PatchFlow: A Two-Stage Patch-Based Approach for Lightweight Optical Flow Estimation

Ahmed Alhawwary, Janne Mustaniemi, and Janne Heikkilä

Center for Machine Vision and Signal Analysis, University of Oulu, Finland
{ahmed.alhawwary,janne.mustaniemi,janne.heikkila}@oulu.fi

Abstract. The deep learning-based optical flow methods have shown noticeable advancements in flow estimation. The dense optical flow map offers high flexibility and quality for aligning neighbouring video frames. However, they are computationally expensive, and the memory requirements for processing high-resolution images such as 2K, 4K and 8K on resources-limited devices such as mobile phones can be prohibitive.
We propose a patch-based approach for optical flow estimation. We redistribute the regular CNN-based optical flow regression into a two-stage pipeline, where the first stage estimates an optical flow for a low-resolution image version. The pre-flow is input to the second stage, where the high-resolution image is partitioned into small patches for optical flow refinement. With such a strategy, it becomes possible to process high-resolution images when the memory requirements are not sufficient. On the other hand, this solution also offers the ability to parallelize the optical flow estimation when possible. Furthermore, we show that such a pipeline can additionally allow for utilizing a lighter and shallower model in the two stages. It can perform on par with FastFlowNet (FFN) while being 1.7x faster computationally and with almost a half of the parameters. Against the state-of-the-art optical flow methods, the proposed solution can show a reasonable accuracy trade-off for running time and memory requirements. Code is available at: https://github.com/ahmad-hammad/PatchFlow

Keywords: Optic flow · Lightweight CNN · High-resolution video.

1 Introduction

Optical flow is a long-standing problem in the computer vision field with ongoing research to date. Given two consecutive frames I1 and I2, the optical flow is per-pixel 2D motion map estimation from the first frame I1 to the next frame I2. It has a vital role in several applications such as video stabilization [36], image stitching [23, 24] , crowd-counting [10], super resolution [4, 31], video interpolation [25], depth estimation [22], SLAM [40] and many robotics applications [40, 19].

Conventional optical flow methods [9, 37, 5] are often formulated as a hand-crafted optimization problem. With the advances in hardware and deep learning research, the deep learning-based optical flow methods have shown noticeable

improvements to the previous conventional methods. In addition, deep neural networks harness the parallel processing of modern GPUs. Thus, they surpass the traditional methods in terms of speed and accuracy. Since the first convolutional neural network-based (CNN) method FlowNet [16], many methods have improved the architecture and the training protocols. RAFT [30] is a milestone in optical flow estimation achieving state-of-the-art results. They compute a huge 4D cost volume and update a single high-resolution (1/8) flow iteratively without warping rather than the coarse-to-fine scheme followed by previous works. On the other hand, pyramidal coarse-to-fine models, such as PWCNet [29], have lower computational and memory requirements. However, they perform worse due to the warping process included in each pyramid level. Some subsequent papers [33, 18] proposed solution that builds on RAFT's achievement to lower the memory requirements but they are still relatively more complex than pyramidal models. Despite the improvements introduced by the CNN-based solutions for the optical flow task, it remains computationally expensive and memory-hungry.

The prevalence of smartphones equipped with high-end cameras capable of recording high-resolution videos makes it tempting to use them for computational photography applications that fuse information over multiple frames. Because of sudden camera movements those applications including, for example, deblurring, denoising, super-resolution, video stabilization, and 3D reconstruction, often require estimation of dense optical flow. However, regardless of mobile GPUs, smartphones are resource-limited devices, both in computational power and memory resources. Thus, the optical flow computational and memory burden make it less attractive to import such applications on those devices. Moreover, for high-resolution images such as 2K, 4K and 8K images, the computation and memory requirements can be prohibitive.

To this end, we propose a patch-based framework for optical flow estimation. We redistribute flow estimation into two stages, where the first stage estimates a flow for a low-resolution image which serves as an initial flow for the next stage. The second stage estimates flow on small equally-sized patches of the images with the guidance of the low-resolution flow. Such a strategy brings double-edged benefits. On one hand, it removes the high memory requirements for high-resolution images. On the other hand, the patch-based strategy can be leverged to speed up the optical flow estimation by parallel processing for applications that have running time priority. Our contributions can be summarized as follows:

- We propose a two-stage pipeline for patch-based optical flow regression eliminating high memory requirements for high-resolution images, especially when it comes to resource-limited devices such as mobile phones. From another aspect, it enables parallel processing when the memory permits.
- We show reasonable speed-accuracy trade-offs against variety of heavier *full-resolution* optical flow methods.
- We show that the two-stage approach enables us to incorporate a lighter and shallower network in each stage without hurting the accuracy.

2 Related Work

With advances in deep learning research, CNN-based optical flow methods have become superior to the conventional methods in terms of accuracy and inference speed. However, they draw inspiration from the steps followed in the classical pipeline. Here we review the deep learning-based optical flow estimation. FlowNet [7] was the start of the end-to-end, CNN-based optical flow methods that paved the way to improve the optical flow estimation. FlowNet proposed two Unet-shaped architectures: FlownetSimple and FlownetCorr. The main difference is that the first one does not include cost (correlation) volume computation. FlowNet2 [16] builds a heavier and more complex solution by cascading blocks of FlownetCorr and FlownetSimple for optical flow refinement. They also showed that the accuracy results improve by carefully scheduling the dataset training. Most of the CNN-based methods contain three main ingredients: feature extraction, cost volume computation, and feature decoder for optical flow prediction. The supervised CNN-based solutions can be roughly divided into two classes: the warping-based coarse-to-fine (pyramidal refinement) optical flow and the recurrent, non-warping solutions. First, we present the warping-based method.

PWCNet [29] first extracts pyramidal features, then in each pyramid level, it computes a cost volume and predicts a flow which is then refined in the finer level. In SPynet [28], they use image pyramids instead of features. In IRR [15], they turn PWCNet [29] and FlownetSimple [7] to iterative, instead of stacking blocks of large networks similar to FlowNet2 [16]. For IRR-PWCNet, the pyramidal coarse-to-fine architecture is preserved, but it is iterative in the sense that they, unlike PWCNet [29], share the weights of the decoder in each pyramid level. Thus, the number of iterations is limited by the number of levels. IRR improves the accuracy by bidirectional estimation of the flow, jointly estimating the occlusions and the bilateral filtering for more refinement for the optical field [16]. Instead of reshaping cost volume to multi-channel 2D arrays as in PWCnet [29] for instance, VCN [35] processes 4D cost volume with separable 2D kernels for efficient computation and memory consumption showing a significant improvement over previous methods. LiteFlowNet [13] was a concurrent work to PWCNet [29] and shared similar techniques with theirs. It has a smaller number of parameters than PWCNet [29] but requires a higher number of FLOPs (Floating Point Operations) while maintaining comparable accuracy. LiteFlowNet2 [14] further improves the accuracy and running time by optimizing the architecture and training protocols of LiteFlowNet [13]. LiteFlowNet3 [12] improves the accuracy, but incurs more computational complexity, by introducing cost volume modulation and flow field deformation (correction) to better handle the occlusion and homogeneous regions. For cost modulation, they estimate affine parameters that are used to transform each pixel's cost. To refine the flow, they replace the flow vector with a more accurate flow vector from the neighbouring pixels based on a confidence map and self-correlation cost volume (correlating the first image feature map with itself). FastFlownet [19] is based on PWCNet [29] architecture and modifies the three main ingredients to produce

a lightweight flow network. They build a lighter feature extractor by replacing the convolution in low-scale levels by pooling layers. For the decoder, they use shuffling layers inspired by Shufflenet [41]. For the cost volume, they compute the correlation in a limited search radius range (local search grid around each pixel) and then resample the correlation grids non-uniformly such that it is dense around the centre and dilated otherwise.

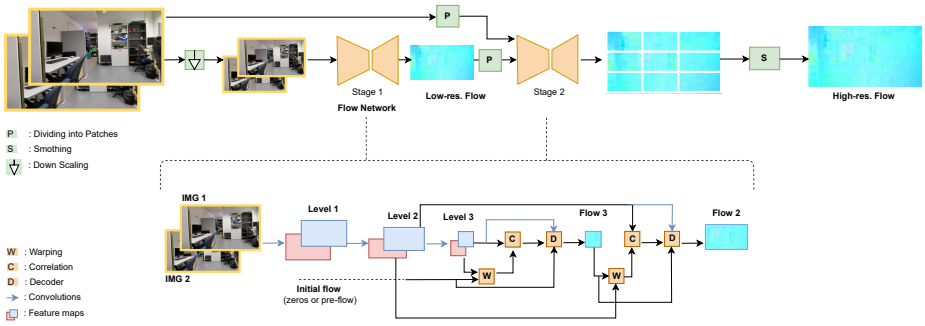

Fig. 1. Architecture of the two-stage optical flow pipeline. In the top of the image, a high-level overview of the full pipeline. The bottom image illustrates the details of the Flow Network incorporated in each stage.

In IOFPL [8], they present ideas to improve the flow estimation in the pyramidal architectures. Among those ideas, they shift from warping to sampling-based cost volume computation in pyramid levels, ameliorating the decline of the performance resulting from warping artefacts. They also propose blocking some gradient components during back-propagation for better convergence and inference performance. Before [8], Devon [26] also proposed sampling-based cost volume. Unlike the coarse-to-fine strategy in [8], they iteratively refine a fixed quarter resolution flow with a shared decoder. To handle the large motion in the fixed resolution, they used dilated search grids for the cost volume computation. Although they perform well on the Sintel [3] clean pass, they perform worse than [8] on the Sintel final pass and KITTI 2015 dataset [27]. RAFT [30] is a remarkable milestone which proposed sampling-based (no-warping), recurrent solution achieving the state-of-the-art (SOTA) results on the optical flow benchmarks when first appeared. They build 4D cost volume by computing the correlation between all pixel pairs on 1/8 resolution feature maps and then pool it recursively to produce multiscale 4d cost volume. The optical flow is iteratively refined at a fixed 1/8 resolution using a GRU-based recurrent module [6]. In each iteration, they look up (sample) the multiscale cost volume with a limited search radius based on the current estimated flow. The RAFT model has high computational and memory requirements. This is mainly due to the huge 4D multiscale volume and many refinement iterations performed on the fixed resolution flow. Assuming an image with equal width and height N, the cost

volume has a computation and memory complexity of $O(N^4)$. This limits its ability to scale up to higher resolutions. For instance, a 4K image can cause an out-of-memory error on a GPU with a memory of 32GB [33]. Several subsequent methods [33, 18, 39, 17, 42] either optimized the RAFT solution while keeping a similar performance [33, 18] or surpassed the performance of RAFT [39, 17]. However, those methods still have high computation and memory complexity. In [33], they decrease the memory requirement by separating the 4d cost volume into 2 3D cost volumes, leading to complexity $O(N^3)$ while achieving comparable accuracy. DIP [42] is inspired by the conventional patchmatch approaches [1, 2] to deal with the high memory consumption of [30]. There is also a direction that adopts the transformer models for optical flow estimation [34, 11].

In the pyramidal approaches, the warping causes occlusion artefacts (copies of the occluding pixels), which may hinder the subsequent levels from picking the correct correspondences, leading to ghosting artefacts[18]. However, the coarse-to-fine often enjoys lower computational and memory requirements than RAFT and its successors. The cost volume in such methods requires memory of complexity $O(N^2R^2)$ where R is the search radius and is much smaller than N. Our work follows the pyramidal coarse-to-fine architecture. We redistribute the optical flow regression into two stages where the first stage estimate low-resolution optical flow and the second stage apply patch-based refinement. The output flow is then smoothed to produce the final result. In each stage, we incorporate a pyramidal network that estimates flow on two pyramid levels only. The resulting pipeline has lower computational complexity and memory consumption while achieving a comparable result with FastFlowNet [19] on a modified FlyingChairs dataset. Our model shows reasonable complexity-accuracy trade-offs against the SOTA methods.

3 Proposed Method

3.1 Overview

We first estimate the optical flow on a low-resolution version which can be computationally cheap compared to the full resolution. For instance, if we have 4K resolution images, we can downsample them to 1/8 and estimate the optical flow. The image pair and the pre-flow are divided into equally-sized patches where each patch is going to be refined independently. The pre-flow serves as an initial flow for the second stage. This pre-flow is utilized to warp the second image features of the coarsest level in the second stage as detailed in Section 3.2. After estimating the optical flow for each patch in the second stage, the patches of optical flow are concatenated and then smoothed to produce the final optical flow field. A high-level illustration of the whole pipeline is shown in the upper part of Figure 1.

3.2 Network architecture

The network consists of two subnetworks or stages as shown in Figure 1. The two stages have the same architecture. We present the details of the subnetwork

in this section. A subnetwork has three main components: the pyramidal feature extraction encoder, correlation (or cost) volume and optical flow decoder. The network follows the warping-based pyramidal optical flow architectures [29, 13].

Pyramidal Feature Extraction. We extract features from the two images separately using a lightweight 3-level pyramidal feature encoder with tied weights as depicted in Figure 1 in the bottom part. The feature maps in each level are 1/2 resolution of the previous level. This results in feature maps of 1/2, 1/4 and 1/8 resolutions for the first, second and third levels, respectively. Each level, except the first level, has three convolution blocks and each of them consists of a 2D convolution layer plus an activation layer. The first level has 2 convolution blocks only. The dimension of a pixel's feature vector (*i.e.* , the number of channels in the output feature maps) starting from the first level are 8, 16, and 32 channels in the last level. All the convolution layers have a kernel of size 3×3.

Computing Correlation Volume. Given the feature maps $F1$ and $F2$ of the first image and the second image, respectively, the visual similarity between two feature vectors $F1(x)$ and $F2(x)$ for a pixel located in a 2D spatial position x in the feature maps is computed by the dot product of the two vectors as the following:

$$C(x, r) = F1(x) \cdot F2(x + r)/M \tag{1}$$

where $r \in [R, -R]^2$ is a 2D offset, and R is a positive integer represents the search radius and M is the length of the feature vector. In other words, each pixel (feature vector) from the first image is correlated with a local square centered at position x in the second image and has an area of $D = (2R + 1)^2$. A cost (correlation) volume is built by computing (1) for all pixels. This is then arranged in a volume of size $H_l \times W_l \times D$ [16], where H_l and W_l are the height and width of the input feature maps in level l [16].

Feature Decoder. In level l, the decoder takes as input the concatenation of the cost volume, feature map of the first image and the upsampled flow $f_{l-1}^{\uparrow}$ from the coarser level $l - 1$. It predicts a residual flow Δf_l which is then added to the flow estimate from the coarser level $f_{l-1}^{\uparrow}$ to get the refined estimate as follows:

$$f_l = f_{l-1}^{\uparrow} + \Delta f_l \tag{2}$$

The same decoder architecture is shared across all levels, but each decoder has its own set of learning parameters (*i.e.* no weight sharing) [29]. The decoder consists of seven consecutive convolution layers with a kernel of size 3×3. All convolution layers are followed by an activation layer except the final one which outputs a 2-channel flow field. The first 4 layers output 64-channel feature maps while the subsequent two layers output 32 channels.

Flow Regression. The flow estimation from the subnetworks in our pipeline follows warping-based coarse-to-fine architectures [29, 13, 19]. We describe the process of predicting flow in one scale as the same process applies to all scales. First, we warp the feature map of the second image using the upsampled flow $f_{l-1}^{\uparrow}$ estimated from the coarser scale $l - 1$. The flow from the coarser level

is upsampled using a deconvolution layer (transposed convolution) [38]. The warping module is based on a differentiable bilinear interpolation [16, 28]. Then, the correlation volume is built by computing the visual similarity between the first image feature map $F_{1,l}$ at level l and the warped second image feature map $F_{2\to1,l}$. The features $F_{1,l}$ is convolved one more time and then concatenated with the correlation volume and the upsampled flow $f^{\uparrow}_{l-1}$ and provided to the decoder. The decoder outputs the current estimated flow f_l at level l according to (2) which is then provided to the next finer level $l+1$, and the process repeats again.

In our network, we predict the optical flow for two levels. This means that the final resulted flow is 1/4 of the original resolution. We upsample the flow to the original resolution by bilinear interpolation.

In the first subnetwork (stage), the initial flow is zeros while in the second subnetwork the initial flow is the pre-flow from the first stage as depicted in Figure 1.

3.3 Smoothing Filter

The concatenated optical flow may contain discontinuities near the borders of the small patches due to the independent regression of flow for each patch. To deal with these discontinuities, we perform simple smoothing operation by minimizing an energy function that has two terms:

$$E = \sum_i c(i)\|u(i) - f(i)\|_2^2 + \lambda \sum_i |\nabla^2 f(i)|^2 \tag{3}$$

where f and u are the target and the smoothed flow to be estimated, respectively, i is the pixel index, c is the confidence map and λ is the smoothing strength. The second term represents the Laplacian of the flow field. The Laplacian term encourages the flow field to be smooth while the data term encourages the flow to keep its values unchanged based on their confidence. Since all terms are quadratic, this minimization problem can be solved using any linear solver. However, it is a time-consuming process due to the high number of unknowns. Instead, we approximate the task with iterative filtering. First, we determine a Laplacian smoothing kernel θ based on λ. The kernel size gets smaller as λ decreases, meaning a weaker smoothing effect and vice versa. After that, we initially set u to be equal to the target flow field f. Then, in each iteration k, we set $u_{k+1} = g_\theta(cf - (c-o)u_k)$ where g_θ is 2D Convolution operator with kernel θ, and o is the average between the max and min values of c. This process is guaranteed to converge to the minimum of (3) as presented in [32].

3.4 Loss Function

Following [16, 29], the optical flow training is supervised using the multiscale $L2$ norm between the ground truth f_j^{GT} and predicted optical flow at the jth scale for all optical flow estimation scales (levels) as follows:

$$\mathcal{L} = \sum_{j=1}^{L} \alpha_j \| f_j^{GT} - f_j \|_2 \tag{4}$$

where L is the number of optical flow prediction scales, and α_j is the loss weight for the jth scale.

4 Experiments

4.1 Training details

The first stage in our pipeline estimates an optical flow on a downscaled version of the high-resolution image pair. However, the resolution of FlyingChairs (FC) is small (384×512) to be downscaled further. Thus, to train our pipeline, we concatenate (vertically and horizontally) multiple small FlyingChairs images to create a higher resolution image. For instance, if we want to synthesize an image of resolution 2304×2048, we concatenate 24 small FC images. We further apply several augmentation techniques similar to previous works [29, 30]. Particularly, we apply geometric transformations such as random rotation, translation, scaling, shear, flipping and cropping. Such transformations effectively diversify the dataset and make the edges or the sharp discontinuities between the concatenated images appear spatially random in the high resolution image. This enables the network to be less influenced by the regular discontinuities during learning. This new modified FlyinChairs dataset (ModFC) is used to train the proposed pipeline. The size of the high-resolution images is 2304×2048 and it is randomly cropped to resolution 1024×1024. The first stage works on $1/8$ resolution (*i.e.* 128×128) while the refinement stage works on patches of size 128×128. We first train the low-resolution stage alone. Then, the second stage is initialized with the learned weights from the first stage, and the two stages are then trained in an end-to-end manner. The initial learning rate is $1e-4$ and decreased by 0.5 at 108, 144 and 180 epochs. We train for 216 epochs and use batch size 1. In the end-to-end training, the optical flow is predicted over two scales in each stage. Therefore, the training loss weights in (4) are set to be $\alpha_1 = 0.2$, $\alpha_2 = 0.8$, $\alpha_3 = 0.008$, and $\alpha_4 = 0.03125$ where α_2 and α_4 correspond to the coarsest level in the first and the second stage, respectively.

As described in Section 3.1, our PatchFlow (PF) consists of two subnetworks with the same architecture, where each subnetwork has its own set of learning parameters, and the pre-flow from the first network is used to warp the second image's feature map in the coarsest level as shown at the bottom of Figure 1. In the experiments, we compare our proposed pipeline with other options based on two design aspects: weight sharing and warping level. To make it easier for the reader to quickly grasp the training setting type, the 'S', 'H', 'T' and 'I' letters refer to weight sharing, non-shared weights, feature-level warping and image-level warping, respectively. In image-level warping, the pre-flow is used to warp the second image directly rather than its features. To this end, besides our

Table 1. Ablation experiments. All models trained with ModFC. The patch size used in the experiments is 256×256, while the values in the brackets are based on 128×128 patches.

Variation	Sintel - train	
	clean	final
Ours- Full	**3.74 (3.93)**	**4.94**
Ours- w/o 1st stage	10.22	10.60
Ours- w/o smoothing	3.74 (3.94)	4.94

Fig. 2. Visualizing the effect of smoothing the optical flow field. The second frame is warped using both smoothed and non-smoothed optical flow fields and overlayed on the first frame by replacing the red channel of the second frame with its counterpart in the first frame. The small coloured crops zoom in the marked regions in the images.

main model, which is referred to by PF-HT, we train two other models: PF-SI and PF-ST.

4.2 Ablation Studies

We perform ablation studies on our architecture to validate the importance of some components in our proposed pipeline. We remove the first stage to check the performance when we apply the patching directly without the pre-flow estimation. In Table 1, it is shown that the performance drops significantly without the pre-flow. Also, we show the results when disabling the smoothing component. While the smoothing seems to have no significant improvements quantitively in terms of the end-point error (EPE) [7], it has a visual effect as shown in Figure 2. It alleviates the artefacts resulting from the discontinuities between the patches of a non-smoothed flow field used for warping the second image.

4.3 Results on ModFC

As shown in Table 2, the performance of the lighter model LF is comparable with the FFN model when trained on downscaled ModFC images (small resolution). This effectively shows that the model reduction is possible while keeping a similar performance. We noticed that just inferring the low-resolution optical flow using the original FFN model pretrained with FC which is provided by [19], gives a significantly higher EPE error of 9.44. This indicates that the network is influenced by the scale of the images used for training.

Considering the 2-stage pipeline performance, it is shown that the performance of the PF-ST is a bit better than PF-SI. This indicates that the feature-level warping is better than the image-level warping. The warping of the second image to pre-align the image pair before dividing them into patching can result in occlusion artefacts in the warped image. Consequently, this can limit the ability of the refinement network to pick the correct correspondence and create ghosting artefacts. Furthermore, the performance becomes even better when each subnetwork has its own learning parameters as shown with our model PF-HT. This helps each network to focus on learning the scale-related features.

To show that the strategy of the two-stage pipeline offers the ability to utilize a lighter architecture in the refinement stage of image patches, we train the original FFN model with a similar approach to the PF-HT model. Since the first stage estimates 1/32 resolution optical flow ($1/8 \times 1/4$), it makes sense to prune the 1/64 and 1/32 flow estimation scales from the second stage. Moreover, using the first stage flow to warp such low-resolution features (1/64) wastes the information in the pre-flow and makes it almost useless for the refinement stage. In our experiments, if we keep those scales, it produces an EPE of 5.50. Thus, we remove those scales from the second stage. We refer to this as the FFN2 model. Additionally, we report the inference performance of the original FFN model on the full resolution of ModFC (no patching and one stage). From Table 2, our PF-HT has comparable performance with the FFN2 and FFN which demonstrates the possibility of decreasing the computational complexity when adopting the two-stage strategy.

4.4 Sintel and KITTI datasets

Next, we compare the performance of our two-stage model trained on ModFC with FFN2, FFN and RAFT using the Sintel dataset [3]. The EPE values are presented in Table 2 together with the runtime and the number of FLOPs.

The PF-HT has the best performance among the other versions (PF-ST and PF-SI) similar to the results on the ModFC dataset. On the other hand, the FFN2 model becomes worse than PF-HT. This shows that our model generalizes better. The lower performance might be an indication of overfitting because of the increased model size. Further, we test the original FFN model on the Sintel dataset by stacking two instances of the model. The first instance estimates the optical flow of 1/4 resolution, and the second estimates the residual motion on patches of size 256×256. It is slightly inferior to PF-ST which estimates 1/8 resolution for the pre-flow while being faster and smaller. This shows that reducing the pre-flow and refinement models does not hurt the accuracy.

Additionally, we report the accuracy on the full-resolution of Sintel (without patching) using the FFN and RAFT methods. Our model PF-HT has about 1.5 pixles higher EPE error than the RAFT model (with 12 iterations) while being 49x computationally less expensive. In comparison to FFN, it is about 0.5 pixels inferior in terms of accuracy, but on the other hand, 1.7 times faster. Also, note that both FFN and RAFT operate on the full resolution while our model works

Table 2. Performance comparison on the ModFC test set and Sintel-clean train dataset. LF refers to the model trained with the low-resolution images of the ModFC dataset. FFN2 is the stacking of two original FFN networks similar to the PF-HT model. For more details about FFN2 training, please refer to the text. FFN2* is the stacking of 2 blocks of the original FFN [19] model without training the 2 stages in an end-to-end fashion. The RAFT computation is done with 12 iterations. In those experiments, the Sintel dataset is cropped to a size of 384×1024. For the patch-based variants, we use patches of size 256×256 and 128×128 for Sintel and ModFC, respectively. The running time is measured on Nvidia RTX2070 GPU. The FLOPs and timing are based on resolution 512×1024.

Method	Training data	ModFC test	Sintel clean-train	Time (ms)	FLOPs (G)	Params (M)
Low Resolution						
LF	ModFC	5.86	-	-	-	0.39
FFN	ModFC	5.66	-	-	-	1.37
Patches Only						
FFN	FC	-	3.98	17	29.1	1.37
Two-stage: Low Resolution + Patches						
PF-ST	ModFC	3.49	3.77	10	17.0	0.39
PF-SI	ModFC	3.77	3.78	10	17.0	0.39
PF-HT	ModFC	3.13	3.62	10	17.0	0.78
FFN2	ModFC	3.08	3.87	19	29.2	2.23
FFN2*	FC	-	3.67	22	30.8	1.37
Full Resolution						
FFN	FC	3.09	3.11	17	29.1	1.37
RAFT	FC	-	2.14	227	827.2	5.30

on patches of size 256×256 which causes some loss of contextual information, and makes the flow estimation problem more difficult.

We further finetune our Network (PF-HT) with the FlyingThing3D (FT) dataset and evaluate on the Sintel [3] and KITTI [27] datasets to compare with the SOTA optical flow methods that operate on the full resolution images. We noticed that the FlyingThings3D is more challenging than FlyingChairs and has more motion with many objects occluding each other. We do not get a performance improvement when training the network with a similar strategy as with ModFC. Therefore, we train the network with the original dataset without concatenation and with a cropping size of 512×768. The first stage estimates optical flow on 1/8 resolution, and the second stage operates on patches of size 256×256. The evaluation on the Sintel and KITTI datasets is shown in Table 3. We also show the parameter count, timing and FLOPs for the different optical flow methods.

As shown in Table 3, finetuning with FT leads to a slight improvement to our model on Sintel (compared to Table 1). However, it keeps similar performance

Table 3. Performance comparison on Sintel and KITTI-2015 datasets after finetuning with FlyingThings3D (FT) dataset. MC refers to ModFC. The RAFT computation is done with 12 iterations. The FLOPs is based on a resolution of 512×1024. The timing is done using Nvidia RTX2070, while the timing with an asterisk (after the slash) is estimated on Nvidia 1080Ti from [19]. There are methods that have timing presented on both GPUs, such that the reader can better grasp the idea of the relative performance.

Training data	Method	Sintel (train)		KITTI (train)		Params	Time	FLOPs
		clean	final	F1-epe	F1-all	(M)	(ms)	(G)
Patches								
MC+FT	Ours	3.64	4.88	15.60	34.57	0.78	10	17.0
Full								
FC+FT	FFN[19]	2.89	4.14	12.24	33.10	1.37	17 /11*	29.1
	LFlownetX[13]	3.58	4.79	15.81	34.90	0.90	-/35*	-
	LFlownet [13]	2.48	4.04	10.39	28.50	5.37	-/55*	327.0
	VCN-small[35]	2.45	3.63	9.43	33.40	5.20	71	73.8
	VCN[35]	2.21	3.62	8.36	25.10	6.20	206	193.0
	Flow1D[33]	1.98	3.27	6.69	22.95	5.73	181	746,2
	SPyNet[28]	4.12	5.57	-	-	1.20	-/50*	299,6
	PWCNet[29]	2.55	3.93	10.35	33.67	8.75	51/34*	187.1
	RAFT-small[30]	2.21	3.35	7.51	26.9	1.0	71	182.2
	RAFT[30]	1.43	2.71	5.40	18.12	5.3	227	827.2

gaps with FFN [19]. The Sintel final pass is more difficult than the clean pass as it includes image degradations such as motion blur, defocus blur, and atmospheric effects [3]. The KITTI dataset is more challenging real-world dataset with large displacements. It becomes even more challenging when estimating the optical flow in patches. With those difficulties, our model shows 1.47% lower accuracy compared to FFN in terms of F1-all metric. Ours is better than SPyNet [28] and comparable with LiteFlownetX [13] while being computationally more efficient. In comparison to LiteFlownet [13], PWCNet [29], RAFT-Small [30], VCN-small, VCN [35], Flow1D [33] and RAFT [30], ours shows reasonable speed-accuracy compromise.

4.5 Real dataset with ground-truth optical flow

We also test our model on GyroFlow real dataset (GOF) [20] that has ground-truth (GT) annotations for the optical flow. The resolution of the images is 800×600 and there are four types of scenes: regular, fog, rain, and dark. In Table 4, we report the inference results of PF-HT, RAFT and FFN on regular scenes only. We use patches of size 256×256 for PF-HT. As shown in Table 4, PF-HT has a similar performance as the FFN model, while not being far from the SOTA method RAFT with 12 iterations. Figure 3 shows qualitative comparisons between our 2-stage pipeline and the FFN and RAFT methods.

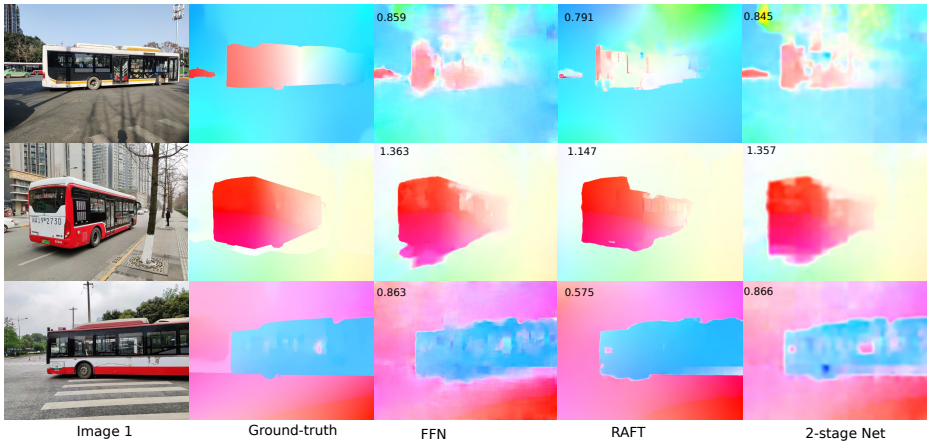

Fig. 3. Qualitative results on GOF dataset [20] for FFN, RAFT, and our 2-stage pipeline. The values indicate the EPE for each example.

Table 4. Performance comparison on GOF clean dataset in terms of EPE. For the 2-stage pipeline, the EPE value is based on patches of size 256×256, while the value in the brackets is for patches of size 128×128. Both FFN and RAFT (12 iters.) models have the full resolution as input.

Method	EPE
Ours	1.09 (1.09)
FFN	1.08
RAFT	0.78

4.6 Real dataset without ground-truth

In addition to the GOF dataset, we collected a dataset of high resolution images with 2K (1080×1920) and 4K (2048×3840) resolutions using Huawei P40 Pro phone. The dataset consists of scenes where almost all the motion is due to the camera movements. Overall, the dataset contains 8 video sequences (six 2K videos and two 4K videos). We call this dataset RealP40. Since the dataset does not contain GT optical flow annotations, we use Root Mean Squared Error (RMSE) for evaluation [21]. It is defined as one minus the normalized cross-correlation (NCC) of two pixels in neighbourhood π of size 5×5:

$$RMSE(I_1, I_2) = \sqrt{\frac{1}{N_v} \sum_{\pi} (1 - NCC(p1, p2))} \tag{5}$$

where N_v is the total number of valid pixels in $I1$ and $I2$ (*i.e.* ignoring the black area resulted from the out-of-boundary pixels when warping $I2$), and $p1$ and $p2$ are the corresponding pixels in I1 and I2, respectively.

Table 5. Performance comparison on RealP40-2K dataset in terms of RMSE. The patch size used is 128×128, while FFN and Flow1D process the full resolution. The timing and memory requirements are based on 4K resolution.

Method	RMSE	Time (ms)	Min Memory (MB)
Ours	0.6642	120	13
FFN [19]	0.6681	213	1882
Flow1D [33]	0.6665	2885	5181

The evaluation of our model against FFN [19] and Flow1D [33] on RealP40-2K is shown in Table 5. For a 4K resolution, ours is faster while achieving similar performance. Note that RAFT produces out-of-memory on a GPU with 32GB of memory for 4K resolution, and requires more than 8GB of memory for 2K images (larger than the 8GB memory on our RTX2070) due to the huge cost volume it computes. RAFT has an alternative implementation where they compute the cost volume on-demand basis. However, it is slow in turn. Qualitative comparisons between our PF and FFN [19] on 4K videos are provided in the supplemental material. An advantage of the patch-based approach is that it requires memory as small as the patch size. For instance, suppose that we have only 1 GB in a GPU memory (such as in mobile phones or micro-controllers for example), and assuming that the memory can take up to 10 patches at a time of size 256×256 pixels from the 120 patches of a 4K image, the device can process the whole image in 12 sequential iterations. At the same time, the models processing the full resolution image can not work because of the limited memory resources. In Table 5, we show the minimum memory required (*i.e.* the peak memory consumed during inference) to process an image by a corresponding method. Ours has lower memory requirements in comparison to [19, 33].

5 Conclusion

We proposed a two-stage pipeline for optical flow estimation of high-resolution images. The first stage takes as input a low-resolution version of the image pair. The pre-flow from the first stage is then fed to the second stage and utilized for warping the target frame feature patches. The second stage estimates the residual motion on the small patches of the input images. We showed that such pre-flow and refinement helps building a shallower model while achieving a comparable result with the FFN model. Thanks to the patch-based approach, the high-resolution image can be processed in limited memory and computational resources like mobile phones. The comparison with SOTA solution RAFT on Sintel and GOF datasets showed that the 2-stage solution can offer a reasonable accuracy while being significantly computationally lighter and more feasible for high-resolution images.

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
2. Bleyer, M., Rhemann, C., Rother, C.: Patchmatch stereo-stereo matching with slanted support windows. In: Bmvc. vol. 11, pp. 1–11 (2011)
3. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: European conference on computer vision. pp. 611–625. Springer (2012)
4. Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., Shi, W.: Real-time video super-resolution with spatio-temporal networks and motion compensation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4778–4787 (2017)
5. Chen, Q., Koltun, V.: Full flow: Optical flow estimation by global optimization over regular grids. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4706–4714 (2016)
6. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)
7. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2758–2766 (2015)
8. Hofinger, M., Bulò, S.R., Porzi, L., Knapitsch, A., Pock, T., Kontschieder, P.: Improving optical flow on a pyramid level. In: European Conference on Computer Vision. pp. 770–786. Springer (2020)
9. Horn, B.K., Schunck, B.G.: Determining optical flow. Artificial intelligence **17**(1-3), 185–203 (1981)
10. Hossain, M.A., Cannons, K., Jang, D., Cuzzolin, F., Xu, Z.: Video-based crowd counting using a multi-scale optical flow pyramid network. In: Proceedings of the Asian Conference on Computer Vision (2020)
11. Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K.C., Qin, H., Dai, J., Li, H.: Flowformer: A transformer architecture for optical flow. arXiv preprint arXiv:2203.16194 (2022)
12. Hui, T.W., Loy, C.C.: Liteflownet3: Resolving correspondence ambiguity for more accurate optical flow estimation. In: European Conference on Computer Vision. pp. 169–184. Springer (2020)
13. Hui, T.W., Tang, X., Loy, C.C.: Liteflownet: A lightweight convolutional neural network for optical flow estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8981–8989 (2018)
14. Hui, T.W., Tang, X., Loy, C.C.: A lightweight optical flow cnn—revisiting data fidelity and regularization. IEEE transactions on pattern analysis and machine intelligence **43**(8), 2555–2569 (2020)
15. Hur, J., Roth, S.: Iterative residual refinement for joint optical flow and occlusion estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5754–5763 (2019)
16. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2462–2470 (2017)

17. Jiang, S., Campbell, D., Lu, Y., Li, H., Hartley, R.: Learning to estimate hidden motions with global motion aggregation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9772–9781 (2021)
18. Jiang, S., Lu, Y., Li, H., Hartley, R.: Learning optical flow from a few matches. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16592–16600 (2021)
19. Kong, L., Shen, C., Yang, J.: Fastflownet: A lightweight network for fast optical flow estimation. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 10310–10316. IEEE (2021)
20. Li, H., Luo, K., Liu, S.: Gyroflow: Gyroscope-guided unsupervised optical flow learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12869–12878 (2021)
21. Li, S., Yuan, L., Sun, J., Quan, L.: Dual-feature warping-based motion model estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4283–4291 (2015)
22. Li, Z., Dekel, T., Cole, F., Tucker, R., Snavely, N., Liu, C., Freeman, W.T.: Learning the depths of moving people by watching frozen people. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4521–4530 (2019)
23. Lin, K., Jiang, N., Liu, S., Cheong, L.F., Do, M., Lu, J.: Direct photometric alignment by mesh deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2405–2413 (2017)
24. Liu, S., Yuan, L., Tan, P., Sun, J.: Steadyflow: Spatially smooth optical flow for video stabilization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4209–4216 (2014)
25. Liu, Y.L., Lai, W.S., Yang, M.H., Chuang, Y.Y., Huang, J.B.: Hybrid neural fusion for full-frame video stabilization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2299–2308 (2021)
26. Lu, Y., Valmadre, J., Wang, H., Kannala, J., Harandi, M., Torr, P.: Devon: Deformable volume network for learning optical flow. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2705–2713 (2020)
27. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3061–3070 (2015)
28. Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4161–4170 (2017)
29. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8934–8943 (2018)
30. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020)
31. Wang, L., Guo, Y., Liu, L., Lin, Z., Deng, X., An, W.: Deep video super-resolution using hr optical flow estimation. IEEE Transactions on Image Processing **29**, 4323–4336 (2020)
32. Wen, R., Zhao, P.: A medium-shifted splitting iteration method for a diagonal-plus-toeplitz linear system from spatial fractional schrödinger equations. Boundary Value Problems **2018**(1), 1–17 (2018)

33. Xu, H., Yang, J., Cai, J., Zhang, J., Tong, X.: High-resolution optical flow from 1d attention and correlation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10498–10507 (2021)
34. Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Tao, D.: Gmflow: Learning optical flow via global matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8121–8130 (2022)
35. Yang, G., Ramanan, D.: Volumetric correspondence networks for optical flow. Advances in neural information processing systems **32** (2019)
36. Yu, J., Ramamoorthi, R.: Learning video stabilization using optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8159–8167 (2020)
37. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime tv-l 1 optical flow. In: Joint pattern recognition symposium. pp. 214–223. Springer (2007)
38. Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional networks. In: 2010 IEEE Computer Society Conference on computer vision and pattern recognition. pp. 2528–2535. IEEE (2010)
39. Zhang, F., Woodford, O.J., Prisacariu, V.A., Torr, P.H.: Separable flow: Learning motion cost volumes for optical flow estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10807–10817 (2021)
40. Zhang, T., Zhang, H., Li, Y., Nakamura, Y., Zhang, L.: Flowfusion: Dynamic dense rgb-d slam based on optical flow. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 7322–7328. IEEE (2020)
41. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018)
42. Zheng, Z., Nie, N., Ling, Z., Xiong, P., Liu, J., Wang, H., Li, J.: Dip: Deep inverse patchmatch for high-resolution optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8925–8934 (2022)