UPFlow: Upsampling Pyramid for Unsupervised Optical Flow Learning

Kunming Luo[1] Chuan Wang[1] Shuaicheng Liu[2,1*] Haoqiang Fan[1] Jue Wang[1] Jian Sun[1]

[1]Megvii Technology [2]University of Electronic Science and Technology of China

https://github.com/coolbeam/UPFlow_pytorch

Abstract

We present an unsupervised learning approach for optical flow estimation by improving the upsampling and learning of pyramid network. We design a self-guided upsample module to tackle the interpolation blur problem caused by bilinear upsampling between pyramid levels. Moreover, we propose a pyramid distillation loss to add supervision for intermediate levels via distilling the finest flow as pseudo labels. By integrating these two components together, our method achieves the best performance for unsupervised optical flow learning on multiple leading benchmarks, including MPI-SIntel, KITTI 2012 and KITTI 2015. In particular, we achieve EPE=1.4 on KITTI 2012 and F1=9.38% on KITTI 2015, which outperform the previous state-of-the-art methods by 22.2% and 15.7%, respectively.

1. Introduction

Optical flow estimation has been a fundamental computer vision task for decades, which has been widely used in various applications such as video editing [14], behavior recognition [31] and object tracking [3]. The early solutions focus on minimizing a pre-defined energy function with optimization tools [4, 33, 30]. Nowadays deep learning based approaches become popular, which can be classified into two categories, the supervised [11, 26] and unsupervised ones [29, 37]. The former one uses synthetic or human-labelled dense optical flow as ground-truth to guide the motion regression. The supervised methods have achieved leading performance on the benchmark evaluations. However, the acquisition of ground-truth labels are expensive. In addition, the generalization is another challenge when trained on synthetic datasets. As a result, the latter category, *i.e*. the unsupervised approaches attracts more attentions recently, which does not require the ground-truth labels. In unsupervised methods, the photometric loss between two images is commonly used to train the optical flow estimation network. To facilitate the training, the pyramid network structure [34, 10] is often adopted, such that both global and local motions can be captured in a coarse-to-fine manner. However, there are two main issues with respect to the pyramid learning, which are often ignored previously. We refer the two issues as **bottom-up** and **top-down** problems. The bottom-up problem refers to the upsampling module in the pyramid. Existing methods often adopt simple bilinear or bicubic upsampling [20, 15], which interpolates cross edges, resulting in blur artifacts in the predicted optical flow. Such errors will be propagated and aggregated when the scale becomes finer. Fig. 1 shows an example. The top-down problem refers to the pyramid supervision. The previous leading unsupervised methods typically add guidance losses only on the final output of the network, while the intermediate pyramid levels have no guidance. In this condition, the estimation errors in coarser levels will accumulate and damage the estimation at finer levels due to the lack of training guidance.

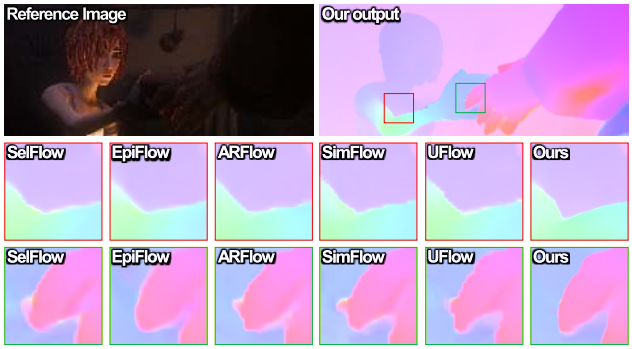


Figure 1. An example from Sintel Final benchmark. Compared with previous unsupervised methods including SelFlow [22], EpiFlow [44], ARFlow [20], SimFlow [12] and UFlow [15], our approach produces sharper and more accurate results in object edges.

To this end, we propose an enhanced pyramid learning framework of unsupervised optical flow estimation. First, we introduce a self-guided upsampling module that supports blur-free optical flow upsampling by using a self-learned interpolation flow instead of the straightforward in-

*Corresponding author

terpolations. Second, we design a new loss named pyramid distillation loss that supports explicitly learning of the intermediate pyramid levels by taking the finest output flow as pseudo labels. To sum up, our main contributions include:

- We propose a **self-guided upsampling module** to tackle the interpolation problem in the pyramid network, which can generate the sharp motion edges.
- We propose a **pyramid distillation loss** to enable robust supervision for unsupervised learning of coarse pyramid levels.
- We achieve superior performance over the state-of-the-art unsupervised methods with a relatively large margin, validated on multiple leading benchmarks.

2. Related Work

2.1. Supervised Deep Optical Flow

Supervised methods require annotated flow ground-truth to train the network [2, 43, 39, 40]. FlowNet [6] was the first work that proposed to learn optical flow estimation by training fully convolutional networks on synthetic dataset FlyingChairs. Then, FlowNet2 [11] proposed to iteratively stack multiple networks for the improvement. To cover the challenging scene with large displacements, SpyNet [26] built a spatial pyramid network to estimate optical flow in a coarse-to-fine manner. PWC-Net [34] and LiteFlowNet [9] proposed to build efficient and lightweight networks by warping feature and calculating cost volume at each pyramid level. IRR-PWC [10] proposed to design pyramid network by an iterative residual refinement scheme. Recently, RAFT [35] proposed to estimate flow fileds by 4D correlation volume and recurrent network, yielding state-of-the-art performance. In this paper, we work in unsupervised setting where no ground-truth labels are required.

2.2. Unsupervised Deep Optical Flow

Unsupervised methods do not need annotations for training [1, 42], which can be divided into two categories: the occlusion handling methods and the alignment learning methods. The occlusion handling methods mainly focus on excluding the impact of the occlusion regions that cannot be aligned inherently. For this purpose, many methods are proposed, including the occlusion-aware losses by forward-backward occlusion checking [24] and range-map occlusion checking [37], the data distillation methods [21, 22, 28], and augmentation regularization loss [20]. On the other hand, the alignment learning methods are mainly developed to improve optical flow learning under multiple image alignment constrains, including the census transform constrain [29], multi-frame formulation [13], epipolar constrain [44], depth constrains [27, 45, 41, 19] and feature similarity constrain [12]. Recently, UFlow [15] achieved the state-of-the-art performance on multiple benchmarks by systematically analyzing and integrating multiple unsupervised components into a unified framework. In this paper, we propose to improve optical flow learning with our improved pyramid structure.

2.3. Image Guided Optical Flow Upsampling

A series of methods have been developed to upsample images, depths or optical flows by using the guidance information extracted from high resolution images. The early works such as joint bilateral upsampling [16] and guided image filtering [8] proposed to produce upsampled results by filters extracted from the guidance images. Recent works [18, 38, 32] proposed to use deep trainable CNNs to extract guidance feature or guidance filter for upsampling. In this paper, we build an efficient and lightweight self-guided upsampling module to extract interpolation flow and interpolation mask for optical flow upsampling. By inserting this module into a deep pyramid network, high quality results can be obtained.

3. Algorithm

3.1. Pyramid Structure in Optical Flow Estimation

Optical flow estimation can be formulated as:

$$V_f = \mathcal{H}(\theta, I_t, I_{t+1}), \tag{1}$$

where I_t and I_{t+1} denote the input images, $\mathcal{H}$ is the estimation model with parameter θ, and V_f is the forward flow field that represents the movement of each pixel in I_t towards its corresponding pixel in I_{t+1}.

The flow estimation model $\mathcal{H}$ is commonly designed as a pyramid structure, such as the classical PWC-Net [34]. The pipeline of our network is illustrated in Fig. 2. The network can be divided into two stages: pyramid encoding and pyramid decoding. In the first stage, we extract feature pairs in different scales from the input images by convolutional layers. In the second stage, we use a decoder module $\mathcal{D}$ and an upsample module $\mathcal{S}_\uparrow$ to estimate optical flows in a coarse-to-fine manner. The structure of the decoder module $\mathcal{D}$ is the same as in UFlow [15], which contains feature warping, cost volume construction by correlation layer, cost volume normalization, and flow decoding by fully convolutional layers. Similar to recent works [10, 15], we also make the parameters of $\mathcal{D}$ and $\mathcal{S}_\uparrow$ shared across all the pyramid levels. In summary, the pyramid decoding stage can be formulated as follows:

$$\hat{V}_f^{i-1} = \mathcal{S}_\uparrow(F_t^i, F_{t+1}^i, V_f^{i-1}), \tag{2}$$

$$V_f^i = \mathcal{D}(F_t^i, F_{t+1}^i, \hat{V}_f^{i-1}), \tag{3}$$

where $i \in \{0, 1, ..., N\}$ is the index of each pyramid level and the smaller number represents the coarser level, F_t^i and

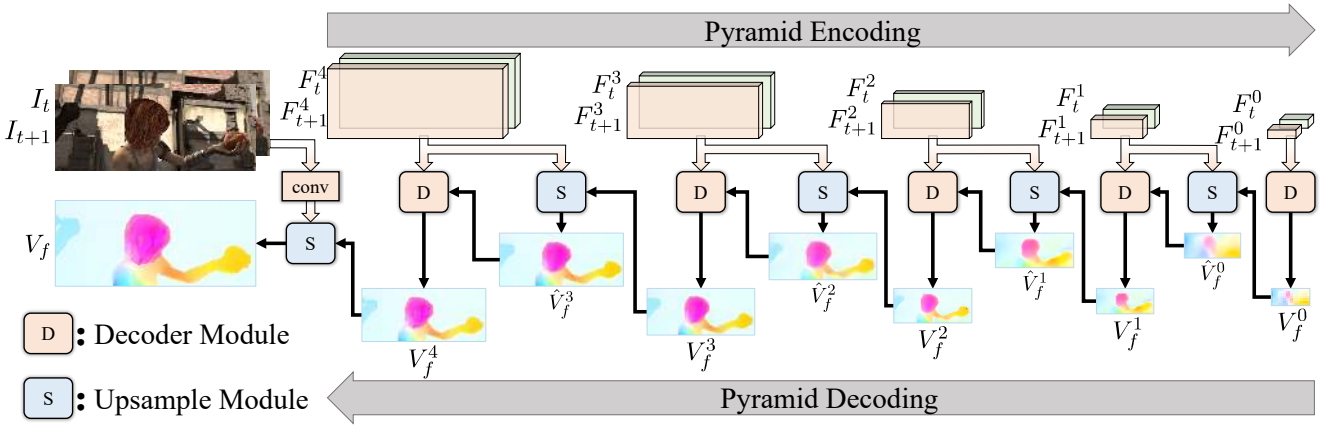


Figure 2. Illustration of the pipeline of our network, which contains two stage: pyramid encoding to extract feature pairs in different scales and pyramid decoding to estimate optical flow in each scale. Note that the parameters of the decoder module and the upsample module are shared across all the pyramid levels.

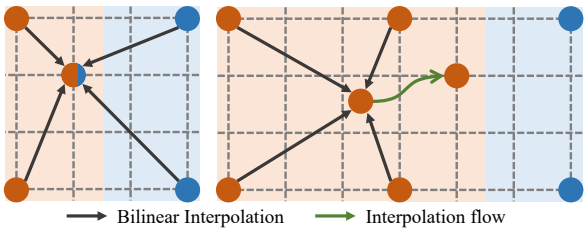


Figure 3. Illustration of bilinear upsampling (left) and the idea of our self-guided upsampling (right). Red and blue dots are motion vectors from different objects. Bilinear upsampling often produces cross-edge interpolation. We propose to first interpolate a flow vector in other position without crossing edge and then bring it to the desired position by our learned interpolation flow.

F^i_{t+1} are features extracted from I_t and I_{t+1} at the i-th level, and $\hat{V}_f^{i-1}$ is the upsampled flow of the $i-1$ level. In practice, considering the accuracy and efficiency, N is usually set to 4 [10, 34]. The final optical flow result is obtained by directly upsampling the output of the last pyramid level. Particularly, in Eq. 2, the bilinear interpolation is commonly used to upsample flow fields in previous methods [10, 15], which may yield noisy or ambiguity results at object boundaries. In this paper, we present a self-guided upsample module to tackle this problem as detailed in Sec. 3.2.

3.2. Self-guided Upsample Module

In Fig. 3 left, the case of bilinear interpolation is illustrated. We show 4 dots which represent 4 flow vectors, belonging to two motion sources, marked as red and blue, respectively. The missing regions are then bilinear interpolated with no semantic guidance. Thus, a mixed interpolation result is generated at the red motion area, resulting in cross-edge interpolation. In order to alleviate this problem, we propose a self-guided upsample module (SGU) to

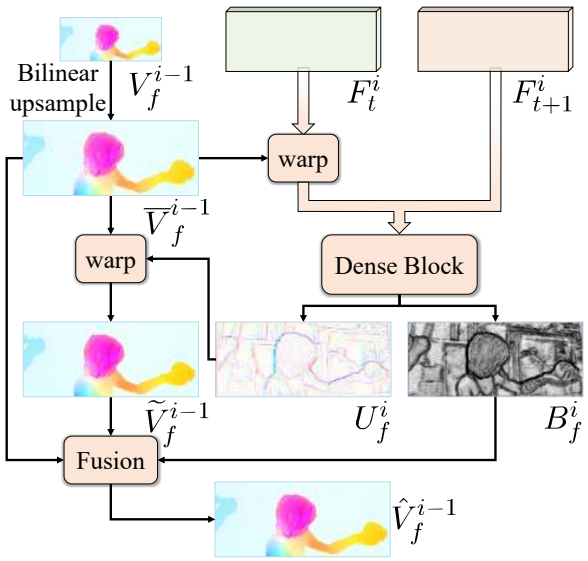


Figure 4. Illustration of our self-guided upsample module. We first upscale the input low resolution flow V_f^{i-1} by bilinear upsampling and use a dense block to compute an interpolation flow U_f^i and an interpolation map B_f^i. Then we generate the high resolution flow by warping and fusion.

change the interpolation source points by an interpolation flow. The main idea of our SGU is shown in Fig. 3 right. We first interpolate a point by its enclosing red motions and then bring the result to the target place with the learned interpolation flow (Fig. 3, green arrow). As a result, the mixed interpolation problem can be avoided.

In our design, to keep the interpolation in plat areas from being changed and make the interpolation flow only applied on motion boundary areas, we learn a per-pixel weight map to indicate where the interpolation flow should be disabled. Thus, the upsampling process of our SGU is a weighted

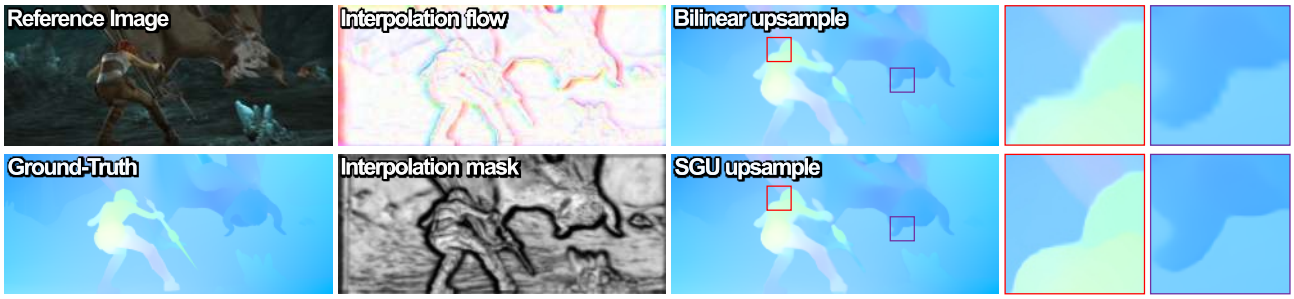


Figure 5. Visual example of our self-guided upsample module (SGU) on MPI-Sintel Final dataset. Results of bilinear method and our SGU are shown. The zoom-in patches are also shown on the right of each sample for better comparison.

combination of the bilinear upsampled flow and a modified flow obtained by warping the upsampled flow with the interpolation flow. The detailed structure of our SGU module is shown in Fig. 4. Given a low resolution flow V_f^{i-1} from the $i-1$-th level, we first generate an initial flow $\overline{V}_f^{i-1}$ in higher resolution by bilinear interpolation:

$$\overline{V}_f^{i-1}(\boldsymbol{p}) = \sum_{\boldsymbol{k}\in\mathcal{N}(\boldsymbol{p}/s)} w(\boldsymbol{p}/s, \boldsymbol{k}) V_f^{i-1}(\boldsymbol{k}), \tag{4}$$

where $\boldsymbol{p}$ is a pixel coordinate in higher resolution, s is the scale magnification, $\mathcal{N}$ denotes the four neighbour pixels, and $w(\boldsymbol{p}/s, \boldsymbol{k})$ is the bilinear interpolation weights. Then, we compute an interpolation flow U_f^i from features F_t^i and F_{t+1}^i to change the interpolation of $\overline{V}_f^{i-1}$ by warping:

$$\widetilde{V}_f^{i-1}(\boldsymbol{p}) = \sum_{\boldsymbol{k}\in\mathcal{N}(\boldsymbol{d})} w(\boldsymbol{d}, \boldsymbol{k}) \overline{V}_f^{i-1}(\boldsymbol{k}), \tag{5}$$

$$\boldsymbol{d} = \boldsymbol{p} + U_f^i(\boldsymbol{p}), \tag{6}$$

where $\widetilde{V}_f^{i-1}$ is the result of warping $\overline{V}_f^{i-1}$ by the interpolation flow U_f^i. Since the interpolation blur only occurs in object edge regions, it is unnecessary to learn interpolation flow in flat regions. We thus use an interpolation map B_f^i to explicitly force the model to learn interpolation flow only in motion boundary regions. The final upsample result is the fusion of $\widetilde{V}_f^{i-1}$ and $\overline{V}_f^{i-1}$:

$$\hat{V}_f^{i-1} = B_f^i \odot \overline{V}_f^{i-1} + (1 - B_f^i) \odot \widetilde{V}_f^i, \tag{7}$$

where $\hat{V}_f^{i-1}$ is the output of our self-guided upsample module and $\odot$ is the element-wise multiplier.

To produce the interpolation flow U_f^i and the interpolation map B_f^i, we use a dense block with 5 convolutional layers. Specifically, we concatenate the feature map F_t^i and the warped feature map F_{t+1}^i as the input of the dense block. The kernel number of each convolutional layer in the dense block is 32, 32, 32, 16, 8 respectively. The output of the dense block is a tensor map with **3 channels**. We use the first two channels of the tensor map as the interpolation flow and use the last channel to form the interpolation map through a sigmoid layer. Note that, no supervision is introduced for the learning of interpolation flow and interpolation map. Fig. 5 shows an example from MPI-Sintel Final dataset, where our SGU produces cleaner and sharper results at object boundaries compared with the bilinear method. Interestingly, the self-learned interpolation map is nearly to be an edge map and the interpolation flow is also focused on object edge regions.

3.3. Loss Guidance at Pyramid Levels

In our framework, we use several losses to train the pyramid network: the unsupervised optical flow losses for the final output flow and the pyramid distillation loss for the intermediate flows at different pyramid levels.

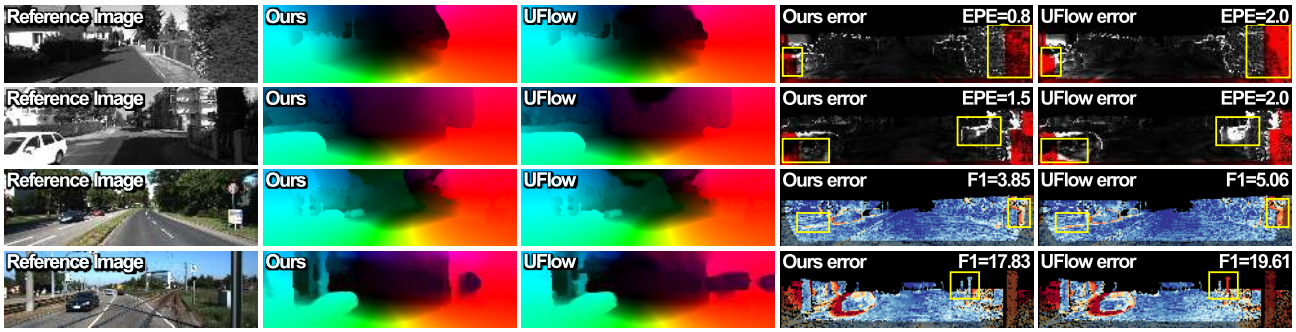
3.3.1 Unsupervised Optical Flow Loss

To learn the flow estimation model $\mathcal{H}$ in unsupervised setting, we use the photometric loss $\mathcal{L}_m$ based on the brightness constancy assumption that the same objects in I_t and I_{t+1} must have similar intensities. However, some regions may be occluded by moving objects, so that their corresponding regions do not exist in another image. Since the photometric loss can not work in these regions, we only add $\mathcal{L}_m$ on the non-occluded regions. The photometric loss can be formulated as follows:

$$\mathcal{L}_m = \frac{\sum_{\boldsymbol{p}} \Psi\Big(I_t(\boldsymbol{p}) - I_{t+1}\big(\boldsymbol{p} + V_f(\boldsymbol{p})\big)\Big) \cdot M_t(\boldsymbol{p})}{\sum_{\boldsymbol{p}} M_1(\boldsymbol{p})}, \tag{8}$$

where M_t is the occlusion mask and Ψ is the robust penalty function [21]: $\Psi(x) = (|x| + \epsilon)^q$ with q, ϵ being 0.4 and 0.01. In the occlusion mask M_t, which is estimated by forward-backward checking [24], 1 represents the non-occluded pixels in I_t and 0 for those are occluded.

To improve the performance, some previously effective unsupervised components are also added, including smooth loss [37] $\mathcal{L}_s$, census loss [24] $\mathcal{L}_c$, augmentation regularization loss [20] $\mathcal{L}_a$, and boundary dilated warping loss [23]



(a) Visual comparison on KITTI 2012 (first two rows) and KITTI 2015 (last two rows).

(b) Visual comparison on Sintel Clean (first two rows) and Sintel Final (last two rows).

Figure 6. Visual comparison of our method with the state-of-the-art method UFlow [15] on KITTI (a) and Sintel (b) benchmarks. The error maps visualized by the benchmark websites are shown in the last two columns with obvious difference regions marked by yellow boxes.

$\mathcal{L}_b$. For simplicity, we omit these components. Please refer to previous works for details. The capability of these components will be discussed in Sec. 4.3.

3.3.2 Pyramid Distillation Loss

To learn intermediate flow for each pyramid level, we propose to distillate the finest output flow to the intermediate ones by our pyramid distillation loss $\mathcal{L}_d$. Intuitively, this is equivalent to calculating all the unsupervised losses on each of the intermediate outputs. However, the photometric consistency measurement is not accurate enough for optical flow learning at low resolutions [15]. As a result, it is inappropriate to enforce unsupervised losses at intermediate levels, especially at the lower pyramid levels. Therefore, we propose to use the finest output flow as pseudo labels and add supervised losses instead of unsupervised losses for intermediate outputs.

To calculate $\mathcal{L}_d$, we directly downsample the final output flow and evaluate its difference with the intermediate flows. Since occlusion regions are excluded from $\mathcal{L}_m$, flow estimation in occlusion regions is noisy. In order to eliminate the influence of these noise regions in the pseudo label, we also downsample the occlusion mask M_t and exclude occlusion regions from $\mathcal{L}_d$. Thus, our pyramid distillation loss can be formulated as follow:

$$\mathcal{L}_d = \sum_{i=0}^{N} \sum_{\boldsymbol{p}} \Psi\left(V_f^i - \mathcal{S}_{\downarrow}(s_i, V_f)\right) \cdot \mathcal{S}_{\downarrow}(s_i, M_t), \tag{9}$$

where s_i is the scale magnification of pyramid level i and $\mathcal{S}_{\downarrow}$ is the downsampling function.

Eventually, our training loss $\mathcal{L}$ is formulated as follows:

$$\mathcal{L} = \mathcal{L}_m + \lambda_d \mathcal{L}_d + \lambda_s \mathcal{L}_s + \lambda_c \mathcal{L}_c + \lambda_a \mathcal{L}_a + \lambda_b \mathcal{L}_b, \tag{10}$$

where λ_d, λ_s, λ_c, λ_a and λ_b are hyper-parameters and we set $\lambda_d = 0.01$, $\lambda_s = 0.05$, $\lambda_c = 1$, $\lambda_a = 0.5$ and $\lambda_b = 1$.

4. Experimental Results

4.1. Dataset and Implementation Details

We conduct experiments on three datasets: MPI-Sintel [5], KITTI 2012 [7] and KITTI 2015 [25]. We use

	Method	KITTI 2012		KITTI 2015		Sintel Clean		Sintel Final	
		train	test	train	test (F1-all)	train	test	train	test
Supervised	FlowNetS [6]	8.26	–	–	–	4.50	7.42	5.45	8.43
	FlowNetS+ft [6]	7.52	9.1	–	–	(3.66)	6.96	(4.44)	7.76
	SpyNet [26]	9.12	–	–	–	4.12	6.69	5.57	8.43
	SpyNet+ft [26]	8.25	10.1	–	35.07%	(3.17)	6.64	(4.32)	8.36
	LiteFlowNet [9]	4.25	–	10.46	–	2.52	–	4.05	–
	LiteFlowNet+ft [9]	(1.26)	1.7	(2.16)	10.24%	(1.64)	4.86	(2.23)	6.09
	PWC-Net [34]	4.14	–	10.35	–	2.55	–	3.93	–
	PWC-Net+ft [34]	(1.45)	1.7	(2.16)	9.60%	(1.70)	3.86	(2.21)	5.13
	IRR-PWC+ft [10]	–	–	(1.63)	7.65%	(1.92)	3.84	(2.51)	4.58
	RAFT [35]	–	–	5.54	–	1.63	–	2.83	–
	RAFT-ft [35]	–	–	–	6.30%	–	2.42	–	3.39
Unsupervised	BackToBasic [42]	11.30	9.9	–	–	–	–	–	–
	DSTFlow [29]	10.43	12.4	16.79	39%	(6.16)	10.41	(6.81)	11.27
	UnFlow [24]	3.29	–	8.10	23.3%	–	9.38	(7.91)	10.22
	OAFlow [37]	3.55	4.2	8.88	31.2%	(4.03)	7.95	(5.95)	9.15
	Back2Future [13]	–	–	6.59	22.94%	(3.89)	7.23	(5.52)	8.81
	NLFlow [36]	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 [44]	(2.51)	3.4	(5.55)	16.95%	(3.54)	7.00	(4.99)	8.51
	SelFlow [22]	1.69	2.2	4.84	14.19%	(2.88)	6.56	(3.87)	6.57
	STFlow [36]	1.64	1.9	3.56	13.83%	(2.91)	6.12	(3.59)	6.63
	ARFlow [20]	1.44	1.8	2.85	11.80%	(2.79)	4.78	(3.87)	5.89
	SimFlow [12]	–	–	5.19	13.38%	(2.86)	5.92	(3.57)	6.92
	UFlow [15]	1.68	1.9	2.71	11.13%	(2.50)	5.21	(3.39)	6.50
	Ours	1.27	1.4	2.45	9.38%	(2.33)	4.68	(2.67)	5.32

Table 1. Comparison with previous methods. We use the average EPE error (the lower the better) as evaluation metric for all the datasets except on KITTI 2015 benchmark test, where the F1 measurement (the lower the better) is used. Missing entries '–' indicates that the result is not reported in the compared paper, and $(\cdot)$ indicates that the testing images are used during unsupervised training. The best unsupervised results are marked in red and the second best are in blue. Note that, for results of the supervised methods, '+ft' means the model is trained on the target domain, otherwise, the model is trained on synthetic datasets such as Flying Chairs [6] and Flying Chairs occ [10]. For unsupervised methods, we report the performance of the model trained using images from target domain.

the same dataset setting as previous unsupervised methods [20, 15]. For MPI-Sintel dataset, which contains $1,041$ training image pairs rendered in two different passes ('Clean' and 'Final'), we use all the training images from both 'Clean' and 'Final' to train our model. For KITTI 2012 and 2015 datasets, we pretrain our model using $28,058$ image pairs from the KITTI raw dataset and then finetune our model on the multi-view extension dataset. The flow ground-truth is only used for validation.

We implement our method with PyTorch, and complete the training in $1000k$ iterations with batch size of 4. The total number of parameters of our model is 3.49M, in which the proposed self-guided upsample module has 0.14M trainable parameters. Moreover, the running time of our full model is 0.05s for a Sintel image pair with resolution 436×1024. The standard average endpoint error (EPE) and the percentage of erroneous pixels (F1) are used as the evaluation metric of optical flow estimation.

4.2. Comparison with Existing Methods

We compare our method with existing supervised and unsupervised methods on leading optical flow benchmarks. Quantitative results are shown in Table 1, where our method outperforms all the previous unsupervised methods on all the datasets. In Table 1, we mark the best results by red and the second best by blue in unsupervised methods.

Comparison with Unsupervised Methods. On KITTI 2012 online evaluation, our method achieves EPE=1.4, which improves the EPE=1.8 of the previous best method ARFlow [20] by 22.2%. Moreover, on KITTI 2015 online evaluation, our method reduces the F1-all value of 11.13% in UFlow [15] to 9.38% with 15.7% improvement.

CL	BDWL	ARL	SGU	PDL	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)	(12.93)	(4.19)	(2.59)	(13.64)
✓					3.39	1.09	16.58	6.89	2.20	28.12	(3.41)	(1.62)	(13.51)	(3.85)	(2.17)	(13.71)
✓	✓				1.42	0.91	4.39	3.00	2.12	6.89	(2.84)	(1.50)	(10.63)	(3.60)	(2.28)	(11.52)
✓	✓	✓			1.37	0.93	3.98	2.64	1.96	6.01	(2.61)	(1.33)	(10.14)	(3.17)	(1.92)	(10.70)
✓	✓	✓	✓		1.33	0.88	4.00	2.56	1.91	5.35	(2.46)	(1.17)	(9.89)	(2.79)	(1.53)	(10.28)
✓	✓	✓		✓	1.36	0.91	4.03	2.61	1.96	5.52	(2.53)	(1.23)	(10.12)	(2.93)	(1.67)	(10.38)
✓	✓	✓	✓	✓	**1.27**	**0.85**	**3.77**	**2.45**	**1.87**	**5.32**	**(2.33)**	**(1.07)**	**(9.66)**	**(2.63)**	**(1.39)**	**(9.91)**

Table 2. Ablation study of the unsupervised components. CL: census loss [24], BDWL: boundary dilated warping loss [23], ARL: augmentation regularization loss [20], SGU: self-guided upsampling, PDL: pyramid distillation loss. The best results are marked in bold.

On the test benchmark of MPI-Sintel dataset, we achieve EPE=4.68 on the 'Clean' pass and EPE=5.32 on the 'Final' pass, both outperforming all the previous methods. Some qualitative comparison results are shown in Fig. 6, where our method produces more accurate results than the state-of-the-art method UFlow [15].

Comparison with Supervised Methods. As shown in Table 1, representative supervised methods are also reported for comparison. In practical applications where flow ground-truth is not available, the supervised methods can only train models using synthetic datasets. In contrast, unsupervised methods can be directly implemented using images from the target domain. As a result, on KITTI and Sintel Final datasets, our method outperforms all the supervised methods trained on synthetic datasets, especially in real scenarios such as the KITTI 2015 dataset.

As for the in-domain ability, our method is also comparable with supervised methods. Interestingly, on KITTI 2012 and 2015 datasets, our method achieve EPE=1.4 and F1=9.38%, which outperforms classical supervised methods such as PWC-Net [34] and LiteFlowNet [9].

4.3. Ablation Study

To analyze the capability and design of each individual component, we conduct extensive ablation experiments on the train set of KITTI and MPI-Sintel datasets following the setting in[22, 12]. The EPE error over all pixels (ALL), non-occluded pixels (NOC) and occluded pixels (OCC) are reported for quantitative comparisons.

Unsupervised Components. Several unsupervised components are used in our framework including census loss [24] (CL), boundary dilated warping loss [23] (BDWL), augmentation regularization loss [20] (ARL), our proposed self-guided upsampling (SGU) and pyramid distillation loss (PDL). We assess the effect of these components in Table. 2. In the first row of Table. 2, we only use photometric loss and smooth loss to train the pyramid network with our SGU disabled. Comparing the first four rows in Table. 2, we can see that by combining CL, BDWL and ARL, the performance of optical flow estimation can be improved, which is equivalent to the current best performance reported in UFlow [15]. Comparing the last four rows in Table. 2, we can see that: (1) the EPE error can be reduced by using our SGU to solve the bottom-up interpolation problem; (2) the top-down supervision information by our PDL can also improve the performance; (3) the performance can be further improved by combining the SGU and PDL.

Method	KITTI 2012	KITTI 2015	Sintel Clean	Sintel Final
Bilinear	1.36	2.61	(2.53)	(2.93)
JBU [16]	1.51	3.00	(2.66)	(2.98)
GF [8]	1.40	2.90	(2.72)	(2.92)
DJF [17]	1.36	2.79	(2.75)	(3.20)
DGF [38]	1.41	3.14	(2.69)	(3.05)
PAC [32]	1.42	2.65	(2.58)	(2.95)
SGU-FM	1.35	2.60	(2.52)	(2.91)
SGU-M	1.33	2.59	(2.41)	(2.86)
SGU	**1.27**	**2.45**	**(2.33)**	**(2.63)**

Table 3. Comparison of our SGU with different upsampling methods: the basic bilinear upsampling, image guided upsampling methods including JBU [16], GF [8], DJF [17], DGF [38] and PAC [32], and the variants of SGU such as SGU-FM, where the interpolation flow and weight map are both removed, and SGU-M, where the only the interpolation map is removed.

Some qualitative comparison results are shown in Fig. 7, where 'Full' represents our full method, 'W/O SGU' means the SGU module of our network is disabled and 'W/O PDL' means the PDL is not considered during training. Comparing with our full method, the boundary of the predicted flow becomes blurry when SGU is removed while the error increases when PDL is removed.

Self-guided Upsample Module. There is a set of methods that use image information to guide the upsampling process, e.g., JBU [16], GF [8], DJF [17], DGF [38] and PAC [32]. We implement them into our pyramid network and train with the same loss function for comparisons. The average EPE errors of the validation sets are reported in Table. 3. As a result, our SGU is superior to the image guided upsampling methods. The reason lies in two folds: (1) the guidance information that directly extracted from images

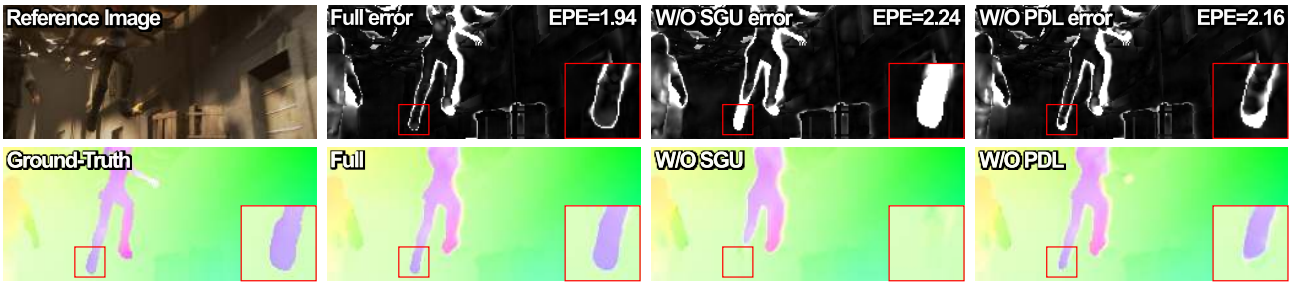


Figure 7. Visual results of removing the SGU or PDL from our full method on Sintel Clean (the first sample) and Sintel Final (the second sampe). The room in flows and error maps are shown in the right corner of each sample.

Method	Sintel Clean train						Sintel Final train					
	×1	×4	×8	×16	×32	×64	×1	×4	×8	×16	×32	×64
w/o PL	(2.46)	(2.53)	(2.78)	(3.38)	(4.70)	(7.39)	(2.79)	(2.89)	(3.11)	(3.73)	(5.07)	(7.59)
PUL-up	(2.45)	(2.52)	(2.75)	(3.35)	(4.61)	(7.32)	(2.77)	(2.86)	(3.09)	(3.68)	(5.00)	(7.52)
PUL-down	(2.49)	(2.56)	(2.82)	(3.43)	(4.84)	(7.69)	(2.80)	(2.89)	(3.12)	(3.74)	(5.18)	(8.26)
PDL w/o occ	(2.37)	(2.42)	(2.61)	(3.15)	(4.17)	(6.31)	(2.73)	(2.81)	(3.00)	(3.59)	(4.82)	(7.16)
PDL	**(2.33)**	**(2.37)**	**(2.56)**	**(3.03)**	**(3.88)**	**(5.58)**	**(2.63)**	**(2.69)**	**(2.87)**	**(3.38)**	**(4.43)**	**(6.39)**

Table 4. Comparison of different pyramid losses: no pyramid loss (w/o PL), pyramid unsupervised loss by upsampling intermediate flows to image resolution to compute unsupervised objective functions (PUL-up) and by downsampling images to the intermediate resolution (PUL-down), our pyramid distillation loss without masking out occlusion regions (PDL w/o occ) and our pyramid distillation loss (PDL). All the intermediate output flows are evaluated on the train set of Sintel Clean and Final.

may not be favorable to the unsupervised learning of optical flow especially for the error-prone occlusion regions; (2) our SGU can capture detail matching information by learning from the alignment features which are used to compute optical flow by the decoder.

In the last three rows of Table. 3, we also compare our SGU with its variants: (1) SGU-FM, where the interpolation flow and interpolation map are both removed so that the upsampled flow is directly produced by the dense block without warping and fusion in Fig. 4; (2) SGU-M, where the interpolation map is disabled. Although the performance of SGU-FM is slightly better than the baseline bilinear method, it is poor than that of SGU-M, which demonstrates that using an interpolation flow to solve the interpolation blur is more effective than directly learning to predict a new optical flow. Moreover, the performance reduced as the interpolation map is removed from SGU, which demonstrates the effectiveness of the interpolation map.

Pyramid Distillation Loss. We compare our PDL with different pyramid losses in Table. 4, where 'w/o PL' means no pyramid loss is calculated, 'PUL-up' and 'PUL-down' represent the pyramid unsupervised loss by upsampling intermediate flows to the image resolution and by downsamling the images to the intermediate resolutions accordingly. 'PDL w/o occ' means the occlusion masks on pyramid levels are disabled in our PDL. In 'PUL-up' and 'PUL-down', the photometric loss, smooth loss, census loss and boundary dilated warping loss are used for each pyramid level and their weights are tuned to our best in the experiments. To eliminate variables, the occlusion masks used in 'PUL-up' and 'PUL-down' are calculated by the same method as in our PDL. As a result, model trained by our pyramid distillation loss can generate better results on each pyramid level than by pyramid unsupervised losses. This is because our pseudo labels can provide better supervision on low resolutions than unsupervised losses. Moreover, the error increased when the occlusion mask is disabled in our PDL, indicating that excluding the noisy occlusion regions can improve the quality of the pseudo labels.

5. Conclusion

We have proposed a novel framework for unsupervised learning of optical flow estimation by bottom-up and top-down optimize of the pyramid levels. For the interpolation problem in the bottom-up upsampling process of pyramid network, we proposed a self-guided upsample module to change the interpolation mechanism. For the top-down guidance of the pyramid network, we proposed a pyramid distillation loss to improve the optical flow learning on intermediate levels of the network. Extensive experiments have shown that our method can produce high-quality optical flow results, which outperform all the previous unsupervised methods on multiple leading benchmarks.

Acknowledgement: This research was supported by National Natural Science Foundation of China (NSFC) under grants No.61872067 and No.61720106004.

References

[1] Aria Ahmadi and Ioannis Patras. Unsupervised convolutional neural networks for motion estimation. In *Proc. ICIP*, 2016.

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

[3] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *Proc. ICCV*, pages 2593–2602, 2017.

[4] Thomas Brox, Andres Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow etimation based on a theory for warping. In *Proc. ECCV*, 2004.

[5] Daniel Butler, Jonas Wulff, Garrett Stanley, and Michael Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, pages 611–625, 2012.

[6] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, pages 2758–2766, 2015.

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

[8] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *European conference on computer vision*, pages 1–14. Springer, 2010.

[9] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proc. CVPR*, pages 8981–8989, 2018.

[10] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proc. CVPR*, pages 5747–5756, 2019.

[11] 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 *Proc. CVPR*, pages 1647–1655, 2017.

[12] Woobin Im, Tae-Kyun Kim, and Sung-Eui Yoon. Unsupervised learning of optical flow with deep feature similarity. In *Proc. ECCV*, 2020.

[13] Joel Janai, Fatma Güney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proc. ECCV*, pages 713–731, 2018.

[14] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR*, 2018.

[15] Rico Jonschkowski, Austin Stone, Jonathan T Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. *arXiv preprint arXiv:2006.04902*, 2020.

[16] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graphics*, 26(3):96–es, 2007.

[17] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep joint image filtering. In *Proc. ECCV*, pages 154–169. Springer, 2016.

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

[19] Liang Liu, Guangyao Zhai, Wenlong Ye, and Yong Liu. Unsupervised learning of scene flow estimation fusing with local rigidity. In *Proc. IJCAI*, 2019.

[20] Liang Liu, Jiangning Zhang, Ruifei He, Yong Liu, Yabiao Wang, Ying Tai, Donghao Luo, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *Proc. CVPR*, pages 6489–6498, 2020.

[21] Pengpeng Liu, Irwin King, Michael Lyu, and Jia Xu. Ddflow:learning optical flow with unlabeled data distillation. In *Proc. AAAI*, pages 8770–8777, 2019.

[22] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. Selflow:self-supervised learning of optical flow. In *Proc. CVPR*, pages 4571–4580, 2019.

[23] Kunming Luo, Chuan Wang, Nianjin Ye, Shuaicheng Liu, and Jue Wang. Occinpflow: Occlusion-inpainting optical flow estimation by unsupervised learning. *arXiv preprint arXiv:2006.16637*, 2020.

[24] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss, 2017.

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

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

[27] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proc. CVPR*, pages 12240–12249, 2019.

[28] 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*, 29:9113–9124, 2020.

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

[30] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proc. CVPR*, pages 1164–1172, 2015.

[31] K Simonyan and A Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. NeurIPS*, 2014.

[32] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proc. CVPR*, pages 11166–11175, 2019.

[33] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Proc. CVPR*, pages 2432–2439, 2010.

[34] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proc. CVPR*, pages 8934–8943, 2018.

[35] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. ECCV*, pages 402–419, 2020.

[36] 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*, 29:8429–8442, 2020.

[37] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *Proc. CVPR*, pages 4884–4893, 2018.

[38] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *Proc. CVPR*, pages 1838–1847, 2018.

[39] Jia Xu, Rene Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *Proc. CVPR*, pages 5807–5815, 2017.

[40] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Proc. NeurIPS*, pages 794–805, 2019.

[41] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proc. CVPR*, pages 1983–1992, 2018.

[42] Jason Yu, Adam Harley, and Konstantions Derpanis. Back to basics:unsupervised learning of optical flow via brightness constancy and motion smoothness. In *Proc. ECCV Workshops*, pages 3–10, 2016.

[43] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I-Chao Chang, and Yan Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proc. CVPR*, 2020.

[44] Yiran Zhong, Pan Ji, Jianyuan Wang, Yuchao Dai, and Hongdong Li. Unsupervised deep epipolar flow for stationary or dynamic scenes. In *Proc. CVPR*, pages 12095–12104, 2019.

[45] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proc. ECCV*, pages 38–55, 2018.