This WACV 2023 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

DCVNet: Dilated Cost Volume Networks for Fast Optical Flow

Huaizu Jiang
Northeastern University
Boston, MA 02115
h.jiang@northeastern.edu

Erik Learned-Miller
UMass Amherst
Amherst, MA 01003
elm@cs.umass.edu

Abstract

The cost volume, capturing the similarity of possible correspondences across two input images, is a key ingredient in state-of-the-art optical flow approaches. When sampling correspondences to build the cost volume, a large neighborhood radius is required to deal with large displacements, introducing a significant computational burden. To address this, coarse-to-fine or recurrent processing of the cost volume is usually adopted, where correspondence sampling in a local neighborhood with a small radius suffices. In this paper, we propose an alternative by constructing cost volumes with different dilation factors to capture small and large displacements simultaneously. A U-Net with sikp connections is employed to convert the dilated cost volumes into interpolation weights between all possible captured displacements to get the optical flow. Our proposed model DCVNet only needs to process the cost volume once in a simple feedforward manner and does not rely on the sequential processing strategy. DCVNet obtains comparable accuracy to existing approaches and achieves real-time inference (30 fps on a mid-end 1080ti GPU).

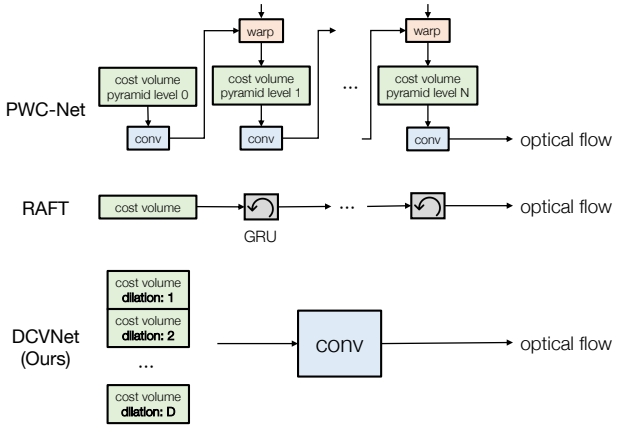


Figure 1. **Illustrations of our proposed model, DCVNet, compared with two representative existing approaches.** DCVNet is an alternative to existing approaches, which does not need the sequential processing of the cost volume. The key idea is to construct cost volumes with different dilation rates to capture small and large displacement at the same time. It achieves real-time inference on a mid-end 1080ti GPU (30 fps) and comparable accuracy to existing approaches.

1. Introduction

Optical flow, as a dense matching problem, is about estimating every single pixel's displacement between two consecutive video frames, capturing the motion of brightness patterns. It is a classical and long-studied problem in computer vision, dating back to the early 1980s [8]. Optical flow has applications in a wide range of other problems, such as scene flow estimation [23], action recognition [25], and video editing and synthesis [3].

Like many other computer vision problems, state-of-the-art approaches for optical flow estimation are all based on deep neural networks. In the beginning, however, deep neural networks for optical flow reported inferior results compared to those of traditional well-engineered energy minimization approaches [6]. To reduce the performance gap, one approach stacked multiple networks to increase capacity [12], resulting in significant accuracy improvements. Increasing the capacity of the network, however, led to huge networks and slower inference. Starting from [24], more and more classical principles derived from traditional optical flow estimation approaches were incorporated into neural network design, allowing deep neural networks to surpass traditional approaches in accuracy. In particular, the cost volume, which is a more discriminative representation for optical flow compared to concatenated feature representations of two images, is now an essential component for state-of-the-art approaches.

To build the cost volume, we need to sample pairs of locations between two input images in a neighborhood along both horizontal and vertical directions to compute their similarity (or cost). A big neighborhood is required to capture large displacements, but it leads to a very large cost volume and a significant computational burden. Conse-

quently, most existing models capture large displacements in a sequential manner by using cost volumes with small neighborhoods for computational efficiency. Specifically, the coarse-to-fine strategy is widely adopted in state-of-the-art advances [29, 11, 36, 32], where a pyramid based on the feature hierarchy of deep Convolutional Neural Networks (CNNs) is built. In each pyramid level, the optical flow estimation in the previous level is used to construct the cost volume with the warping operation. Although a full-range cost volume is constructed in the recent ground-breaking work [30], a Recurrent Neural Network (RNN) is employed to process only a *partial* cost volume with a small neighborhood at each recurrence to capture large displacements sequentially.

In this paper, we propose an alternative to these prevailing approaches, which does not use the sequential estimation strategy to process the cost volume. Instead, cost volumes with different dilation rates are constructed at the same time. Even with a small search neighborhood, both small and large displacements can be captured simultaneously. By concatenating such cost volumes together, a simple convolutional network (a U-Net) is then employed to process the dilated cost volumes only once to obtain the optical flow. Specifically, we estimate the interpolation weights between all possible displacements captured in the dilated cost volumes to get the optical flow. In addition to computing the loss for the optical flow, we also design a loss term to better supervise the interpolation weights, leading to better accuracy.

Compared with other approaches, such as PWC-Net [29] and RAFT [30], our approach is conceptually simpler, and does not require sequential processing of the cost volumes, as shown in Fig. 1. While obtaining comparable error rates on standard benchmarks, our approach runs significantly faster at inference time, achieving 30 fps (frames per second) for a Sintel-resolution image (with a size of 1024×436) on a mid-end 1080ti GPU. Our code as well as the model weights will be made publicly available.

2. Related Work

In this section, we discuss previous optical flow methods. Due to space limits, we focus on neural network-based approaches.

FlowNet, proposed in [6], has two variants, FlowNetS and FlowNetC, both of which have an encoder-decoder structure. FlowNetS simply *concatenates* the feature representations of the two images obtained from the encoder and lets the decoder learn how to compute optical flow. In contrast, FlowNetC constructs a cost volume by computing matching costs (or similarity) between two feature maps. To improve the accuracy, especially for large displacements, FlowNet2 [12] concatenates the FlowNetS and FlowNetC variants in a cascade, where the optical flow estimation is progressively refined. This is the first instance where a neural network reports better or on-par optical flow results with classical engineered approaches.

Although FlowNet2 achieves good accuracy, it has 162M parameters. The more compact SpyNet is proposed in [24]. It computes flow in a coarse-to-fine manner by using a *pyramid* structure borrowed from classical approaches. PWC-Net [29] extends the pyramid structure used in SpyNet. In each pyramid level, a cost volume is built by *warping* the second image's feature map using the optical flow estimation in the previous level. As a result, large displacements can be captured in a sequential manner. A similar coarse-to-fine strategy is used in other approaches. LiteFlowNet [11] also uses a pyramid structure to estimate optical flow in a cascade manner and proposes a flow regularization layer. In the recent extension LiteFlowNet3 [10], an adaptive modulation prior is added to the cost volume, and local flow consistency is used to improve the final accuracy. HD^3 [37] converts optical flow estimation into discrete distribution decomposition. SENSE [15] extends PWC-Net to solve optical flow and stereo disparity at the same time with a shared encoder. In [36], a separable 4D convolution is proposed to process the cost volume, which is converted into two successive 3D convolutions. In [32], 2D convolutions are independently applied to each sampled displacement in the cost volume for displacement-invariant cost learning (DICL). In [35], optical flow estimation is modeled as a global matching problem by computing the similarities of features, where iterative refinement shows improved accuracy.

Instead of using a feature hierarchy for coarse-to-fine estimation, an RNN is used in RAFT [30]. It builds a full-range cost volume capturing the similarity between all pairs of locations between two images. But at each recurrence step, only a partial cost volume in a small neighborhood is used to estimate an offset. This offset is used to move the estimated optical flow (displacement) iteratively closer to the optimum. In [16], sparse matches instead of a full cost volume is used to reduce the memory consumption. In a similar effort [34], 1D attention and correlation is used so that the RAFT can be used for high-resolution images. Zhang *et al*. propose to use a separable cost volume module using non-local aggregation layers to reduce motion ambiguity [38]. Kernel patch attention is used to better use the local affinity to implicitly enfore the smoothness constratint [21]. DIP [41] uses an new inverse propagation inspired by the classical PatchMatch algorithm to better estimate the cost volume. CRAFT proposes to replace the dot-product correlations with transformer cross-frame attention [27]. In [40], argmax is applied on the 4D cost volume to efficiency compute the global matching to better capture large displacement. In [2], deep equilibrium (DEQ) flow estimators are proposed to replace the RNN.

Such sequential estimation approaches are inherently slow as optical flow estimation at each pyramid level or recurrence step is dependent on the results in the previous one. In contrast, Unlike these coarse-to-fine or recurrence-based approaches, we build cost volumes with different dilation factors to effectively capture small and large displacements simultaneously. Consequently, our approach does not need the sequential estimation strategy.

There are other approaches, whose efforts are complementary to ours. A set of improvements about model training protocols, including the data sampling process, model regularization, and data augmentation, are presented in ScopeFlow [4]. A learnable cost volume is proposed in [33], which considers the effectiveness of different feature channels by assigning different weights to different channels. Sun *et al*. propose to learn to generate training data to train optical flow models [28]. Depth learned using a monocular depth estimation model is used to generate optical flow with a virtual camera in [1]. Detail-preserving residual feature pyramid modules are proposed in, which retains important details in the feature maps to better compute the cost volume [19]. Self-supervised consistency loss are proposed in [14] to improve an optical flow model's accuracy.

Using dilations in cost volumes is not completely new. FlowNetC [6] only uses a single dilation factor of 2, which does not fully exploit the potential of using dilations to capture large displacements. In Devon [20], dilated cost volumes are used as a replacement for the warping modules in a sequential coarse-to-fine estimation model. By sharp contrast to FlowNetC [6], we use multiple dilation factors to better capture small and large displacements. Additionally, unlike Devon [20], we use dilated cost volumes as an alternative for the sequential estimation strategy to compute optical flow. Moreover, our model achieves significantly better accuracy than both FlowNetC [6] and Devon [20].

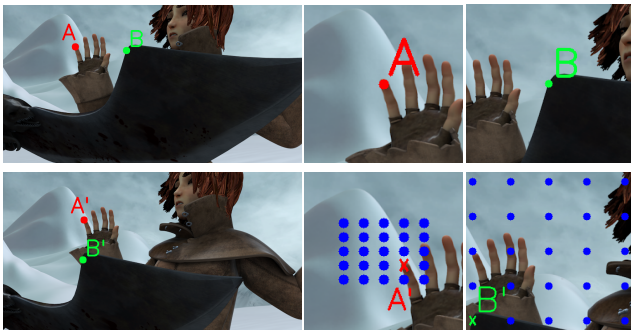
3. Dilated Cost Volume Networks

3.1. Dilated Cost Volumes

Given two input images I_1 and I_2 with height H and width W, we extract their L_2-normed feature representations $\mathbf{x}_1^s$ and $\mathbf{x}_2^s$ at stride s using a CNN, where s corresponds to the spatial resolution downsample factor w.r.t. the input images. To search for the correct correspondence for a position $p = (x, y)$ in $\mathbf{x}_1^s$, we need to compare its feature vector with that of a candidate position $q = (x + u, y + v)$ in $\mathbf{x}_2^s$, where u and v are the offsets of the pixel from p to q. To measure the similarity between feature vectors at p and q, we have

$$\mathbf{c}^s(u, v, x, y) = f\left(\mathbf{x}_1^s(x, y), \mathbf{x}_2^s(x + u, y + v)\right), \quad (1)$$

where $f(\cdot, \cdot)$ is a function measuring the similarity between two feature vectors. Here we divide each of the vectors $\mathbf{x}_1^s$ and $\mathbf{x}_2^s$ into C sub-vectors and compute the cosine similarity between each pair of corresponding sub-vectors. The output of f therefore has C dimensions. As we need to sample in a local 2D neighborhood for all possible correspondences, we have $u \in [-k, k]$ and $v \in [-k, k]$, where k is the neighborhood radius. The full cost volume size is thus $C \times U \times V \times \frac{H}{s} \times \frac{W}{s}$, where $U = V = 2k + 1$.



(a) input images (b) dilation=1 (c) dilation=5

Figure 2. **Illustration of using dilation to capture both small and large displacements.** (a) input two images where points A and B move to A' and B', respectively. (b) two patches around A in two images. (c) two patches around B in two images. Blue dots in (b) and (c) correspond to candidate displacements when constructing cost volumes. With a small search radius (2 in this example), correct displacements (denoted by red and blue crosses, respectively) can be captured using two different dilation factors. Best viewed in color.

Due to the striding factor, such a cost volume $\mathbf{c}^s$ captures candidate horizontal displacements[1] across two input images in the range of $s \odot [-k, k]$, where $\odot$ denotes the elementwise multiplication between a scalar and a vector. For simplicity, we use only the horizontal displacement for illustration here (vertical displacement can be analyzed similarly). To account for large displacements, which is critical for accurate optical flow estimation, either a larger stride s or neighborhood radius k can be used. Both of them are problematic, however. A larger stride means more downsampling and loss of spatial resolution. A large neighborhood radius, on the other hand, results in a large cost volume and heavy computation.

Instead, we propose to use dilation factors to construct cost volumes to deal with small and large displacement at the same time. Specifically, we have

$$\mathbf{c}^{s,d}(u^d, v^d, x, y) = f\left(\mathbf{x}_1^s(x, y), \mathbf{x}_2^s(x + u^d, y + v^d)\right), \quad \text{where } u^d \in d \odot [-k, k], v^d \in d \odot [-k, k]. \quad (2)$$

Here d is a dilation factor. Now the search range of displacement over two input images is $sd \odot [-k, k]$. In this

[1] In this paper, we use "displacement" to denote a pixel's offset over two input images and "correspondence" to refer to offsets over two feature maps. So a displacement is the multiplication of a correspondence by the stride of the feature map.

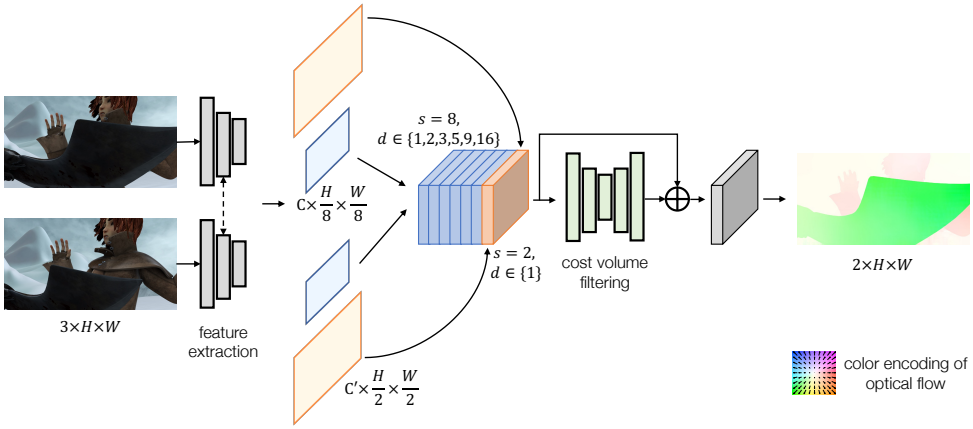


Figure 3. **Pipeline of DCVNet.** Feature representations of two input images are obtained from the encoder, which are used to construct the dilated cost volumes at different strides and dilation rates. A U-Net is employed to process the cost volumes to produce a set of interpolation weights over the captured displacements in the cost volume to compute the optical flow.

Table 1. **Displacements over input images** captured using different strides and dilation factors.

stride (s)	dilation (d)	candidate horizontal displacements
2	1	{-8, -6, -4, -2, 0, 2, 4, 6, 8}
8	1	{-32, -24, -16, 8, 0, 8, 16, 24, 32}
8	5	{-160, -120, -80, -40, 0, 40, 80, 120, 160}
8	9	{-288, -216, -144, -72, 0, 72, 144, 216, 288}
8	16	{-512, -384, -256, -128, 0, 128, 256, 384, 512}

way, we can capture large displacements by having a large d while maintaining small k and s, which preserves both computational efficiency and spatial resolution for the cost volume. Fig. 2 illustrates how dilation helps capture both small and large displacements with a small neighborhood radius. Specifically, in this paper, we consider $s = 8$ and $d \in \{1, 2, 3, 5, 9, 16\}$. As we can see in Table 1, a displacement as large as 512 pixels can be captured using a dilation factor $d = 16$, stride $s = 8$, and neighborhood radius $k = 4$.

As the dilation factors increase, the gap between candidate displacements also gets larger. To deal with this issue, we also build a cost volume with $s = 2$ and $d = 1$ to capture small and fine displacement. We do spatial sampling of 4 to make the spatial resolution compatible with the cost volumes constructed over the stride of 8. Finally, we concatenate all cost volumes over different strides and dilation factors. Our final cost volume has a dimension of $C' \times \frac{H}{8} \times \frac{W}{8}$, where $C' = D \times C \times U \times V$ and D is the total number of dilation factors ($D = 7$ in our case).

3.2. Cost Volume Filtering for Optical Flow

So far, we have introduced our dilated cost volumes. How can we translate such cost volumes into exact pixel-wise displacements, *i.e.*, optical flow? Instead of directly *regressing* optical flow values, we do *interpolation* between all possible displacements similar to [36, 32]. Specifically, we have

$$\mathbf{f} = \sum_{i,s,d} \omega_{i,s,d} \mathbf{f}_{i,s,d}, \tag{3}$$

where $\sum_{i,s,d} \omega_{i,s,d} = 1$ and $\omega_{i,s,d} \geq 0$. $\mathbf{f}_{i,s,d} = (\mu_{i,s,d}, \nu_{i,s,d})$ is a single 2D displacement at stride s with

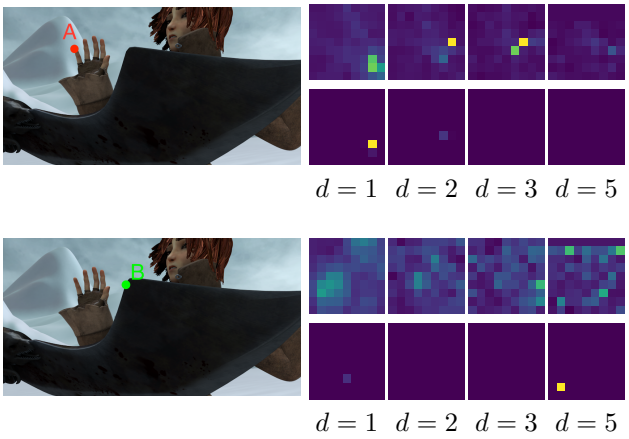


Figure 4. **Illustration of interpolation weights.** For both points A and B, in the right, we show the interpolation weights obtained with and without the U-Net filtering on the top and bottom, respectively. Each image represents $U \times V$ (9×9) interpolation weights. The feature stride is 8 and different dilation factors are shown in the bottom. We can see that for the point A, whose motion magnitude is small, a small dilation factor is sufficient to capture the correspondence. While for the point B, whose motion magnitude is large, a large dilation factor is needed. (Color encoding: blue is close to 0 and yellow is close to 1. Best viewed in color.)

dilation d, where $\mu_{i,s,d} \in sd\odot[-k,k], \nu_{i,s,d} \in sd\odot[-k,k]$. At a particular stride, for each dilation factor, there are UV such sampled displacements in a cost volume. To obtain the interpolation weights, we use a U-Net taking our dilated cost volumes as input to estimate $\omega_{i,s,d}$, where a skip connection from the cost volume to the output is added. A `softmax` is performed on top of the U-Net's output to ensure that the constraints on $\omega_{i,s,d}$ are satisfied.

The architecture of our proposed dilated cost volume network (DCVNet) is illustrated in Fig. 3. We use a feature encoder similar to that used in [30], except we only use a single residual block in the stride of 2, with Instance Normalization [31] layers to extract features of input images to construct the cost volume. We empirically found that having another context encoder is not significantly helpful and yet substantially increases the number of parameters. There are no normalization layers for the rest of the DCVNet. We use Leaky ReLUs with a slope of 0.1 for the entire network. We use the same convex upsampling strategy used in [30] to upsample estimated optical flow to the input's resolution. We provide more details of the network architecture in the supplementary material.

3.3. Loss Function

Denote the estimated optical flow before and after the upsampling as $\hat{\mathbf{f}}'$ and $\hat{\mathbf{f}}$, respectively, and the ground-truth as $\mathbf{f}$. We use the `L1` loss to supervise the network training.

$$\mathcal{L}_f = \alpha||\hat{\mathbf{f}}' - \mathbf{f}'||_1 + ||\hat{\mathbf{f}} - \mathbf{f}||_1, \tag{4}$$

where $\mathbf{f}'$ is the downsampled ground truth of $\mathbf{f}$, which has the same resolution as $\hat{\mathbf{f}}'$. α is empirically set as 0.25.

At the same time, we found that adding extra constraints to the interpolation weights $\omega_{i,s,d}$ leads to better results. Note there are many plausible solutions of $\omega_{i,s,d}$ that yield the same optical flow. To add constraints of the interpolation weights for each pixel, we compute the bilinear interpolation weights over the four nearest displacement vectors surrounding the ground-truth optical flow value. We use the `CrossEntropy` loss between the estimated $\hat{\omega}_{i,s,d}$ and the generated ground-truth $\omega_{i,s,d}$.

$$\mathcal{L}_\omega = -\sum_{i,s,d} \omega_{i,s,d} \log \hat{\omega}_{i,s,d}. \tag{5}$$

The final loss is defined as

$$\mathcal{L} = \mathcal{L}_f + \beta\mathcal{L}_\omega, \tag{6}$$

where β balances $\mathcal{L}_f$ and $\mathcal{L}_\omega$. We empirically found that it leads to better accuracy by annealing β using a cosine schedule, where the initial value is 1 and the final value is 0. We hypothesize that at the beginning of the training, adding the strong prior using the bilinear interpolation weights to $\omega_{i,s,d}$ helps the training but it becomes less effective as training goes on.

Table 2. **Average EPE results on MPI Sintel optical flow dataset.** "-ft" means fine-tuning on the MPI Sintel *training* set. The numbers in parentheses are results on the data the methods have been fine-tuned on. They are not directly comparable and put here for completeness. † indicates a model uses extra training data.

Methods	Training		Test		Time	#Para
	Clean	Final	Clean	Final	(s)	(M)
FlowNet2 [12]	2.02	3.14	3.96	6.02	0.12	162
PWC-Net [29]	2.55	3.93	-	-	0.04	8.8
LiteFlowNet [11]	2.48	4.04	-	-	0.07	5.4
LiteFlowNet2	2.24	3.78	-	-	**0.03**	-
FlowNet3 [13]	2.08	3.94	3.61	6.03	0.07	117
HD^3 [37]	3.84	8.77	-	-	0.14	38.6
SENSE [15]	1.91	3.78	-	-	0.04	8.3
VCN [36]	2.21	3.68	-	-	0.26	6.2
MaskFlow [39]	2.25	3.61	-	-	-	-
Devon [20]	2.45	3.72	-	-	0.04	-
DICL [32]	1.94	3.77	-	-	0.08	9.8
RAFT-small [30]	2.21	3.35	-	-	0.05	**1.0**
RAFT [30]	**1.43**	**2.71**	-	-	0.3	5.3
Ours	1.91	3.28	-	-	**0.03**	7.9
FlowNetS-ft [6]	(3.66)	(4.44)	6.96	7.52	0.02	38.7
FlowNetC-ft [6]	(3.50)	(3.89)	6.85	8.51	0.03	39.1
SpyNet-ft [24]	(3.17)	(4.32)	6.64	8.36	0.16	1.2
FlowNet2-ft [12]	(1.45)	(2.01)	4.16	5.74	0.12	162
PWC-Net-ft [29]	(1.70)	(2.21)	3.86	5.13	0.04	8.8
LiteFlowNet-ft [11]	(1.45)	(1.78)	4.54	5.38	0.07	5.4
LiteFlowNet2-ft [9]	(1.30)	(1.62)	3.48	4.69	0.03	-
LiteFlowNet3-ft [10]	(1.32)	(1.76)	2.99	4.45	0.05	5.2
FlowNet3-ft [13]	(1.47)	(2.12)	4.35	5.67	0.07	117
HD^3-ft [37]	(1.87)	(1.17)	4.79	4.67	0.14	38.6
SENSE-ft [15]	(1.54)	(2.05)	3.60	4.86	0.04	8.3
VCN-ft† [36]	(1.66)	(2.24)	2.81	4.40	0.26	6.2
MaskFlow-ft† [39]	-	-	2.52	4.17	-	-
Devon-ft [20]	(1.97)	(2.67)	4.34	6.35	0.04	-
DICL-ft [32]	(1.11)	(1.60)	2.12	3.44	0.08	9.8
RAFT-ft† [30]	(0.77)	(1.27)	**1.61**	**2.86**	0.3	5.3
Ours-ft†	(1.04)	(1.37)	2.36	3.66	**0.03**	7.9

4. Experiments

4.1. Implementation Details

Pre-training. We train our model on the synthetic SceneFlow dataset [22] following [15], which consists of FlyingThings3D, Driving, and Monkaa. We found using FlyingChairs [6] and FlyingThings leads to worse results for our model. Only optical flow annotations are used for training. Interestingly, such a pre-training results in worse results for RAFT (3.16 vs 2.71 in terms of average end-point-error on the final pass of the MPI-Sintel training set [5]).

During training, we closely follow the setting used in RAFT [30]. Specifically, we use extensive data augmentations including color jittering, random crops, random resizing, and random horizontal and vertical flips. The crop size is 400×720. The DCVNet is trained for 800K iterations with a batch size of 8 using the AdamW optimizer [17]. The initial learning rate is 0.0002 and is updated following the OneCycle learning rate schedule [26] with a linear anneal-

Table 3. **Results on the KITTI optical flow dataset.** "-ft" means fine-tuning on the KITTI *training* set and the numbers in the parenthesis are results on the data the methods have been fine-tuned on.

Methods	KITTI 2012			KITTI 2015		
	AEPE *train*	AEPE *test*	Fl-Noc *test*	AEPE *train*	Fl-all *train*	Fl-all *test*
FlowNet2 [12]	4.09	-	-	10.06	30.37%	-
PWC-Net [29]	4.14	-	-	10.35	33.67%	-
FlowNet3 [13]	3.69	-	-	9.33	-	-
HD^3 [37]	4.65	-	-	13.17	24.9%	-
SENSE [15]	2.55	-	-	6.23	23.29%	-
VCN [36]	-	-	-	8.36	25.1%	-
MaskFlow [39]	-	-	-	-	23.1%	-
Devon [20]	4.73	-	-	10.65	-	-
DICL [32]	-	-	-	8.70	23.60%	-
RAFT-small [30]	-	-	-	7.51	26.91%	-
RAFT [30]	-	-	-	5.04	**17.40%**	-
Ours	**2.56**	-	-	**4.83**	23.68%	-
SpyNet-ft [24]	(4.13)	4.7	12.31%	-	-	35.07%
FlowNet2-ft [12]	(1.28)	1.8	4.82%	(2.30)	(8.61%)	10.41 %
PWC-Net-ft [29]	(1.45)	1.7	4.22%	(2.16)	(9.80%)	9.60%
LiteFlowNet-ft [11]	(1.26)	1.7	-	(2.16)	(8.16%)	10.24 %
LiteFlowNet3-ft [11]	(0.91)	**1.3**	2.51%	(1.26)	(3.82%)	7.34 %
FlowNet3-ft [13]	(1.19)	-	3.45%	(1.79)	-	8.60%
HD^3-ft [37]	(0.81)	1.4	**2.26%**	(1.31)	(4.10%)	6.55%
SENSE-ft [15]	(1.18)	1.5	3.03%	(2.05)	(9.69%)	8.16%
VCN-ft [36]	-	-	-	(1.16)	(4.10%)	6.30%
MaskFlow-ft [39]	-	-	-	-	-	6.10%
Devon-ft [20]	(1.29)	2.6	-	(2.00)	-	14.31%
DICL-ft [32]	-	-	-	(1.02)	(3.60%)	6.31%
RAFT-ft [30]	-	-	-	(0.63)	(1.50%)	**5.10%**
Ours-ft	(0.94)	1.6	5.33%	(1.22)	(4.41%)	9.62%

ing strategy and a warmup factor of 0.05. We also perform gradient norm clip with a value of 1.

Fine-tuning. For Sintel, we fine-tune the pre-trained model using both *final* and *clean* passes. Following previous works, we optionally use extra data from KITTI2015 [7] and HD1K [18] for training. The model is trained for 400K with a batch size of 8. The initial learning rate is set to be 0.000125 and updated following the same OneCycle scheduler as we use in the pre-training stage. For KITTI, we train the model for 400K iterations with a batch size of 8. The initial learning rate is 0.0001 and the OneCyle learning rate scheduler is used. For both Sintel and KITTI, we perform similar data augmentations used in the pre-training stage. The crop sizes for Sintel and KITTI are 368×768 and 336×944, respectively.

4.2. Main Results

Optical flow results. Quantitative results on the MPI Sintel and KITTI benchmark datasets of different neural network-based approaches are summarized in Table 2 and Table 3, respectively. We can see that our approach compares favorably to other approaches before and after fine-tuning. Specifically, on the more photo-realistic final pass with factors such as lighting condition changes, shadow effects, motion blur, etc, our proposed model, DCVNet, achieves on-par end-point-error (EPE) with state-of-the-art approaches like DICL [32] and RAFT [30]. Particularly, our model outperforms Devon [20], which uses dilated cost volumes as a replacement of the warping module in a sequential cost-to-fine optical flow model.

Table 4. **Number of parameters, GPU memory consumption, and inference speed** of different optical flow models, measured on Sintel.

	PWC-Net	VCN	DICL	RAFT	Ours
#Para. (M)	9.37	6.23	9.78	**5.26**	7.87
#Mem. (GB)	**1.11**	2.33	2.78	1.37	1.16
Speed (fps)	25	3.8	12.5	3.3	**30**

We show some visual results of estimated optical flow from different approaches in Fig. 5. We can see that our approach DCVNet can capture the motion of challenging scenes, leading to visually appealing results akin to others. Particularly, for the bamboo images, our approach produces sharper motion boundaries and smoother motion estimates in the background compared to both PWCNet [29] and VCN [36]. We refer readers to the supplementary materials for more visual results.

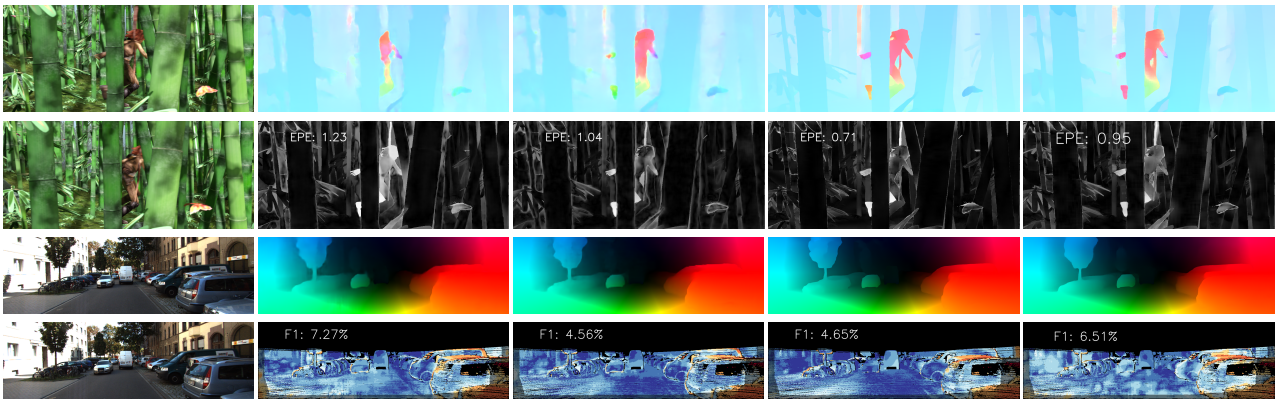
Model size and memory. Our model achieves reasonable model compactness and memory consumption, as shown in Table 4, compared to other state-of-the-art approaches. For GPU memory, our DCVNet needs 1.16GB, which is lower than RAFT, DICL, and VCN.

Inference speed. Compared to existing approaches, our DCVNet runs significantly faster, meeting the real-time inference requirement. On a mid-end 1080ti GPU, our approach takes only 33ms to process two RGB images from the Sintel dataset (with a resolution of 1024×436), running at 30 fps. We use a CUDA implementation for the cost volume construction, which takes 10ms. Most of the time is spent on the feature extraction part that takes 14ms. The decoder part to convert the cost volume to optical flow needs 9ms.

4.3. Ablation Studies

To validate the effectiveness of dilated cost volumes and the loss term for the interpolation weights, we perform ablation studies. We train the models on the SceneFlow dataset.

Effectiveness of dilation. We first investigate the effectiveness of using dilation for the cost volume. We vary the number of dilation rates from 7 to 1. To maintain the capability of capturing large displacement, we keep the largest stride and dilation rate and gradually remove smaller ones. We report error rate vs. number of dilation rates in Fig. 6. We can clearly see that the error rates steadily decrease on both MPI Sintel and KITTI 2015 datasets as number of dilation increases. It validates our core idea of constructing cost



(a) input images (b) PWCNet [29] (c) VCN [36] (d) RAFT [30] (e) our DCVNet

Figure 5. **Visual comparison of optical flow estimations**. From left to right: (a) input images, (b) PWCNet [29], (c) VCN [36], (d) RAFT [30], and (e) our DCVNet. For each method, we show colorized optical flow and error maps (obtained from online servers). For the error maps, white and red indicate large error while black and blue mean small error. Best viewed in color.

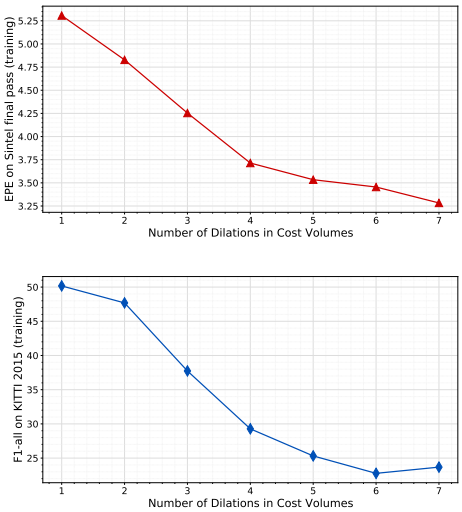


Figure 6. **Effectiveness of dilation for cost volumes.** Top: EPE vs. number of dilation rates on the *final* pass of the MPI Sintel training set. Bottom: F1-all error rate vs. number of dilation rates on the KITTI2015 training set.

volumes with dilations to deal with both small and large displacements simultaneously.

Supervision of the interpolation weights. We investigate the effectiveness of the loss term $\mathcal{L}_\omega$. By setting $\beta = 0$, we completely remove the supervision of the interpolation weights. On the other hand, by setting $\beta = 1$, instead of annealed version, we impose strong constraints to the interpolation weights. We can see from Table 5, none of them works better than the annealed β.

Table 5. **Effectiveness of the loss term $\mathcal{L}_\omega$** to supervise the interpolation weights.

	Sintel-clean	Sintel-final	KTTI 2012	KITTI 2015
β=0	1.99	3.47	2.65	23.91%
β=1	**1.91**	3.32	2.61	24.13%
annealed β	**1.91**	**3.28**	**2.56**	**23.68%**

5. Conclusion

In this paper, we presented DCVNet, a dilated cost volume network for optical flow. Our core idea is to use different dilation rates to construct cost volumes to capture both small and large displacements at the same time with a small neighborhood to retain model efficiency. By doing so, our approach no longer relies on the sequential estimation strategy for optical flow, leading to a fast optical flow model. Our approach runs at 30fps on a mid-end 1080ti GPU and achieves comparable accuracy to existing models on standard benchmarks.

References

[1] Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia. Learning optical flow from still images. In *CVPR*, 2021.

[2] Shaojie Bai, Zhengyang Geng, Yash Savani, and J. Zico Kolter. Deep equilibrium optical flow estimation. In *CVPR*, 2022.

[3] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011.

[4] Aviram Bar-Haim and Lior Wolf. Scopeflow: Dynamic scene scoping for optical flow. In *CVPR*, 2020.

[5] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, 2012.

[6] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, Thomas Brox, et al. FlowNet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015.

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

[8] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981.

[9] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow cnn - revisiting data fidelity and regularization. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 2020.

[10] Tak-Wai Hui and Chen Change Loy. Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation. In *ECCV*, 2020.

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

[12] 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*, 2017.

[13] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *Proc. ECCV*, 2018.

[14] Jisoo Jeong, Jamie Menjay Lin, Fatih Porikli, and Nojun Kwak. Imposing consistency for optical flow estimation. In *CVPR*, 2022.

[15] Huaizu Jiang, Deqing Sun, Varun Jampani, Zhaoyang Lv, Erik G. Learned-Miller, and Jan Kautz. SENSE: A shared encoder network for scene-flow estimation. In *ICCV*, 2019.

[16] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning optical flow from a few matches. In *CVPR*, 2021.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[18] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Güssefeld, M. Rahimimoghaddam, S. Hofmann, C. Brenner, and B. Jähne. The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *CVPRW*, 2016.

[19] Libo Long and Jochen Lang. Detail preserving residual feature pyramid modules for optical flow. In *WACV*, 2022.

[20] Yao Lu, Jack Valmadre, Heng Wang, Juho Kannala, Mehrtash Harandi, and Philip Torr. Devon: Deformable volume network for learning optical flow. In *WACV*, March 2020.

[21] Ao Luo, Fan Yang, Xin Li, and Shuaicheng Liu. Learning optical flow with kernel patch attention. In *CVPR*, 2022.

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

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

[24] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proc. CVPR*, 2017.

[25] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

[26] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[27] Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Siow Mong Goh, and Hongyuan Zhu. CRAFT: cross-attentional flow transformer for robust optical flow. In *CVPR*, 2022.

[28] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T. Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *CVPR*, 2021.

[29] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, June 2018.

[30] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[31] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[32] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Kaihao Zhang, Pan Ji, and Hongdong Li. Displacement-invariant matching cost learning for accurate optical flow estimation. In *NeurIPS*, 2020.

[33] Taihong Xiao, Jinwei Yuan, Deqing Sun, Qifei Wang, Xin-Yu Zhang, Kehan Xu, and Ming-Hsuan Yang. Learnable cost volume using the cayley representation. In *ECCV*, 2020.

[34] Haofei Xu, Jiaolong Yang, Jianfei Cai, Juyong Zhang, and Xin Tong. High-resolution optical flow from 1d attention and correlation. In *ICCV*, 2021.

[35] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *CVPR*, 2022.

[36] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *NeurIPS*, 2019.

[37] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *CVPR*, 2019.

[38] Feihu Zhang, Oliver J. Woodford, Victor Prisacariu, and Philip H. S. Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *ICCV*, 2021.

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

[40] Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris N. Metaxas. Global matching with overlapping attention for optical flow estimation. In *CoRR*, 2022.

[41] Zihua Zheng, Ni Nie, Zhi Ling, Pengfei Xiong, Jiangyu Liu, Hao Wang, and Jiankun Li. DIP: deep inverse patchmatch for high-resolution optical flow. In *CVPR*, 2022.