve this problem, we design an adaptive flow upsampling module to adaptively interpolate flow fields with learnable weights. Our motivation is to upsample the flow field using learnable interpolation weights. In order to avoid interpolation blur that causes by fusion across edges, we use adaptive gumbel softmax to make the interpolation weights close to a one-hot distribution in edge areas. The detail of our AFU module is shown in Fig. 7.

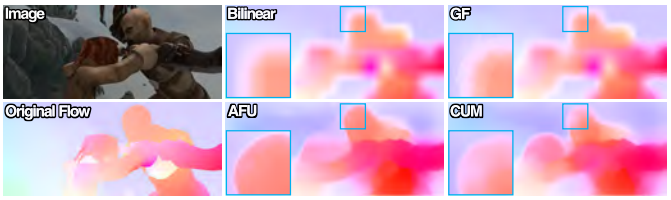


Fig. 8. An example of using our AFU module to upsample optical flow in the pyramid network. We first downsample the ground-truth optical flow (original flow) by 64 times, which is the smallest scale of our pyramid network. Then we upsample the downscaled flow to the original size using our AFU, bilinear method, guided filter [53] (GF) and the learned convex upsample module [12] (CUM). Compared with previous methods, our AFU can preserve more object structure in the upsampling process.

Fig. 9. An example of using our AFU module to upsample images. We first downsample the original image by 64 times, which is the smallest scale of our pyramid network. Then we iteratively upsample the downscaled image to the original size using our AFU, as Eq. 11. Compared with bilinear upsampling, our AFU upsampling can preserve detail object structures.

Given a low-resolution flow field V_1^{i-1} of size $\frac{H}{r} \times \frac{W}{r} \times 2$ and a high-resolution adaptive sampling map U_1^i with size of $H \times W \times 10$, our goal is to produce a high-resolution flow field V_1^i with size of $H \times W \times 2$. We define $\boldsymbol{p}$ as a spatial coordinate in V_1^i and $\boldsymbol{q} \in \mathcal{N}(\boldsymbol{p}/r)$ as its corresponding neighbors in V_1^{i-1}. The flow vectors in high-resolution flow field V_1^i can be calculated by the following formulation (the '$\sum$' and 'Assign' operation in Fig. 7):

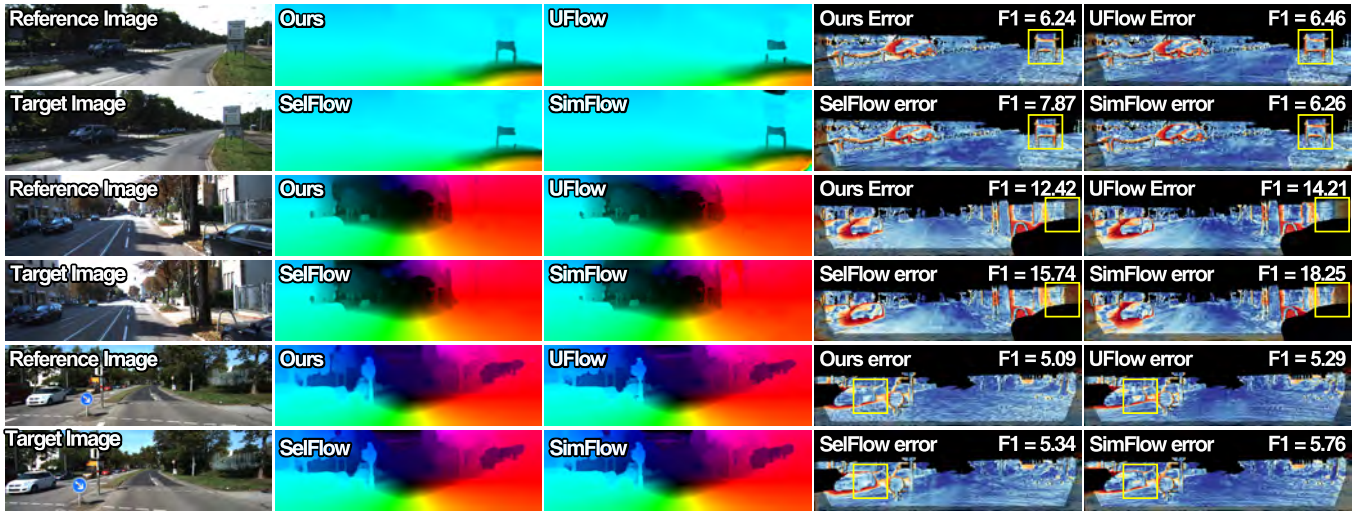
$$V_1^i(\boldsymbol{p}) = \sum_{\boldsymbol{q} \in \mathcal{N}(\boldsymbol{p}/r)} u_{\boldsymbol{p}}(\boldsymbol{q}) V_1^{i-1}(\boldsymbol{q}), \tag{9}$$

where $u_{\boldsymbol{p}}(\boldsymbol{q})$ is a sampling probability kernel to indicate the contribution probability of $V_1^{i-1}(\boldsymbol{q})$ to $V_1^i(\boldsymbol{p})$. Here the sampling probability kernel $u_{\boldsymbol{p}}(\boldsymbol{q})$ is generated from U_1^i using adaptive gumbel softmax similar to Eq. 7 and Eq. 8, where small probabilities are compressed to zeros. Thus the flow vectors in the high-resolution flow field are generated by adaptively fusing flow vectors in low-resolution flow field based on sampling probability kernels. An example of using our AFU to upsample optical flow is shown in Fig. 8, where optical flow is first downscaled by 64 times as the smallest scale in the pyramid network. As can be seen, our AFU can preserve object structure during the pyramid upsampling process.

D. Unsupervised Losses

In order to train our network in the unsupervised setting where ground-truth labels are not available, we use a set of unsupervised losses as our training objective. Our main

(a) Qualitative visualization comparison on KITTI 2012 benchmark.



(b) Qualitative visualization comparison on KITTI 2015 benchmark.

Fig. 10. We show qualitative comparison with the previous methods UFlow [24], SimFlow [27], SelFlow [28] and DDFlow [21] on online evaluation benchmarks, including KITTI 2012 (a) and KITTI 2015 (b). The error maps of the predictions are visualized in the last two columns. In the error maps, brighter regions indicate the larger estimation errors except that visualized by KITTI 2015 benchmark where correct estimations are displayed in blue and wrong ones in red.

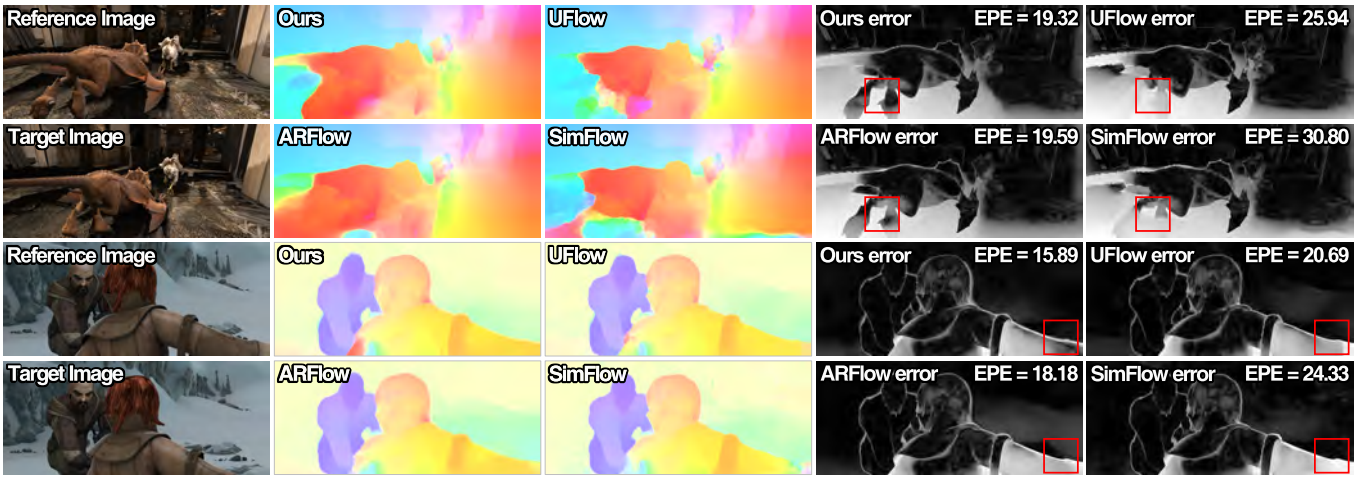
objective is the photometric loss $\mathcal{L}_d$, which is designed based on the brightness constancy assumption that the object appearance should be invariable in input frames. However, occlusion regions caused by moving objects can not be optimized by the photometric loss. We explicitly exclude these regions in the photometric loss by forward-backward consistency checking [20]. As a result, the photometric loss $\mathcal{L}_d$ is formulated as follows:

$$\mathcal{L}_d = \frac{\sum_{\boldsymbol{p}} \Psi\Big(I_1(\boldsymbol{p}) - I_2\big(\boldsymbol{p} + V_1(\boldsymbol{p})\big)\Big) \cdot O_1(\boldsymbol{p})}{\sum_{\boldsymbol{p}} O_1(\boldsymbol{p})}, \qquad (10)$$

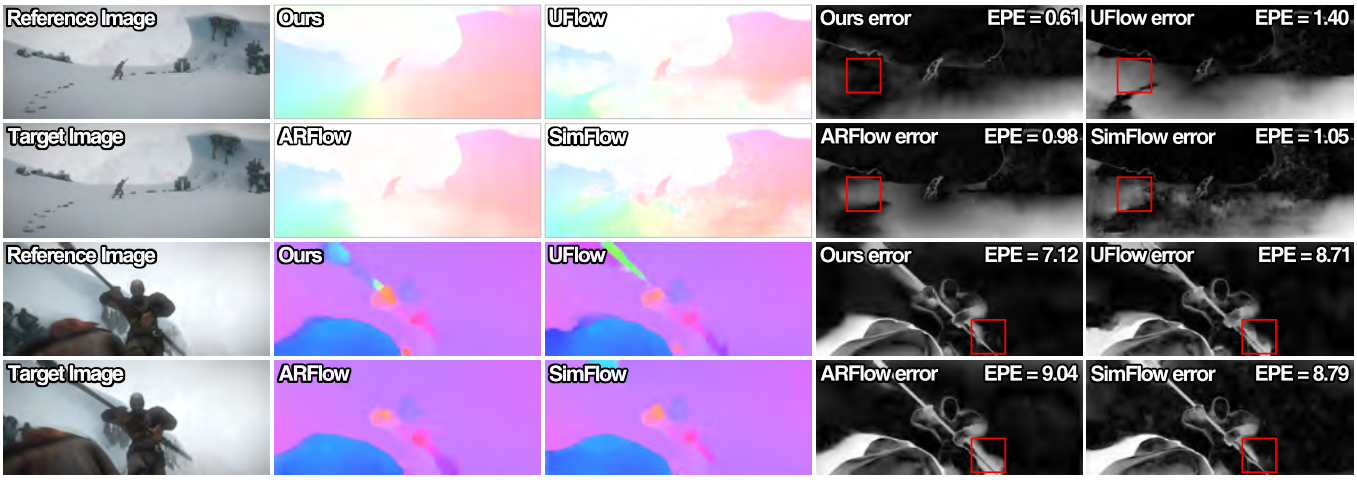
where O_1 is the occlusion mask generated by forward-backward consistency checking. '1' indicates the non-occluded pixel and '0' means the occluded pixel. Ψ is the robust penalty function [21]: $\Psi(x) = (|x| + \epsilon)^q$ in which q and ϵ are set to 0.4 and 0.01.

Following previous works, several loss functions are used to train our model, including the edge-aware smooth loss $\mathcal{L}_s$ that improves the smoothness of output flow field [40], the census loss $\mathcal{L}_c$ that increases the robustness under illumination changes [20], the boundary dilated warping loss $\mathcal{L}_b$ to learn motions towards outside the image plane [26], the augmentation regularization loss $\mathcal{L}_a$ that introduces the equivariance constrain to encourage the robustness to variations [23].

In order to ensure the upsampled flow fields better fit object boundaries, we design a sampling regularization loss $\mathcal{L}_r$ to constrain the learned adaptive sampling maps $\{U_1^i\}$. We first downscale the input image I_1 to I_1^0, whose size is the same as V_1^0. Then we iteratively upsample the downscaled image and compute a reconstruction loss with the original image, which



(a) Qualitative visualization comparison on Sintel Clean benchmark.



(b) Qualitative visualization comparison on Sintel Final benchmark.

Fig. 11. We show qualitative comparison with the previous methods UFlow [24], ARFlow [23] and SimFlow [27] on online evaluation benchmarks, including Sintel Clean (a) and Sintel FInal (b). The last two columns visualize the error maps, where brighter regions indicate the larger estimation errors.

is formulated as follows:

$$I_1^i = \mathcal{U}(I_1^{i-1}, U_1^i), \tag{11}$$

$$\mathcal{L}_r = \sum_{\boldsymbol{p}} \Psi(I_1(\boldsymbol{p}) - I_1^N(\boldsymbol{p})), \tag{12}$$

where $\mathcal{U}$ is our AFU module, I_1^N is the reconstructed image by the iterative upsampling process described in Eq. 11. This means that we use the same parameters to upsample image as in upsampling optical flow. An example of using AFU module to upsample image is shown in Fig. 9. We use our AFU module and bilinear method to upsample the low-resolution image ($\frac{1}{64}$ scale), respectively. As can be seen, our learned AFU module can preserve detailed object structures in the image upsampling process and it can be further applied to the flow upsampling process by adding our sampling regularization loss.

Eventually, our loss function is a weighted combination of above individual loss terms:

$$\mathcal{L} = \mathcal{L}_d + \lambda_b \mathcal{L}_b + \lambda_s \mathcal{L}_s + \lambda_c \mathcal{L}_c + \lambda_a \mathcal{L}_a + \lambda_r \mathcal{L}_r, \tag{13}$$

where λ_b, λ_s, λ_c, λ_a and λ_r are hyper-parameters, set to $\lambda_b =$, $\lambda_s = 0.05$, $\lambda_c = 1$, $\lambda_a = 0.5$, $\lambda_r = 0.1$ in our experiments.

IV. Experimental Results

A. Datasets and Implementation Details

We conduct comprehensive experiments on three widely-used optical flow benchmarks, including MPI-Sintel [25], KITTI 2012 [29], and KITTI 2015 [30]. MPI-Sintel contains 1,041 training image pairs extracted from the rendered open-source movie, divided into 'Clean' and 'Final' passes. Following previous works [23], [24], [27], we use both versions of rendering images to train our model. KITTI 2012 and KITTI 2015 are real-world datasets collected in driving conditions. There are 194 training pairs and 195 test pairs in KITTI 2012, and 200 training pairs and 200 test pairs in KITTI 2015. The two datasets also provide their multi-view extension datasets, which are video sequences without optical flow labels. We train our model on the multi-view extension datasets and evaluation on the train sets of KITTI 2012 and KITTI 2015. Results of the test set are uploaded to the KITTI website for benchmark comparison.

The implementation of the proposed ASFlow is based on the PyTorch toolbox. We train our model on 2 NVIDIA GeForce

TABLE I

QUANTITATIVE COMPARISON WITH STATE-OF-THE-ART METHODS ON FOUR WIDELY-USED DATASETS USING EPE AND F1-MEASURE METRICS (THE LOWER THE BETTER). FOLLOWING PREVIOUS WORKS [23], [24], [27], '−' MEANS THE RESULT IS NOT REPORTED IN THE PAPER, '( )' INDICATES IMAGES FROM TEST SET ARE USED DURING UNSUPERVISED TRAINING, AND '+FT' MEANS THE SUPERVISED METHODS USE IMAGES OF TARGET DOMAIN FOR TRAINING, OTHERWISE USING SYNTHETIC DATA LIKE FLYING CHAIRS [15] AND FLYING CHAIRS OCC [19]. THE BEST UNSUPERVISED METHOD IS MARKED IN **BOLD** AND THE SECOND BEST IS MARKED IN BLUE FOR BETTER COMPARISON.

	Method	KITTI 2012		KITTI 2015		Sintel Clean		Sintel Final	
		train	test	train	test (F1-all)	train	test	train	test
Supervised	FlowNetS [15]	8.26	–	–	–	4.50	7.42	5.45	8.43
	FlowNetS+ft [15]	7.52	9.1	–	–	(3.66)	6.96	(4.44)	7.76
	SpyNet [16]	9.12	–	–	–	4.12	6.69	5.57	8.43
	SpyNet+ft [16]	8.25	10.1	–	35.07%	(3.17)	6.64	(4.32)	8.36
	LiteFlowNet [17]	4.25	–	10.46	–	2.52	–	4.05	–
	LiteFlowNet+ft [17]	(1.26)	1.7	(2.16)	10.24%	(1.64)	4.86	(2.23)	6.09
	PWC-Net [18]	4.57	–	13.20	–	3.33	–	4.59	–
	PWC-Net+ft [18]	(1.45)	1.7	(2.16)	9.60%	(1.70)	3.86	(2.21)	5.13
	IRR-PWC+ft [19]	–	–	(1.63)	7.65%	(1.92)	3.84	(2.51)	4.58
	RAFT [12]	–	–	5.54	–	1.63	–	2.83	–
	RAFT-ft [12]	–	–	–	6.30%	–	2.42	–	3.39
Unsupervised	BackToBasic [39]	11.30	9.9	–	–	–	–	–	–
	DSTFlow [38]	10.43	12.4	16.79	39%	(6.16)	10.41	(6.81)	11.27
	UnFlow [20]	3.29	–	8.10	23.3%	–	9.38	(7.91)	10.22
	OAFlow [40]	3.55	4.2	8.88	31.2%	(4.03)	7.95	(5.95)	9.15
	Back2Future [42]	–	–	6.59	22.94%	(3.89)	7.23	(5.52)	8.81
	NLFlow [50]	3.02	4.5	6.05	22.75%	(2.58)	7.12	(3.85)	8.51
	DDFlow [21]	2.35	3.0	5.72	14.29%	(2.92)	6.18	(3.98)	7.40
	EpiFlow [22]	(2.51)	3.4	(5.55)	16.95%	(3.54)	7.00	(4.99)	8.51
	SelFlow [28]	1.69	2.2	4.84	14.19%	(2.88)	6.56	(3.87)	6.57
	STFlow [50]	1.64	1.9	3.56	13.83%	(2.91)	6.12	(3.59)	6.63
	ARFlow [23]	1.44	1.8	2.85	11.80%	(2.79)	4.78	(3.87)	5.89
	SimFlow [27]	–	–	5.19	13.38%	(2.86)	5.92	(3.57)	6.92
	UFlow [24]	1.68	1.9	2.71	11.13%	(2.50)	5.21	(3.39)	6.50
	ASFlow(ours)	**1.26**	**1.5**	**2.47**	**9.67%**	**(2.40)**	**4.56**	**(2.89)**	**5.86**

TABLE II

ABLATION FOR UNSUPERVISED COMPONENTS. CL: CENSUS LOSS [20], BDWL: BOUNDARY DILATED WARPING LOSS [26], ARL: AUGMENTATION REGULARIZATION LOSS [23], SGU: SELF-GUIDED UPSAMPLING, PDL: PYRAMID DISTILLATION LOSS. THE BEST RESULTS ARE MARKED IN **BOLD**.

CL	BDWL	ARL	CAP	AFU	KITTI 2012			KITTI 2015			Sintel Clean			Sintel Final		
					ALL	NOC	OCC	ALL	NOC	OCC	ALL	NOC	OCC	ALL	NOC	OCC
✓					4.52	1.76	19.63	7.58	2.46	30.43	(3.52)	(1.87)	(12.9)	(4.19)	(2.59)	(13.64)
✓	✓				3.39	1.09	16.58	6.89	2.20	28.12	(3.41)	(1.62)	(13.5)	(3.85)	(2.17)	(13.71)
✓	✓	✓			1.42	0.91	4.39	3.00	2.12	6.89	(2.84)	(1.50)	(10.6)	(3.60)	(2.28)	(11.52)
✓	✓	✓	✓		1.37	0.93	3.98	2.64	1.96	6.01	(2.61)	(1.33)	(10.1)	(3.17)	(1.92)	(10.70)
✓	✓	✓	✓		1.29	0.89	3.78	2.53	1.98	5.16	(2.51)	(1.27)	(9.79)	(2.98)	(1.79)	(9.98)
✓	✓	✓		✓	1.30	0.88	3.82	2.57	1.99	5.08	(2.46)	(1.23)	(9.63)	(2.94)	(1.73)	(10.07)
✓	✓	✓	✓	✓	**1.26**	**0.87**	**3.72**	**2.47**	**1.93**	**5.02**	**(2.40)**	**(1.20)**	**(9.36)**	**(2.89)**	**(1.71)**	**(9.89)**

GTX 2080Ti GPUs for about 1000k iterations. For better generalization, we follow previous work [23] to use basic data augmentation strategies like random crop and horizontal flip for training. We use different crop sizes for the Sintel and KITTI datasets. For Sintel dataset, the image size is 436×1024 and we set the crop size as 320×768. For KITTI datasets, the original image size is around 376×1240, we crop patches of size 320×896 for training. During the training process, the training batch size is 2 and the learning rate is 10^{-4}.

B. Comparison with State-of-the-Art

In Tab. I, We compare our method with State-of-the-Art (SOTA) works, including both supervised and unsupervised methods, on four widely-used benchmarks. The standard evaluation metrics, i.e., average endpoint error (EPE) and the percentage of erroneous pixels (F1-measure), are used to evaluate the performance of the predicted optical flow. The best unsupervised method is marked in **bold** and the second best is marked in blue for better comparison.

1) Comparison with Unsupervised Methods.: As shown in Tab. I, our ASFlow consistently achieves better performance than other methods on four standard benchmarks. Specifically, our method achieves an EPE error of 1.5 on the KITTI 2012 test set, which surpasses previous top-ranked methods UFlow [24] and ARFLow [23] by around 21.1% (1.9 → 1.5) and 16.7% (1.8 → 1.5), respectively. For KITTI 2015 online evaluation, our

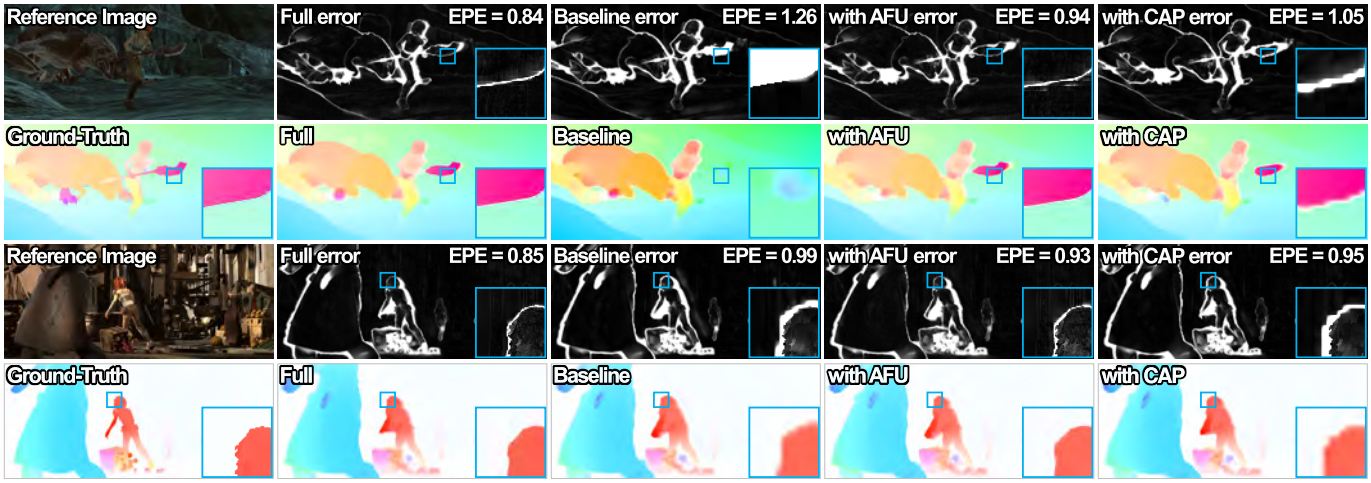


Fig. 12. Qualitative visualizations of the proposed method on Sintel Clean. The room-in flow results and error maps are shown in the right corner of each sample.

TABLE III
COMPARISON OF OUR AFU WITH CLASSICAL UPSAMPLING METHODS, SUCH AS JBU [52] AND GF [53], AND DEEP-BASED UPSAMPLING METHODS, SUCH AS DJF [59], DGF [55], PAC [56], AND CUM [12]. AFU-NA MEANS THAT THE CONTROL PARAMETER $\tau(\boldsymbol{p})$ IS FIXED WITH 0.25 IN THE ADAPTIVE GUMBEL SOFTMAX BLOCK. AFU-NRL DENOTES THE SAMPLING REGULARIZATION LOSS IS DISABLED.

Method	KITTI 2012	KITTI 2015	Sintel Clean	Sintel Final
Bilinear	1.29	2.53	(2.51)	(2.98)
JBU [52]	1.51	3.00	(2.66)	(2.98)
GF [53]	1.40	2.90	(2.72)	(2.92)
DJF [59]	1.36	2.79	(2.75)	(3.20)
DGF [55]	1.41	3.14	(2.69)	(3.05)
PAC [56]	1.42	2.65	(2.58)	(2.95)
CUM [12]	1.30	2.60	(2.51)	(2.93)
AFU-NA	1.29	2.57	(2.44)	(2.91)
AFU-NRL	1.28	2.52	(2.45)	(2.90)
AFU	**1.26**	**2.47**	**(2.40)**	**(2.89)**

method set new records of 2.47 in EPE on the training set and 9.67% in F1-measure, which outperforms previous methods by a large margin. On the most challenging dataset MPI-Sintel, our method achieves EPE error of 4.56 on 'Clean' pass for online testing. It obtains EPE = 5.86 on 'Final' pass, outperforming previous top methods SimFlow [27] and UFlow [24] by 1.06 and 0.64 in terms of EPE. It is worth noting that our method is the first one to achieve the best results on all benchmarks, as shown in each line of Tab. I (best viewed in colors).

Fig. 10 and Fig. 11 provide some qualitative comparisons with the previous methods such as UFlow [24], SimFlow [27], ARFlow [23], SelFlow [28] and DDFlow [21]. As can be seen, our method is clearly able to make accurate and smooth predictions, especially when handling the tough regions around the foreground boundaries.

2) Comparison with Supervised Methods.: We also report the results of representative supervised methods for comprehensive comparison, see Tab. I. For cross-domain evaluation, we consider the ground-truth optical flow is not available for training. Thus, the supervised models are trained on synthetic data such as Flying Chairs [15] and Flying Chairs occ [19], while the training procedure of the unsupervised methods can be directly performed only using target domain images. As can be seen, our method achieves better performance than all the supervised methods. Especially in real scenarios like KITTI 2015 dataset, it significantly outperforms the well-known supervised methods like LiteFlowNet [17], PWC-Net [18] and RAFT [12] by a large margin (7.99, 10.73 and 3.07 in EPE, respectively).

As for in-domain evaluation, our method generally achieves competitive performance with the supervised methods. Specially, on KITTI 2012 and 2015 datasets, our method achieves 1.5 in EPE and 9.67% in F1-measure, which surprisingly exceed the recent supervised methods like and LiteFlowNet [17].

C. Ablation Study

In this section, we conduct a series of ablation experiments to evaluate each component in the proposed network. Following [27], [28], we train our model on train sets of KITTI and MPI-Sintel. The EPE error over all pixels (ALL), non-occluded pixels (NOC) and occluded pixels (OCC) are reported for quantitative comparisons.

1) Unsupervised Components.: Following the success of prior works [20], [26], we employ some effective components to boost the training of our model in an unsupervised manner. As shown in the first line of Tab. II, we first train a baseline model using photometric loss and smooth loss without the proposed modules. After adding census loss [20] (CL), boundary dilated warping loss [26] (BDWL) and augmentation regularization loss (ARL), it obtains consistent improvements by three metrics on all datasets, which demonstrates these three modules' benefit to boosting a better prediction. Meanwhile, the performance of this model (CL + BDWL + ARL) is equivalent to that reported in the previous best method UFlow [24].

Note that the components we used here are compatible with each other because they are designed to tackle different problems. For example, census loss (CL) is to increase the

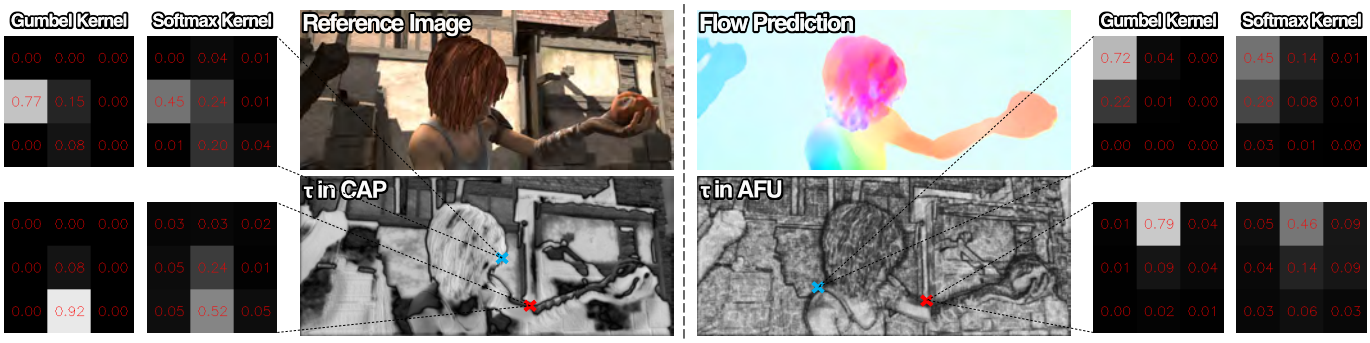


Fig. 13. Illustration of the learned sampling kernels in our CAP and AFU. Left: the input reference image, the learned control parameter $\tau(\boldsymbol{p})$ in CAP, softmax kernels and gumbel softmax kernels. Right: the prediction flow, the control parameter in AFU, softmax kernels and gumbel softmax kernels. The learned control parameter $\tau(\boldsymbol{p})$ is visualized by the gray image, where the darker area represents the smaller parameter. The kernels in edge regions (localized by the blue and red crosses) are also visualized, where the red numbers are the probabilities in the kernel and the brightness of the grid corresponds to its probability.

TABLE IV
COMPARISON OF OUR CAP WITH DIFFERENT FEATURE POOLING METHODS: AVERAGE POOLING (AVE), MAX POOLING (MAX), AND STRIDING IN CONVOLUTION (SIC).

Method	KITTI 2012	KITTI 2015	Sintel Clean	Sintel Final
Bilinear	1.51	2.81	(2.75)	(3.20)
AVE	1.39	2.75	(2.66)	(2.98)
MAX	1.40	2.69	(2.72)	(3.02)
SIC	1.30	2.57	(2.46)	(2.94)
CAP	**1.26**	**2.47**	**(2.40)**	**(2.89)**

robustness of illumination changes. Boundary dilated warping loss (BDWL) is designed to learn motions towards outside the image plane, which is crucial to improve the performance in near image boundary regions for datasets with large camera motions, e.g., KITTI dataset. The augmentation regularization loss (ARL) can introduce the equivariance constrain to encourage the robustness to variations. The proposed pyramid sampling method CAP and AFU can increase the feature representation ability and improve the quality of flow upsampling. Thus replacing the original striding strategy with our CAP in each stage of the encoder network can greatly improve the performance. Similarly, we append our AFU module on decoders and observe that the three metrics are clearly reduced (the lower is better). Finally, we fully equip the model with both CAP and AFU, which brings about 10% performance improvement.

2) Ablation for Upsampling Modules.: There have been several works that attempt to propose general upsampling operations based on image information, such as JBU [52], GF [53], DJF [59], DGF [55] and PAC [56]. Recently, RAFT [12] also introduced a convex upsample module (CUM) to improve motion boundaries in the upsampling process. However, these methods are not suitable for this challenging task. Here we propose a task-specific upsampling strategy to better serve the need of optical flow upsampling. To verify the effect of our method, we carry out extensive comparisons with the upsampling methods. Specifically, we build a simple pyramid network with the same loss functions and repetitively change upsampling operations with the modules mentioned above for fair comparison. As we can see in Tab. III, our AFU achieves the best performance overall the competitors. This is because AFU can adaptively interpolate flow fields with learnable weights in pyramid decoders, so that the blur artifacts caused by cross-edge interpolation can be avoided, see column 4 of Fig. 12.

We also report an empirical study of the adaptive gumbel softmax (row 8) and the sampling regularization loss (row 9) in Tab. III. AFU-NA is the non-adaptive softmax method, in which the control parameter $\tau(\boldsymbol{p})$ in the adaptive gumbel softmax block is empirically fixed with 0.25 following RAFT [12]. AFU-NRL means that the sampling regularization loss is disabled. Comparing AFU-NA and AFU, we can see that the content-aware ability of the adaptive gumbel softmax method can improve the performance, e.g., In KITTI 2015 dataset, the adaptive gumbel softmax method reduces the EPE error from 2.57 to 2.47. The reason is that during the sampling process, a smooth kernel is needed to capture context information in flatten regions and a sharp kernel is needed to preserve boundary information in edge regions. Thus the proposed adaptive method can produce better results than the non-adaptive method. Moreover. Comparing AFU-NRL and AFU, we can see that the performance can be further improved by using the sampling regularization loss. In Sintel Clean dataset, the sampling regularization loss improves the EPE error from 2.45 to 2.40, which demonstrates the effectiveness of ensuring the upsampled flow fields fit object boundaries. However, in Sintel Final dataset, the improvement is relatively small. The reason is that the Sintel Final dataset contains a lot of motion blur and noise, which makes it difficult to learn to compute suitable sampling kernels in object boundary regions.

3) Ablation for Feature Pooling Strategies.: Tab. IV reports the comparison of our CAP with typical pooling strategies, including average pooling (AVE), max pooling (MAX), and striding in convolution (SIC). For a fair comparison, all the experiments are conducted under the same setting. As we can see, our CAP consistently obtains better scores than others on four datasets. As mentioned in Sec. III-B, the features are adaptively grouped based on content and appearance similarity, which helps the network to maintain spatial details of different objects. Experimental results demonstrate the obtained

TABLE V
QUANTITATIVE EVALUATION OF THE PROPOSED PYRAMID SAMPLING METHOD IN SUPERVISED SETTING. THE NETWORKS ARE TRAINED ON FLYINGCHAIRS DATASET AND EVALUATED ON THE TEST SET OF FLYINGCHAIRS DATASET AND THE TRAIN SETS OF SINTEL DATASET.

method	Chairs	Sintel Clean			Sintel Final		
	ALL	ALL	NOC	OCC	ALL	NOC	OCC
IRR-PWC [19]	2.08	2.80	–	–	4.13	–	–
baseline	2.09	2.72	1.63	8.50	4.05	2.93	10.27
baseline+CAP	2.04	2.62	1.56	8.50	4.03	2.91	10.18
baseline+AFU	2.01	2.60	1.53	8.52	3.98	2.87	10.19
ours	**1.96**	**2.54**	**1.50**	**8.29**	**3.92**	**2.79**	**10.09**

distinctive information is crucial for recovering the optical flow on thin stuffs, as shown in Fig. 12 (first sample, column 3 and 5).

For a better understanding of the proposed CAP and AFU modules, we also show the learned sampling kernels and control parameter $\tau(\boldsymbol{p})$ in our adaptive gumbel softmax blocks in CAP and AFU, respectively. Fig. 13 visualizes these results. As can be seen, the control parameter $\tau(\boldsymbol{p})$ tends to be smaller in object edge regions than in flat regions, which forces the sampling kernels in edge regions to be close to a discrete one-hot distribution. Therefore, adaptive gumbel softmax is effective in preventing the phenomenon of crossing edge sampling.

4) Ablation in supervised setting. : To further demonstrate the effectiveness of the proposed pyramid sampling method, we conduct a set of experiments in supervised setting. Following IRR-PWC [19], we use the train set of Flying Chairs dataset to train networks and use the test set of Flying Chairs dataset and the train set of Sintel dataset for evaluation. Results are shown in Table V. The first and second lines are the performance of our baseline model reported by IRR-PWC [19] and by our implementation, respectively. As can be seen from Table V, both the proposed CAP module and AFU module can improve the performance, which demonstrates that improving the feature representation ability and flow upsampling quality is also useful in supervised learning settings.

V. CONCLUSION

We have presented ASFlow, an adaptive pyramid sampling method for unsupervised optical flow estimation. Two modules have been proposed, the content-aware pooling (CAP) for the pyramid downsampling and the adaptive flow upsampling (AFU) for the upsampling. Specifically, the CAP module can assemble similar features together to improve the capability of the multi-scale feature pyramid. The AFU module can adaptively interpolate flow vectors without crossing edges, resulting in sharper motion boundaries. We compared our method with previous representative optical flow methods on several leading benchmarks. In the further, we will explore the proposed two modules in the other applications, especially the CAP for the high-level vision tasks.

REFERENCES

[1] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[2] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *Proc. CVPR*, pp. 9000–9008, 2018.

[3] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger, "Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios?," in *Proc. ICCV*, pp. 2593–2602, 2017.

[4] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. NeurIPS*, pp. 568–576, 2014.

[5] J. Zhang, J. Pan, D. Wang, S. Zhou, X. Wei, F. Zhao, J. Liu, and J. Ren, "Deep dynamic scene deblurring from optical flow," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[6] Y. Zhou, X. Xu, F. Shen, X. Zhu, and H. T. Shen, "Flow-edge guided unsupervised video object segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[7] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *Proc. CVPR*, pp. 2432–2439, 2010.

[8] D. Sun, E. B. Sudderth, and M. J. Black, "Layered image motion with explicit occlusions, temporal consistency and depth ordering," in *Proc. NeurIPS*, pp. 2226–2234, 2010.

[9] M. A. Mohamed, H. A. Rashwan, B. Mertsching, M. A. García, and D. Puig, "Illumination-robust optical flow using a local directional pattern," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1499–1508, 2014.

[10] D. Sun, C. Liu, and H. Pfister, "Local layering for joint motion estimation and occlusion detection," in *Proc. CVPR*, pp. 1098–1105, 2014.

[11] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black, "Optical flow with semantic segmentation and localized layers," in *Proc. CVPR*, pp. 3889–3898, 2016.

[12] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Proc. ECCV*, pp. 402–419, 2020.

[13] M. Zhai, X. Xiang, R. Zhang, N. Lv, and A. El Saddik, "Optical flow estimation using dual self-attention pyramid networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 10, pp. 3663–3674, 2020.

[14] J. Chen, J. Lai, Z. Cai, X. Xie, and Z. Pan, "Optical flow estimation based on the frequency-domain regularization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 1, pp. 217–230, 2021.

[15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proc. ICCV*, pp. 2758–2766, 2015.

[16] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proc. CVPR*, pp. 2720–2729, 2017.

[17] T.-W. Hui, X. Tang, and C. C. Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *Proc. CVPR*, pp. 8981–8989, 2018.

[18] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proc. CVPR*, pp. 8934–8943, 2018.

[19] J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," in *Proc. CVPR*, pp. 5747–5756, 2019.

[20] S. Meister, J. Hur, and S. Roth, "Unflow: Unsupervised learning of optical flow with a bidirectional census loss," in *Proc. AAAI*, pp. 7251–7259, 2018.

[21] P. Liu, I. King, M. Lyu, and J. Xu, "Ddflow:learning optical flow with unlabeled data distillation," in *Proc. AAAI*, pp. 8770–8777, 2019.

[22] Y. Zhong, P. Ji, J. Wang, Y. Dai, and H. Li, "Unsupervised deep epipolar flow for stationary or dynamic scenes," in *Proc. CVPR*, pp. 12095–12104, 2019.

[23] L. Liu, J. Zhang, R. He, Y. Liu, Y. Wang, Y. Tai, D. Luo, C. Wang, J. Li, and F. Huang, "Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation," in *Proc. CVPR*, pp. 6489–6498, 2020.

[24] R. Jonschkowski, A. Stone, J. T. Barron, A. Gordon, K. Konolige, and A. Angelova, "What matters in unsupervised optical flow," in *Proc. ECCV*, pp. 557–572, 2020.

[25] D. Butler, J. Wulff, G. Stanley, and M. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. ECCV*, pp. 611–625, 2012.

[26] S. Liu, K. Luo, N. Ye, C. Wang, J. Wang, and B. Zeng, "Oiflow: Occlusion-inpainting optical flow estimation by unsupervised learning," *IEEE Trans. on Image Processing*, vol. 30, pp. 6420–6433, 2021.

[27] W. Im, T.-K. Kim, and S.-E. Yoon, "Unsupervised learning of optical flow with deep feature similarity," in *Proc. ECCV*, pp. 172–188, 2020.

[28] P. Liu, M. Lyu, I. King, and J. Xu, "Selflow:self-supervised learning of optical flow," in *Proc. CVPR*, pp. 4571–4580, 2019.

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. CVPR*, pp. 3354–3361, 2012.

[30] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. CVPR*, pp. 3061–3070, 2015.

[31] T.-W. Hui and C. C. Loy, "Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation," in *Proc. ECCV*, pp. 169–184, 2020.

[32] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. CVPR*, pp. 1647–1655, 2017.

[33] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Models matter, so does training: An empirical study of cnns for optical flow estimation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 42, no. 6, pp. 1408–1423, 2020.

[34] G. Yang and D. Ramanan, "Volumetric correspondence networks for optical flow," in *Proc. NeurIPS*, pp. 794–805, 2019.

[35] S. Zhao, Y. Sheng, Y. Dong, E. I.-C. Chang, and Y. Xu, "Maskflownet: Asymmetric feature matching with learnable occlusion mask," in *Proc. CVPR*, pp. 6278–6287, 2020.

[36] J. Wang, Y. Zhong, Y. Dai, K. Zhang, P. Ji, and H. Li, "Displacement-invariant matching cost learning for accurate optical flow estimation," in *Proc. NeurIPS*, pp. 15220–15231, 2020.

[37] A. Ahmadi and I. Patras, "Unsupervised convolutional neural networks for motion estimation," in *Proc. ICIP*, pp. 1629–1633, 2016.

[38] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised deep learning for optical flow estimation," in *Proc. AAAI*, pp. 1495–1501, 2017.

[39] J. Yu, A. Harley, and K. Derpanis, "Back to basics:unsupervised learning of optical flow via brightness constancy and motion smoothness," in *Proc. ECCV Workshops*, pp. 3–10, 2016.

[40] Y. Wang, Y. Yang, Z. Yang, L. Zhao, P. Wang, and W. Xu, "Occlusion aware unsupervised learning of optical flow," in *Proc. CVPR*, pp. 4884–4893, 2018.

[41] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sanchez, "Symmetrical dense optical flow estimation with occlusions detection," *International Journal of Computer Vision*, vol. 75, no. 3, pp. 371–385, 2007.

[42] J. Janai, F. Güney, A. Ranjan, M. Black, and A. Geiger, "Unsupervised learning of multi-frame optical flow with occlusions," in *Proc. ECCV*, pp. 713–731, 2018.

[43] Z. Ren, W. Luo, J. Yan, W. Liao, X. Yang, A. Yuille, and H. Zha, "Stflow: Self-taught optical flow estimation using pseudo labels," *IEEE Trans. on Image Processing*, vol. 29, pp. 9113–9124, 2020.

[44] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proc. CVPR*, pp. 1164–1172, 2015.

[45] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation," in *Proc. CVPR*, pp. 12240–12249, 2019.

[46] Y. Zou, Z. Luo, and J.-B. Huang, "Df-net: Unsupervised joint learning of depth and flow using cross-task consistency," in *Proc. ECCV*, pp. 38–55, 2018.

[47] Z. Yin and J. Shi, "Geonet: Unsupervised learning of dense depth, optical flow and camera pose," in *Proc. CVPR*, pp. 1983–1992, 2018.

[48] L. Liu, G. Zhai, W. Ye, and Y. Liu, "Unsupervised learning of scene flow estimation fusing with local rigidity," in *Proc. IJCAI*, pp. 876–882, 2019.

[49] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu, "Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos," in *Proc. CVPR*, pp. 8063–8073, 2019.

[50] L. Tian, Z. Tu, D. Zhang, J. Liu, B. Li, and J. Yuan, "Unsupervised learning of optical flow with cnn-based non-local filtering," *IEEE Trans. on Image Processing*, vol. 29, pp. 8429–8442, 2020.

[51] Z. Ren, J. Yan, X. Yang, A. Yuille, and H. Zha, "Unsupervised learning of optical flow with patch consistency and occlusion estimation," *Pattern Recognition*, vol. 103, p. 107191, 2020.

[52] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele, "Joint bilateral upsampling," *ACM Trans. Graphics*, vol. 26, no. 3, pp. 96–es, 2007.

[53] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proc. ECCV*, pp. 1–14, 2010.

[54] Y. Li, D. Min, M. N. Do, and J. Lu, "Fast guided global interpolation for depth and motion," in *Proc. ECCV*, pp. 717–733, 2016.

[55] H. Wu, S. Zheng, J. Zhang, and K. Huang, "Fast end-to-end trainable guided filter," in *Proc. CVPR*, pp. 1838–1847, 2018.

[56] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz, "Pixel-adaptive convolutional neural networks," in *Proc. CVPR*, pp. 11166–11175, 2019.

[57] U. Ojha, K. K. Singh, C.-J. Hsieh, and Y. J. Lee, "Elastic-infogan: Unsupervised disentangled representation learning in imbalanced data," in *Proc. NeurIPS*, pp. 18063–18075, 2020.

[58] Y. Wang and J. M. Solomon, "Prnet: Self-supervised learning for partial-to-partial registration," in *Proc. NeurIPS*, pp. 8812–8824, 2019.

[59] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep joint image filtering," in *Proc. ECCV*, pp. 154–169, 2016.

Shuaicheng Liu received the Ph.D. and M.Sc. degrees from the National University of Singapore, Singapore, in 2014 and 2010, respectively, and the B.E. degree from Sichuan University, Chengdu, China, in 2008. In 2014, he joined the University of Electronic Science and Technology of China and is currently an associate professor with the Institute of Image Processing, School of Information and Communication Engineering, Chengdu, China. His research interests include computer vision and computer graphics.

Kunming Luo received the B.E. and M.S. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2016 and 2019. Now he is a researcher of Megvii Technology Chengdu. His research interests include computer vision and deep learning.

Ao Luo received the BE degree in automation from Southwest Jiaotong University (SWJTU), Chengdu, China, in 2013, and the Ph.D. degree in control science and engineering with the School of Automation Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2020. He was a visiting scholar in the department of computer science, State University of New York at Albany (UAlbany). Currently, he is a researcher in MEGVII Technology, Chengdu, China. His research interest is focused on deep learning for computer vision.

Chuan Wang received his Ph.D degree from The University of Hong Kong in 2015, and B.Eng degree from University of Science and Technology of China in 2010. He was a computer vision staff researcher in Lenovo Group Limited, Hong Kong. He worked as a visiting scholar in National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, China in 2009 and State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, China in 2010. He started his training program in Megvii in 2018. His research interests include video analysis and computer vision.

Fanman Meng received the Ph.D. degree in signal and information processing from the University of Electronic Science and Technology of China, Chengdu, China, in 2014. From 2013 to 2014, he was a Research Assistant with the Division of Visual and Interactive Computing, Nanyang Technological University, Singapore. He is currently an Associate Professor with the School of Electronic Engineering, University of Electronic Science and Technology of China. He has authored or co-authored numerous technical articles in well-known international journals and conferences. His current research interests include image segmentation and object detection. He is a member of the IEEE Circuits and Systems Society. He was a recipient of the Best Student Paper Honorable Mention Award at the 12th Asian Conference on Computer Vision, Singapore, in 2014, and the Top 10% Paper Award at the IEEE International Conference on Image Processing, Paris, France, in 2014.

Bing Zeng received the B.E. and M.Sc. degrees in Electronic Engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1983 and 1986, respectively, and the Ph.D. degree in Electrical Engineering from the Tampere University of Technology, Tampere, Finland, in 1991. He worked as a Postdoctoral Fellow with the University of Toronto from September 1991 to July 1992 and as a Researcher with Concordia University from August 1992 to January 1993. Then he joined the Hong Kong University of Science and Technology (HKUST). After 20 years of service, he returned to the UESTC in the summer of 2013, through Chinas 1000-Talent-Scheme. At UESTC, he leads the Institute of Image Processing to work on image and video processing, 3-D and multiview video technology, and visual big data. During his tenure with the HKUST and UESTC, he graduated more than 30 Master's and Ph.D. students, received about 20 research grants, filed 8 international patents, and published more than 250 papers. He served as an Associate Editor for the IEEE TCSVT for 8 years and received the Best Associate Editor Award in 2011. He was General Co-Chair of the IEEE VCIP-2016, held in Chengdu, China, in November 2016. He is currently on the Editorial Board of Journal of Visual Communication and Image Representation and serves as General Co-Chair of PCM-2017. He was the recipient of a 2nd Class Natural Science Award (the first recipient) from the Ministry of Education of China in 2014 and was elected as an IEEE Fellow in 2016 for contributions to image and video coding.