DIP: Deep Inverse Patchmatch for High-Resolution Optical Flow

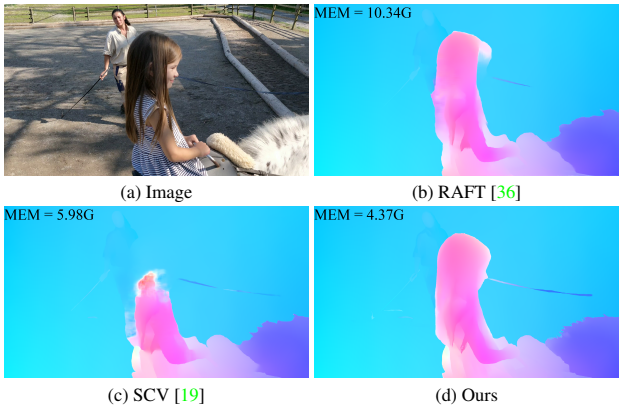
Zihua Zheng[1] Ni Nie[1] Zhi Ling[1] Pengfei Xiong[2] Jiangyu Liu[1] Hao Wang[1] Jiankun Li[1]
[1]Megvii [2]Tencent
{zhengzihua, nieni, lingzhi, liujiangyu, wanghao03, lijiankun}@megvii.com
xiongpengfei2019@gmail.com

Abstract

Recently, the dense correlation volume method achieves state-of-the-art performance in optical flow. However, the correlation volume computation requires a lot of memory, which makes prediction difficult on high-resolution images. In this paper, we propose a novel Patchmatch-based framework to work on high-resolution optical flow estimation. Specifically, we introduce the first end-to-end Patchmatch based deep learning optical flow. It can get high-precision results with lower memory benefiting from propagation and local search of Patchmatch. Furthermore, a new inverse propagation is proposed to decouple the complex operations of propagation, which can significantly reduce calculations in multiple iterations. At the time of submission, our method ranks 1^{st} on all the metrics on the popular KITTI2015 [28] benchmark , and ranks 2^{nd} on EPE on the Sintel [7] clean benchmark among published optical flow methods. Experiment shows our method has a strong cross-dataset generalization ability that the F1-all achieves 13.73%, reducing 21% from the best published result 17.4% on KITTI2015. What's more, our method shows a good details preserving result on the high-resolution dataset DAVIS [1] and consumes 2× less memory than RAFT [36]. Code will be available at github.com/zihuazheng/DIP

1. Introduction

Optical flow, the 2D displacement field that describes apparent motion of brightness patterns between two successive images [13], provides valuable information about the spatial arrangement of the viewed objects and the change rate of the arrangement [39]. Since Horn and Schunck (HS) [13] and Lucas and Kanade (LK) [25] proposed the differential method to calculate optical flow in 1981, many extension algorithms [22, 30, 42] have been proposed. Hence, optical flow has been widely used in various applications such as visual surveillance tasks [43], segmentation [38], action recognition [31], obstacle detection [12] and image sequence super-resolution [26].



(a) Image (b) RAFT [36] (c) SCV [19] (d) Ours

Figure 1. Comparisons on high-resolution (1080 × 1920) images from DAVIS dataset. Compared with RAFT and SCV, our method has achieved better details with lower memory.

Recently, deep learning has made great progress in solving the problem of optical flow. Since FlowNetC [10], many methods have achieved state-of-the-art results. For deep learning, in addition to accuracy, performance and memory are also challenges especially when predicting flow at high-resolution. To reduce complexity of computation and usage of memory, previous approaches [16–18, 34, 46] use coarse-to-fine strategy, they may suffer from low-resolution error recovery problems. In order to maintain high accuracy on large displacements, especially for fast moving small targets, RAFT [36] constructs an all-pairs 4D correlation volume and look up with a convolution GRU block. However, it runs into memory problems when predicting high-resolution optical flow.

In order to reduce the memory while maintaining high accuracy, instead of using the sparse global correlation strategies like [19, 44] which suffer from loss of accuracy, we introduce the idea of Patchmatch to the computation of correlation. Patchmatch implements a random initialization, iterative propagation and search algorithm for approximate nearest neighbor field estimation [5, 6, 14]. It only needs to perform correlation calculations on nearby pixels and propagate its cost information to the next match-

ing point iteratively, without the need to construct a global matching cost. Therefore, the Patchmatch algorithm greatly reduces the memory overhead caused by the correlation volume. Moreover, the iterative propagation and search in Patchmatch can be easily achieved using GRU [36]. To this end, we propose a Patchmatch-based framework for optical flow, which can effectively reduce memory while maintaining high accuracy. It contains two key modules: propagation module and local search module. The propagation module reduces the search radius effectively, and the local search module accelerates convergence and further improves accuracy. At the same time, we have achieved high-resolution predictions of high-precision optical flow through adaptive-layers iterations.

Furthermore, a new inverse propagation method is proposed, which offsets and stacks target patches in advance. Then, it only needs to do warping once for all propagations compared with propagation which requires offset and warping in each propagation, so as to reduce the calculation time significantly.

We demonstrate our approach on the challenging Sintel [7] and KITTI-15 [28] datasets. Our model ranks first on KITTI-15 and second on Sintel-Clean. Fig. 1 shows the results of our Deep Inverse Patchmatch(DIP). Comparing to previous approaches [20, 36], DIP keeps the best effect while memory usage is the lowest. At the same time, our method has a strong cross-dataset generalization that the F1-all achieves 13.73%, reduced 21% from the best published result 17.4% on KITTI2015 [28]. In addition, the supplementary material shows the domain invariance of our DIP in the Stereo field.

To sum up, our main contributions include:

- We design an efficient framework which introduces Patchmatch to the end-to-end optical flow prediction for the first time. It can improve the accuracy of optical flow while reducing the memory of correlation volume.
- We propose a novel inverse propagation module. Compared with propagation, it can effectively reduce calculations while maintaining considerable performance.
- Our experiments demonstrate that the method achieves a good trade-off between performance and memory, a comparable results with the state of the art methods on public datasets and a good generalization on different datasets.

2. Related Work

Deep Flow Methods The first end-to-end CNN-based version for flow estimation can be traced back to [10], which proposed a U-net like architecture FlowNetS to predict flow directly. A correlation layer was included in a diverse version named FlowNetC. In FlowNet2, Ilg *et al*. [18] introduced a warping mechanism and stacked hourglass network to promote the performance on small motion areas. PWC-Net [34] used feature warping and a coarse-to-fine cost volume with a context network for flow refinement, further improving the accuracy and reducing the model size simultaneously. To address ambiguous correspondence and occlusion problem, Hui *et al*. [15] proposed LiteFlowNet3 with adaptive affine transformation and local flow consistency restrictions. RAFT [36] introduced a shared weight iterative refinement module to update the flow field retrieved from a 4D all-pair correlation volume. To reduce the computation complexity of 2D searching in high-resolution images, Xu *et al*. [44] factorized the 2D search to 1D in two directions combined with attention mechanism. Jiang *et al*. [20] proposed to construct a sparse correlation volume directly by computing the k-Nearest matches in one feature map for each feature vector in the other feature map. The memory consumption of them is less compare to RAFT but their accuracy is inferior. Another line of work is focused on joining image segmentation and flow estimation task together [8,9,33,37], which propagated two different complementary features, aiming at improving the performance of flow estimation and vice versa.

Patchmatch Based Methods Patchmatch has been originally proposed by Barnes *et al*. [5]. Its core work is to compute patch correspondences in a pair of images. The key idea behind it is that neighboring pixels usually have coherent matches. M Bleyer *et al*. [6] applied Patchmatch to stereo matching and proposed a slanted support windows method for computing aggregation to obtain sub-pixel disparity precision. In order to reduce the error caused by the motion discontinuity of Patchmatch in optical flow, Bao *et al*. [3] proposed the Edge-Preserving Patchmatch algorithm. Hu *et al*. [14] proposed a Coarse-to-Fine Patchmatch strategy to improve the speed and accuracy of optical flow. In deep learning, Bailer *et al*. [2] regarded Patchmatch as a 2-classification problem and proposed a thresholded loss to improve the accuracy of classification. Shivam *et al*. [11] developed a differentiable Patchmatch module to achieve real-time in the stereo disparity estimation network. But this method is sparse and only works on the disparity dimension. Wang *et al*. [40] introduced iterative multi-scale Patchmatch, which used one adaptive propagation and differentiable warping strategy, achieved a good performance in the Multi-View Stereo problem.

3. Approach

We start with our observation and analysis of different correlation volume in optical flow task. These methods require high memory usage and computation to compute the correlation volume. Inspired by the high efficiency of Patchmatch on the correspondence points matching, we use

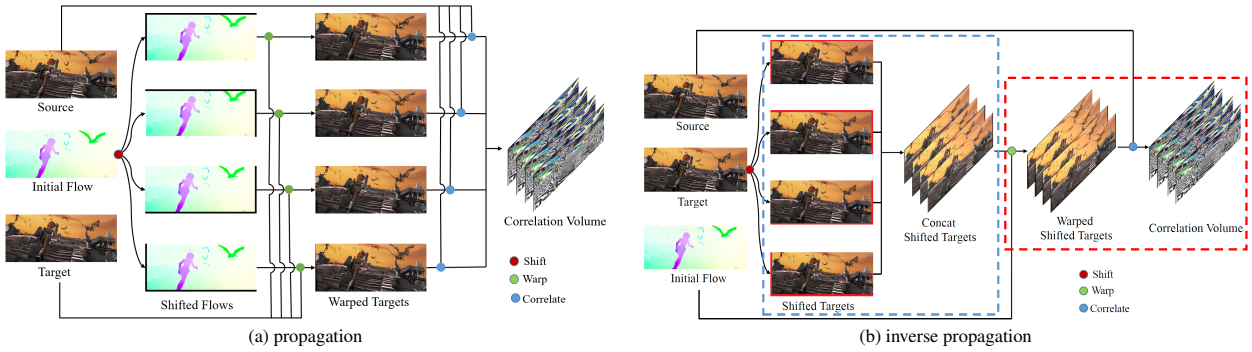


Figure 2. The correlation calculation process of propagation and inverse propagation. Where the red points in the graph represent shift operator on the optical flow or images according to the seed points, the green points represent warping operator on the images according to the optical flow, and the blue points represent correlation caculating operator between the source image and the warped images. The blue box in (b) represents the initialization stage, and the red box represents the running stage.

it to reduce the search space of optical flow.

3.1. Observations

Local Correlation Volume In modern local correlation volume based optical flow approaches [10], the computation of it can be formulated as follows:

$$Corr = \{F_1(\mathbf{x}) \cdot F_2(\mathbf{x}+\mathbf{d}) | \mathbf{x} \in X, \mathbf{d} \in D\}, \quad (1)$$

where F_1 is the source feature map and F_2 is the target feature map, $\mathbf{d}$ is the displacement along the x or y direction. $X = [0, h) \times [0, w)$, $D = [-d_{max}, d_{max}]^2$, h is the height of feature map, w is the width of feature map. So the memory and calculation of the correlation volume are linear to $hw(2d_{max}+1)^2$ and quadratic to the radius of the search space. Limited by the size of the search radius, it is difficult to obtain high-precision optical flow in high-resolution challenging scenes.

Global Correlation Volume Recently, RAFT [36] proposed an all-pairs correlation volume which achieved the state-of-the-art performance. The global correlation computation at location (i, j) in F_1 and location (k, l) in F_2 can be defined as follows:

$$Corr^m_{ijkl} = \frac{1}{2^{2m}} \sum_{p}^{2^m} \sum_{q}^{2^m} \left(F_1(i,j) \cdot F_2(2^m k + p, 2^m l + q)\right), \quad (2)$$

where m is the pyramid layer number. 2^m is the pooled kernel size. Compared with local correlation volume, global correlation volume contains N^2 elements, where $N = hw$. When the h or w of F increases, the memory and calculation will multiply. So the global method suffers from insufficient memory when inferring at high- resolution.

Patchmatch Method Patchmatch is proposed by Barnes *et al.* [5] to find dense correspondences across images for structural editing. The key idea behind it is that we can get some good guesses by a large number of random samples. And based on the locality of image, once a good match is found, the information can be efficiently propagated to its neighbors. So, we propose to use the propagation strategy to reduce the search radius and use local search to further improve accuracy. And the complexity of Patchmatch method is $hw(n + r^2)$, where n is the number of propagation, r is the local search radius, and both values are very small and do not change with the increase of displacement or resolution. Details are described in the next subsection.

3.2. Patchmatch In Flow Problem

The traditional Patchmatch methods [5, 6, 14, 23] has three main components. 1) Random initialization. It gets some good guesses by a large number of random samples. 2) Propagation. Based on image locality, once a good match is found, the information can be efficiently propagated from its neighbors. 3) Random search. It is used in the subsequent propagation to prevent local optimization and make it possible to obtain the good match when no good match exist in its neighbors.

Iterative propagation and search are the key points to solve the flow problem. In propagation stage, we treat a point of feature maps as a patch and select 4 neighbor seed points. So every point can get the flow candidates from its neighbors by shifting the flow map toward the 4 neighbors. Then we can compute a 5 dimension correlation volume based on the neighbor flow candidates and its flow. Given a shift Δp for all flow, the correlation calculation of propagation can be defined as:

$$Corr = F_1 \cdot \mathbf{W}(F_2, \mathbf{S}(flow, \Delta p)), \quad (3)$$

Where, $\mathbf{S}(flow, \Delta p)$ refers to shift flow according to Δp, $\mathbf{W}$ refers to warp F_2 with shifted flow. There is no doubt that the more seed points are selected, the more operations are needed. When choosing n seed points for m iterations

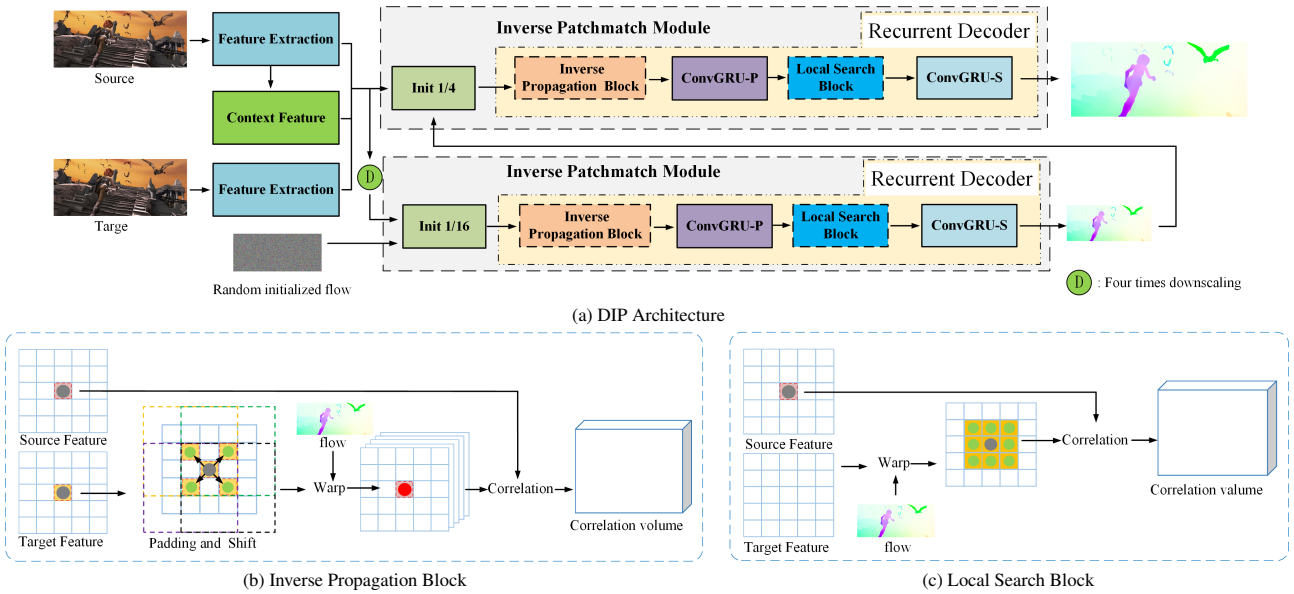


Figure 3. (a) Architecture overview. Given a pair of flow images, we first extract deep 1/4 and 1/16 scale features and context information. The extracted features and context information are then used to the initialization of 1/16 inverse Patchmatch, which is fed into the inverse propagation Block, local search Block and GRU modules for iterative optimization of flow. Then we use the optical flow predicted on 1/16 for the initialization of 1/4 inverse Patchmatch, and repeat the operation of Inverse Patchmatch Network. Please note that the parameters used by 1/4 and 1/16 Inverse Patchmatch Network are exactly the same. (b) Inverse Propagation Block propagates neighbor information. (c) Local Search Block is used to refine the flow.

of propagation, propagation needs to shift the optical flow $n \times m$ times and warp the source feature $n \times m$ times. This increases memory operations and interpolation calculations, especially when predicting high-resolution optical flow. In order to reduce the number of options, for the first time we replace propagation with inverse propagation. In the search stage, we change the random search to a local search method which is more suitable for end-to-end network and achieves higher accuracy. More details in patchmatch method can be seen in the supplementary.

3.3. Deep Inverse Patchmatch

Inverse Propagation In propagation, the optical flow shift and feature warping are serial and coupled, since the warping process depends on the shifted flow. Moreover, multiple flow shifts are necessary in each iteration, so the computations increase. In theory, the spatial relative position of shifting the flow to the down-right is the same as shifting the target to the top-left. And the correlation maps of the two methods have one pixel offset in the absolute space coordinates. We name the way of shifting targets as inverse propagation, and the inverse propagation can be formulated as follows:

$$Corr = F_1 \cdot \mathbf{S}(F_2^{'}, -\Delta p), \tag{4}$$

and

$$F_2^{'} = \mathbf{W}(\mathbf{S}(F_2, \Delta p), flow) \tag{5}$$

In theory, combining Eq. (5) and Eq. (4) is completely equivalent to Eq. (3). Since Δp is very small, we ignore the process of back propagation in our implementation. Then Eq. (4) can be replaced with:

$$Corr = F_1 \cdot F_2^{'} \tag{6}$$

In inverse propagation, a target feature point is scattered to its seed points and warped by the optical flow of the seed points. Thus, we can shift and stack the target features in advance, then perform warping only once to obtain the warped target features in each iteration. The details of inverse propagation can be described in Fig. 3b.

In this work, the seed points is static and do not change with the increase of iterations. Hence target features only need to be shifted to seed points once and shifted target features can be reused in every iteration. In this way, if there are n seed points for m iterations of propagation, we only need to shift target features n times and warp the shifted target features m times. Fig. 2b shows the inverse propagation stage and whole the stage can be divided into two sub-stages:

- **Initialization Stage:** Input source feature, target feature. Shift the target feature according to the seed points, and then stack these shifted target features as shared target features along the depth dimension.
- **Running Stage:** Input a flow, warp shared target features according to the flow, and compute correlation between source feature and warped target features.

Local Search It is difficult to obtain very accurate optical flow by patch propagation alone, since the range of randomly initialized flow values is very sparse. Therefore, a local neighborhood search is performed after each patch propagation in this work. Unlike [5], which performs a random search after each propagation and reduces the search radius with increasing iteration. We only perform a fixed small radius search after each propagation and call it local search. The entire local search block is shown in Fig. 3c. Given an optical flow increment Δf, the local search can be formulated as:

$$Corr = F_1 \cdot \mathbf{S}(\mathbf{W}(F_2, flow), \Delta f) \tag{7}$$

In this work, we set the final search radius to 2 according to the experimental results. Details are described in Section 4.2.

To this end, the Inverse Patchmatch module, as shown in Fig. 3a, consists mainly of the Inverse Propagation Block and the Local Search Block. In each iteration, an inverse propagation is followed by a local search. It is worth noting that both blocks use GRU [36] for cost aggregation.

3.4. Network Architecture

In order to obtain high-precision optical flow on high-resolution images, we designed a new optical flow prediction framework named DIP. The overview of DIP can be found in Fig. 3. It can be described as two main stages: (1) feature extraction; (2)multi-scale iterative update.

Feature Extraction At first, a feature encoder network is applied to the input images to extract the feature maps at 1/4 resolution. Unlike previous works [19,20,36,44] which use a context network branch to specifically extract the context. DIP directly activates the source feature map as a context map. Then we use the Average Pooling module to reduce the feature maps to 1/16 resolution. And we use the same backbone and parameters for both 1/4 resolution and 1/16 resolution. Therefore, DIP can be trained in two stages, and we use more stages for inference when processing large images.

Multi-scale Iterative Update Our method is based on neighborhood propagation and thus must iteratively update the optical flow. Our network consists of two modules, an inverse propagation module and a local search module. In the training stage, we start the network with a random flow of size 1/16 and then iteratively optimize the optical flow at both scale 1/16 and scale 1/4 using a pyramid method. During the inference stage, we can perform the same process as in the training stage. To obtain a more accurate optical flow, we can also refine the optical flow at scale 1/8 and then optimize the result at scale 1/4. More high-resolution detailed comparisons can be found in the supplementary material.

Our network also accepts the initialized optical flow as input in the inference stage. In this case, we adapt the number of inference layers of the pyramid according to the maximum value of the initialized optical flow. For example, the forward interpolation of the optical flow of the previous image is used as input for the current image when the optical flow of the video images is processed. With the information of the previous optical flow, we can use two or more pyramids for large displacements to ensure accuracy, and use one pyramid for small displacements to reduce inference time.

4. Experiments

In this section we demonstrate the state-of-the-art performance of DIP on Sintel [7] and KITTI [28] leaderboards and show that it outperforms existing methods in the zero-shot generalization setting on Sintel and KITTI. The endpoint error (EPE) is reported in the evaluation. For KITTI, another evaluation metric, F1-all, is also reported, which indicates the percentage of outliers for all pixels. For benchmark performance evaluation, d_{0-10} and d_{10-60} on Sintel are also used to estimate the optical flow in small motion regions. Here, d_{0-10} means the endpoint error over regions closer than 10 pixels to the nearest occlusion boundary.

4.1. Training schedule

DIP is implemented in Pytorch [29] with 16 RTX 2080 Ti GPUs. Following RAFT [36], we use the AdamW [24] optimizer and the OneCycle learning rate schedule [32] in the training process.

Training Details In the generalization experiment, we train our model on the datasets FlyingChairs [10] and FlyingThings3D [27] and evaluate the generalization ability on the training set of Sintel [7] and KITTI2015 [28]. In the pre-train stage, we decide to combine FlyingChairs and FlyingThings3D in a ratio of 1:10. First, the training size is set to 512×384, and the model is trained for 100k steps with a batch size of 32. Then the model is finetuned on size of 768×384 for another 100k steps with batch size of 16. During training and inference of ablation studies, we use 6 iterations for DIP flow regression. And the number of iterations is set to 12 during benchmark performance evaluation.

We also performed fine-tuning on Sintel [7], KITTI [28] and HD1K [21] datasets. We perform fine-tuning on Sintel for 100k by combining data from Sintel and FlyingThings3D [27] and training size is 768×384. Finally, we perform fine-tuning using a combination of data from FlyingThings, Sintel, KITTI-15, and HD1K for 100k with a training size of 832×320.

Loss Our loss function is similar with RAFT [36]. DIP outputs two optical flows for each iteration. Thus, $N =$

Method	Sintel (train)		KITTI-15 (train)		Params	448×1024		1088×1920	
	Clean	Final	EPE	F1-all		Memory	Time(ms)	Memory	Time (ms)
Sparse global [20]	1.29	2.95	6.80	19.30	5.00M	3.04G	839	5.98G	3971
Dense global	1.30	2.97	4.96	**14.02**	3.40M	10.47G	234	OOM	-
only p(N=4)	1.62	3.40	7.63	19.81	2.78M	**1.48G**	112	**3.27G**	**325**
only ls(r=1)	1.48	3.02	12.38	23.76	3.40M	1.56G	**96**	3.45G	373
pm(N=4, r=1)	**1.26**	**2.93**	**4.89**	14.33	5.10M	1.56G	106	3.70G	372

Table 1. Ablation study concerning correlation volume. Models are trained on FlyingChairs [10] and FlyingThings3D [27]. Memory and inference time are measured on a RTX2080 Ti GPU. *global* means global correlation volume. *only p(N=4), ls(r=1)* means that only use propagation with seeds of 4 or local search with radius 1. *pm(N=4, r=1)* means Patchmatch that combines propagation and local search. The number of iterations is set to 6 for Patchmatch and 12 for other methods. The best results are marked with bold and the second best results are marked with underline.

pm		Sintel		KITTI-15		1088×1920
N	r	clean	final	EPE	F1-all	Time(ms)
4	1	**1.26**	2.93	4.89	14.33	**372**
4	**2**	1.27	2.83	**4.41**	**13.51**	432
4	3	1.31	2.85	4.54	13.80	523
8	2	1.28	**2.79**	4.45	13.77	503

Table 2. Ablation study of the number of seeds and the local search radius based on Patchmatch. Validated on Sintel and KITTI-15 training datasets and iteration is set to 6. The best results are marked with bold and the second best results are marked with underline.

Method	Sintel		KITTI-15		1088×1920
	clean	final	EPE	F1-all	Time(ms)
pm	**1.27**	2.83	4.41	**13.51**	432
ipm	1.30	**2.82**	**4.29**	13.73	**327**

Table 3. Ablation study of Patchmatch and inverse Patchmatch on Sintel and KITTI-15 training datasets. Where pm means Patchmatch and ipm means inverse Patchmatch. Among them, the seeds of propagation is 4. The radius of local search is 2. The best results are marked in bold.

$iters \times 2 \times 2$ predictions are output throughout the training process when N iterations are used at both 1/16 and 1/4 resolution. Since there are multiple outputs for supervise, we use the similar strategy with RAFT, to compute a weighting sequence and sum the loss of the prediction sequence with it. The total loss can be formulated as follows:

$$loss = \sum_{i=0}^{i=N} w_i \cdot M(|f_i - f_{gt}|), \tag{8}$$

where N is the length of the prediction sequence, $M(x)$ represent the mean of the matrix x, and the w_i can be computed by Eq. (9), we use $\gamma = 0.8$ in our training.

$$w_i = \gamma^{N-i-1} \tag{9}$$

4.2. Ablation study

Correlation Volume We first analyze the accuracy, memory and inference time of key components in our proposed method in Tab. 1. In this comparative experiment, SCV(*Sparse global*) [20] is selected as a benchmark because it has low correlation volume in memory and state-of-the-art performance. In addition, we construct 4D correlation volumes with a resolution of (*Dense global*) 1/16 and 1/4 resolution respectively, and each iteration performs a lookup like RAFT [36]. Using these benchmarks, we have conducted a partial experimental comparison. In the experiment, we implement a propagation experiment with a seed point of 4 and a local search experiment with a radius of 1 respectively. The results are clearly that only propagation(*only p*) or local search(*only ls*) has great advantages in terms of memory and speed at large resolutions, but the accuracy is reduced compared to the global method. The combination of propagation and local search (*pm*) uses less time and memory to achieve comparable or better results than the global method. Especially, DIP consumes 10× less inference time than SCV on the size of 1088×1920.

Hyperparameters Based on Patchmatch, we further experiments with hyperparameters and present them in Tab. 2. At first, the number of propagation seed points is set to 4, and the radius of local search is changed from 1 to 3. We can see that the accuracy is further improved when the search radius is increased from 1 to 2. When it is increased to 3, the accuracy is basically the same as radius 2, but the model inference time increases by 21% . So the radius of the local search is fixed at 2. Then we change the number of propagation seed points from 4 to 8. However, the result is not improved significantly, but the model consumption increases. So we set the number of seed points to 4 for further optimization.

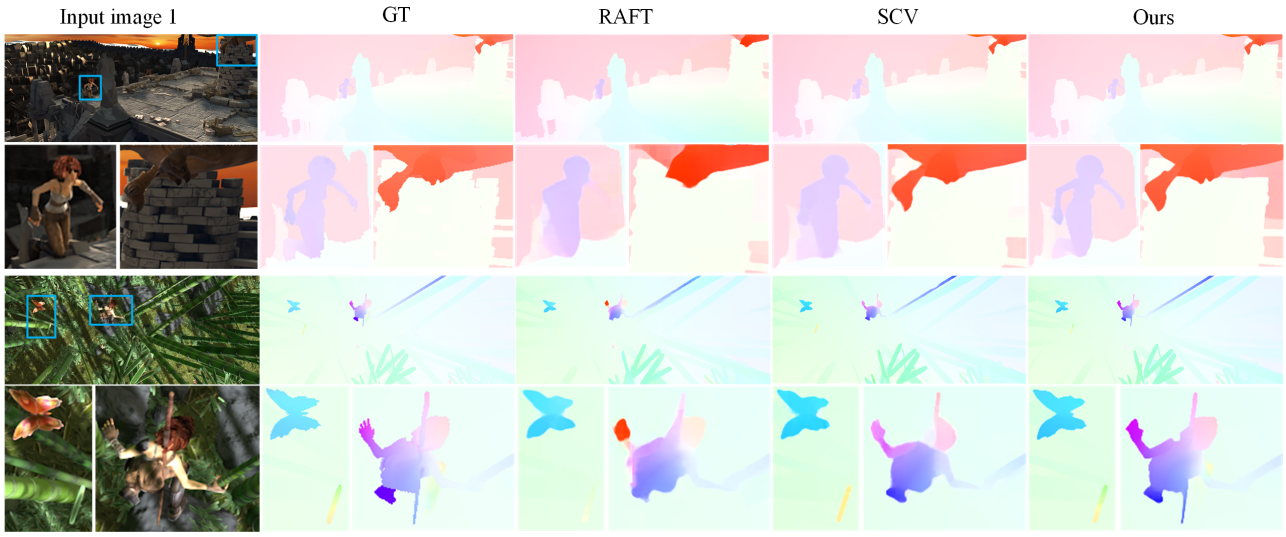


Figure 4. Visual comparison of optical flow estimates on the Sintel-Clean dataset. Compared with RAFT and SCV, our method performs particularly well, and our result is close to GT in the enlarged image frame. More results can be found in supplementary materials.

Patchmatch and Inverse Patchmatch Finally, we verified the effectiveness of the inverse Patchmatch and showed it in Tab. 3. In this experiment, we replaced the calculation method of correlation from propagation to inverse propagation, and adopted the previous training and evaluation strategy. The experiment shows that inverse propagation can achieve almost the same results as propagation. With a size of 1088×1920, the inference time of inverse Patchmatch is reduced by 24% compared to Patchmatch.

In summary, based on our Patchmatch framework, we can achieve better performance with lower memory, and use inverse Patchmatch instead of Patchmatch to achieve the same performance with faster inference speed.

4.3. Comparison with Existing Methods

To demonstrate the superiority of our method, we have made a comprehensive comparison with the existing methods, including generalization, memory and special results.

Generalization In order to verify the generalization of the model, we choose to use FlyingChairs [10] and FlyingThings3D [27] for training and Sintel [7], KITTI [28] for test. Details are described in Section 4.1 and results are show in Tab. 4. Experiments show that our method exhibits strong generalization and achieves state-of-the-art results in the KITTI-15 dataset. Among them, F1-all is 13.73%, reducing 21% from the best published result (17.4%). On the Sintel dataset, we have also achieved results comparable to the state-of-the-art methods.

Memory and High-resolution Results We measure the accuracy and memory of different correlation volume algorithms at different resolutions in Fig. 5. Since there are few real and high-resolution datasets for the flow task, in the experiment we use the up-sampled kitti dataset for memory and accuracy evaluation. It can be seen that under the limitation of 11GB memory, the maximum output image scale of RAFT [36] is only 2.25. Moreover, the accuracy of SCV [20] is rapidly decreasing as the image scale increases. This demonstrates the effectiveness of our approach in saving memory and stabling accuracy when scaling correlation volumes to higher resolutions.

Method	Sintel(Train)		KITTI-15(train)	
	Clean	Final	EPE	F1-all
HD3 [46]	3.84	8.77	13.17	24.0
LiteFlowNet [16]	2.48	4.04	10.39	28.50
PWC-Net [34]	2.55	3.93	10.35	33.7
LiteFlowNet2 [17]	2.24	3.78	8.97	25.90
VCN [45]	2.21	3.68	8.36	25.10
MaskFlowNet [47]	2.25	3.61	-	23.10
FlowNet2 [18]	2.02	3.54	10.08	30
DICL [41]	1.94	3.77	8.70	23.60
RAFT [36]	1.43	**2.71**	5.04	17.40
Flow1D [44]	1.98	3.27	6.69	22.95
SCV [20]	**1.29**	2.95	6.80	19.30
ours	1.30	2.82	**4.29**	**13.73**

Table 4. Results on Sintel and KITTI. EPE refers to the average endpoint error and F1-all refers to the percentage of optical flow outliers over all pixels. The best results are marked with bold and the second best results are marked with underline. Missing entries '-' indicates that the result is not reported in the compared paper.

Benchmark Results The performances of our DIP on the Sintel and KITTI-15 benchmarks are shown in Tab. 5. We

	Method	Sintel (test)						KITTI-15 (test)	
		Clean			Final			F1-all	
		EPE	d0-10	d10-60	EPE	d0-10	d10-60	All pixels	Non-Occ pixels
2-view	FlowNet2 [18]	4.16	3.27	1.46	5.74	4.81	2.55	11.48	6.94
	PWC-Net+ [35]	3.45	3.91	1.24	4.6	4.78	2.04	7.72	4.91
	LiteFlowNet2 [17]	3.48	3.27	1.43	4.69	4.04	1.89	7.74	4.42
	HD3 [46]	4.79	3.22	1.37	4.67	3.58	1.76	6.55	-
	VCN [45]	2.81	3.26	0.86	4.4	4.38	1.78	6.3	3.89
	MaskFlowNet [47]	2.52	2.74	0.9	4.17	3.78	1.74	6.1	3.92
	ScopeFlow [4]	3.59	3.45	1.26	4.1	4.02	1.68	6.82	4.45
	DICL [41]	2.12	2.2	0.58	3.44	3.27	1.28	6.31	-
	RAFT [36]	1.94	-	-	**3.18**	-	-	5.1	3.07
	Flow1D [44]	2.24	2.18	0.87	3.81	3.60	1.75	6.27	-
	SCV [20]	1.72	1.39	**0.45**	3.6	3.24	1.42	6.17	3.43
	Ours	**1.67**	**1.18**	**0.45**	3.22	**2.68**	**1.23**	**4.21**	**2.43**
warm-start	RAFT	1.61	1.62	0.51	2.86	3.11	1.13	-	-
	SCV	1.77	-	-	3.88	-	-	-	-
	Ours	**1.44**	**1.10**	**0.41**	**2.83**	**2.72**	**1.09**	-	-

Table 5. Benchmark performance on Sintel and KITTI Test datasets. Missing entries '-' indicates that the result is not reported in the compared paper and could not found on online benchmark. The best results are marked with bold and the second best results are marked with underline.

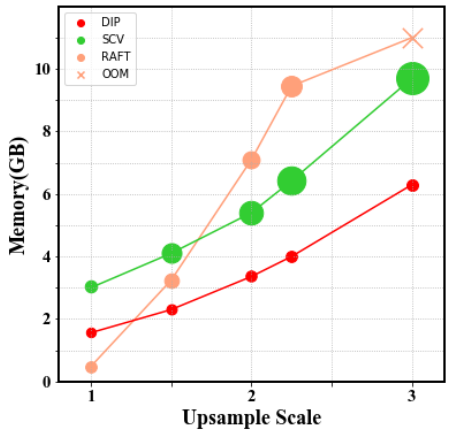


Figure 5. Upsampling to high-resolution size results. The memory limit is 11GB and the area of the bubbles is a mapping of the F1-all metric. We use upsampling of the KITTI dataset to evaluate memory and accuracy, and the resolution at the scale of 1 is 375 x 1242. 'OOM' means out of memory.

have achieved state-of-the-art results (1.72 $\rightarrow$ 1.67) on the Sintel-Clean dataset in the two-view case. Similar to RAFT, we also adopt the "warm-start" strategy which initialises current optical flow estimation with the flow estimates of the previous frame. On the Sintel-Clean benchmark our method ranks second for EPE. Compared with RAFT, we have improved the EPE from 1.61 to 1.44 (10.5% improvement). What's interesting is that our method achieves the best results on the d_{0-10} and d_{10-60}, which shows that our method has obvious advantages in estimating the optical flow in small motion areas. Fig. 4 shows qualitative results of DIP on Sintel. Compared with RAFT and SCV, our results are much closer to the ground truth in the fine structure area.

On the KITTI-15 benchmark, our method ranks first on all the metrics among the published optical flow methods. Compared with RAFT, we have improved the F1-all from 3.07% to 2.43% (20.8% improvement) on Non-occluded pixels and the F1-all from 5.10% to 4.21% (17.5% improvement) on all pixels.

5. Conclusions

We propose a deep inverse Patchmatch framework for optical flow that focuses on reducing the computational cost and memory consumption of dense correlation volume. By reducing the computational and memory overhead, our model can work at a high-resolution and preserve the details of fine-structure. We also show a good trade-off between performance and cost. At the same time, we achieve comparable results with the state-of-the-art methods on public benchmarks and good generalization on different datasets. We believe that our inverse Patchmatch scheme can be used in more tasks, such as stereo matching, multi-view stereo vision and so on. In the future, more attention will be paid on the motion blur, large occlusion and other extreme scenes.

References

[1] Mohammed Almatrafi and Keigo Hirakawa. Davis camera optical flow. *IEEE Transactions on Computational Imaging*, 6:396–407, 2019. 1

[2] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3250–3259, 2017. 2

[3] Linchao Bao, Qingxiong Yang, and Hailin Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3534–3541, 2014. 2

[4] Aviram Bar-Haim and Lior Wolf. Scopeflow: Dynamic scene scoping for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7998–8007, 2020. 8

[5] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 1, 2, 3, 5

[6] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11, 2011. 1, 2, 3

[7] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012. 1, 2, 5, 7

[8] Jason Chang and John W Fisher. Topology-constrained layered tracking with latent flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 161–168, 2013. 2

[9] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *Proceedings of the IEEE international conference on computer vision*, pages 686–695, 2017. 2

[10] 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 *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 1, 2, 3, 5, 6, 7

[11] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4384–4393, 2019. 2

[12] HW Ho, Christophe De Wagter, BDW Remes, and Guido CHE de Croon. Optical flow for self-supervised learning of obstacle appearance. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3098–3104. IEEE, 2015. 1

[13] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 1

[14] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5712, 2016. 1, 2, 3

[15] Tak-Wai Hui and Chen Change Loy. Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation. In *European Conference on Computer Vision*, pages 169–184. Springer, 2020. 2

[16] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8981–8989, 2018. 1, 7

[17] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow cnn—revisiting data fidelity and regularization. *IEEE transactions on pattern analysis and machine intelligence*, 43(8):2555–2569, 2020. 1, 7, 8

[18] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 1, 2, 7, 8

[19] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. *arXiv preprint arXiv:2104.02409*, 2021. 1, 5

[20] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning optical flow from a few matches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16592–16600, 2021. 2, 5, 6, 7, 8

[21] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 19–28, 2016. 5

[22] Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision*, pages 471–488. Springer, 2016. 1

[23] Fangjun Kuang. Patchmatch algorithms for motion estimation and stereo reconstruction. Master's thesis, 2017. 3

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[25] Bruce D Lucas, Takeo Kanade, and Others. An iterative image registration technique with an application to stereo vision. In *Proc. of the Intl. Joint Conference on Artificial Intelligence*, volume 81, pages 674–679, 1981. 1

[26] Osama Makansi, Eddy Ilg, and Thomas Brox. End-to-end learning of video super-resolution with motion compensation. In *German conference on pattern recognition*, pages 203–214. Springer, 2017. 1

[27] Nikolaus Mayer, Eddy Ilg, Philip Hausser, 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 5, 6, 7

[28] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2:427, 2015. 1, 2, 5, 7

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 5

[30] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016. 1

[31] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014. 1

[32] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019. 5

[33] Deqing Sun, Jonas Wulff, Erik B Sudderth, Hanspeter Pfister, and Michael J Black. A fully-connected layered model of foreground and background flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2451–2458, 2013. 2

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

[35] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1408–1423, 2019. 8

[36] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 1, 2, 3, 5, 6, 7, 8

[37] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J Black. Video segmentation via object flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3899–3908, 2016. 2

[38] Zhigang Tu, Zuwei Guo, Wei Xie, Mengjia Yan, Remco C Veltkamp, Baoxin Li, and Junsong Yuan. Fusing disparate object signatures for salient object detection in video. *Pattern Recognition*, 72:285–299, 2017. 1

[39] Zhigang Tu, Wei Xie, Dejun Zhang, Ronald Poppe, Remco C Veltkamp, Baoxin Li, and Junsong Yuan. A survey of variational and cnn-based optical flow techniques. *Signal Processing: Image Communication*, 72:9–24, 2019. 1

[40] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14194–14203, 2021. 2

[41] Jianyuan Wang, Yiran Zhong, Yuchao Dai, Kaihao Zhang, Pan Ji, and Hongdong Li. Displacement-invariant matching cost learning for accurate optical flow estimation. *arXiv preprint arXiv:2010.14851*, 2020. 7, 8

[42] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE international conference on computer vision*, pages 1385–1392, 2013. 1

[43] Fanyi Xiao and Yong Jae Lee. Track and segment: An iterative unsupervised approach for video object proposals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 933–942, 2016. 1

[44] Haofei Xu, Jiaolong Yang, Jianfei Cai, Juyong Zhang, and Xin Tong. High-resolution optical flow from 1d attention and correlation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10498–10507, 2021. 1, 2, 5, 7, 8

[45] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. *Advances in neural information processing systems*, 32:794–805, 2019. 7, 8

[46] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6044–6053, 2019. 1, 7, 8

[47] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020. 7, 8