FlowFormer: A Transformer Architecture for Optical Flow

Zhaoyang Huang[1,3*], Xiaoyu Shi[1,3*], Chao Zhang[2], Qiang Wang[2], Ka Chun Cheung[3], Hongwei Qin[4], Jifeng Dai[4], and Hongsheng Li[1†]

[1]Multimedia Laboratory, The Chinese University of Hong Kong
[2]Samsung Telecommunication Research
[3]NVIDIA AI Technology Center [4]SenseTime Research
{drinkingcoder@link, xiaoyushi@link, hsli@ee}.cuhk.edu.hk

Abstract. We introduce optical Flow transFormer, dubbed as FlowFormer, a transformer-based neural network architecture for learning optical flow. FlowFormer tokenizes the 4D cost volume built from an image pair, encodes the cost tokens into a cost memory with alternate-group transformer (AGT) layers in a novel latent space, and decodes the cost memory via a recurrent transformer decoder with dynamic positional cost queries. On the Sintel benchmark, FlowFormer achieves 1.144 and 2.183 average end-ponit-error (AEPE) on the clean and final pass, a 17.6% and 11.6% error reduction from the best published result (1.388 and 2.47). Besides, FlowFormer also achieves strong generalization performance. Without being trained on Sintel, FlowFormer achieves 0.95 AEPE on the Sintel training set clean pass, outperforming the best published result (1.29) by 26.9%.

1 Introduction

Optical flow targets at estimating per-pixel correspondences between a source image and a target image, in the form of a 2D displacement field. In many downstream video tasks, such as action recognition [45, 36, 60], video inpainting [28, 49, 13], video super-resolution [30, 5, 38], and frame interpolation [50, 33, 20], optical flow serves as a fundamental component providing dense correspondences as valuable clues for prediction.

A general assumption adopted in optical flow estimation is that the appearance of corresponding locations in the two images induced from optical flows remains unchanged. Traditionally, optical flow is modeled as an optimization problem that maximizes visual similarities between cross-image corresponding locations with regularization terms. With the rapid development of deep learning and emerging training data, this field has been significantly advanced by deep convolutional neural network-based methods. The recent methods compute costs (i.e. visual similarities) between feature pairs, upon which flows are regressed.

*Zhaoyang Huang and Xiaoyu Shi assert equal contributions.
†Corresponding author: Hongsheng Li

Most successful architecture designs in optical flow are achieved via better designs of cost encoding and decoding. PWC-Net [42] and RAFT [46] are two recent representative deep learning-based methods. PWC-Net [42] builds hierarchical local cost volumes with warped features and progressively estimates flows from such local costs. RAFT [46] forms an $H \times W \times H \times W$ 4D cost volume that measures similarities between all pairs of pixels of the $H \times W$ image pair and iteratively retrieves local costs within local windows for regressing flow residuals.

Recently, transformers have attracted much attention for their ability of modeling long-range relations, which can benefit optical flow estimation. Perceiver IO [24] is the pioneering work that learns optical flow regression with a transformer-based architecture. However, it directly operates on pixels of image pairs and ignores the well-established domain knowledge of encoding visual similarities to costs for flow estimation. It thus requires a large number of parameters and $\sim 80\times$ training examples to capture the desired input-output mapping. We therefore raise a question: can we enjoy both advantages of transformers and the cost volume from the previous milestones? Such a question calls for designing novel transformer architectures for optical flow estimation that can effectively aggregate information from the cost volume. In this paper, we introduce the novel optical Flow TransFormer (FlowFormer) to address this challenging problem.

FlowFormer adopts an encoder-decoder architecture for cost volume encoding and decoding. After building a 4D cost volume, FlowFormer consists of two main components: 1) a cost volume encoder that embeds the 4D cost volume into a latent cost space and fully encodes the cost information in such a space, and 2) a recurrent cost decoder that estimates flows from the encoded latent cost features. Compared with previous works, the main characteristic of our FlowFormer is to adapt the transformer architectures to effectively process cost volumes, which are compact yet rich representations widely explored in optical flow estimation communities, for estimating accurate optical flows.

A naive strategy to transform the 4D cost volume with transformers is directly tokenizing the 4D cost volume and applying transformers. However, such a strategy needs to use thousands of tokens, which is computationally unbearable. To tackle this challenge, we propose two key designs in our cost encoder. We propose a two-step tokenization: 1) converting each of the 2D cost maps, which records visual similarities between one source pixel and all target pixels, from the 4D cost volume into patches as commonly done in transformer networks, and 2) further projecting cost-map patches of each cost map into K latent cost tokens. In this way, the $H \times W \times H \times W$ 4D cost volume can be transformed into $H \times W \times K$ tokens. Secondly, instead of performing self-attention among all tokens, we alternatively conduct attention over tokens within the same cost map and tokens across different cost maps. In other words, an interweaving stack of aggregations of latent cost tokens belonging to the same source pixel and those across different source pixels. Combining these two designs, FlowFormer encodes the cost volume into compact and globally aware latent cost tokens, dubbed as the *cost memory*.

Classical transformer architectures, such as DETR [4], decodes information from the encoded memory via stacked cross-attention layers. In contrast to them, inspired by RAFT, our cost decoder adopts only a recurrent attention layer that formulates the cost decoding as a recurrent query process with dynamic positional cost queries: based on current estimated flows, we query the cost memory for regressing the flow residuals. In each iteration, we compute the corresponding positions in the target image for all source pixels according to current flows and then dynamically update positional cost queries with such positions. Then, we fetch cost features from the cost memory via cross-attention and use a shared gated recurrent unit (GRU) head for residual flow regression. Moreover, RAFT only utilizes a shallow CNN as the image feature encoder. We find that our FlowFormer can be benefited from using an ImageNet-pretrained transformer backbone.

Our contributions can be summarized as fourfold. 1) We propose a novel transformer-based neural network architecture, FlowFormer, for optical flow estimation, which achieves state-of-the-art flow estimation performance. 2) We design a novel cost volume encoder, effectively aggregating cost information into compact latent cost tokens. 3) We propose a recurrent cost decoder that recurrently decodes cost features with dynamic positional cost queries to iteratively refine the estimated optical flows. 4) To the best of our knowledge, we validate for the first time that an ImageNet-pretrained transformer can benefit the estimation of optical flow.

2 Related Work

Optical Flow. Traditionally, optical flow was modeled as an optimization problem that maximizes visual similarity between image pairs with regularizations [17, 1, 2, 40]. Major improvements in this era came from better designs of similarity and regularization terms. The rise of deep neural networks significantly advanced this field. FlowNet [12] was the first end-to-end convolutional network for optical flow estimation. Its successive work, FlowNet2.0 [23], adopted a stacked architecture with warping operation, performing on par with state-of-the-art (SOTA) methods. Then a series of works, represented by SpyNet [37], PWC-Net [42, 43], LiteFlowNet [21, 22] and VCN [53], employed coarse-to-fine and iterative estimation methodology. These models inherently suffered from missing small fast-motion objects in coarse stage. To remedy this issue, Teed and Deng [46] proposed RAFT [46], which performs optical flow estimation in a coarse-and-fine (i.e. multi-scale search window in each iteration) and recurrent manner. Based on RAFT architecture, many works [25, 48, 26, 57, 16] were proposed to either reduce the computational costs or improve the flow accuracy. Recently, optical flow was extended to more challenging settings, such as low-light [61], foggy [52], and lighting variations [18].

Among these explorations, visual similarity is computed by the correlation of high dimensional features encoded by a convolutional neural network, and the cost volume that contains visual similarity of pixels pairs acts as a core

component supporting optical flow estimation. However, their cost information utilization lacks effectiveness. We propose FlowFormer that aggregates the cost volume in a latent space with transformers [47]. Perceiver IO [24] pioneered the use of transformers [47, 11, 4] that is able to establish long-range relationship in optical flow and achieved state-of-the-art performance. It ignored the cost volume, showing the strong expressive capacity of transformer architecture at the cost of $\sim 80\times$ training examples. In contrast, we propose to keep cost volume as a compact similarity representation and push search space to the extreme by globally aggregating similarity information via a transformer architecture. Such global encoding operation is especially beneficial in the hard cases of large displacement and occlusion.

Transformers for Computer Vision. Transformers achieved great success in Natural Language Processing [47, 9, 10], which inspired the development of self-attention for image classification [34, 11, 8]. Since then, transformer-based architectures has been introduced into many other vision tasks, such as detection [4], point cloud processing [15, 58], image restoration [6, 31], video inpainting[56, 32], visual grounding [54], etc, and achieves state-of-the-art in most tasks. The appealing performance is generally attributed to the long-range modeling capacity, which is also a desired property in optical flow estimation. One of the challenges that vision transformers are faced with is the large number of visual tokens because the computational cost quadratically increases along with the token number. Twins [8] proposed a spatially separable self-attention (SS Self-Attention) layer that propagates information over tokens arranged in a 2D plane. We also adopt the SS Self-Attention in the cost volume encoder to propagate information inter-cost-maps. Perceiver IO [24] proposed a general transformer backbone, which although requires a large amount of parameters, achieves state-of-the-art optical flow performance. Visual correspondence tasks [44, 19, 7, 27, 51] is a main stream in computer vision. Recently, transformers also lead a trend in such tasks [39, 44, 7, 27], which is more related to ours.

3 Method

The task of optical flow estimation requires to output a per-pixel displacement field $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that maps every 2D location $\mathbf{x} \in \mathbb{R}^2$ of a source image $\mathbf{I}_s$ to its corresponding 2D location $\mathbf{p} = \mathbf{x}+\mathbf{f}(\mathbf{x})$ of a target image $\mathbf{I}_t$. To take advantage of the recent vision transformer architectures as well as the 4D cost volumes widely utilized by previous CNN-based optical flow estimation methods, we propose FlowFormer, a transformer-based architecture that encodes and decodes the 4D cost volume to achieve accurate optical flow estimation. In Fig. 1, we show the overview architecture of FlowFormer, which processes the 4D cost volumes from siamese features with two main components: 1) a cost volume encoder that encodes the 4D cost volume into a latent space to form cost memory, and 2) a cost memory decoder for predicting a per-pixel displacement field based on the encoded cost memory and contextual features.

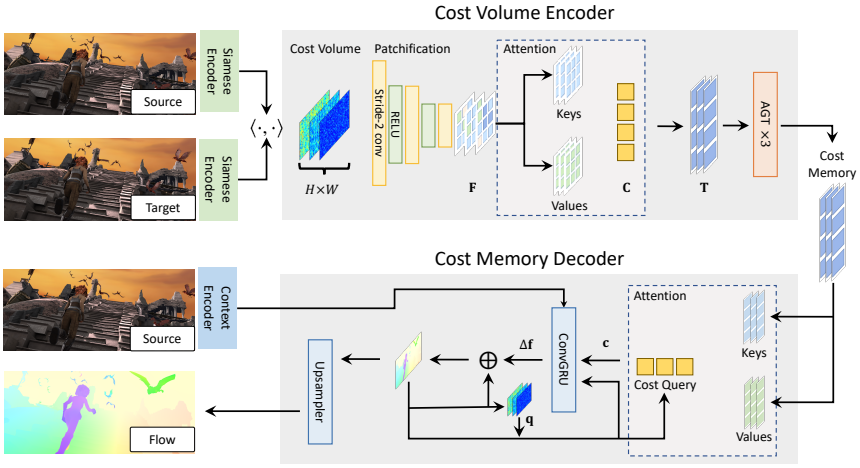

Fig. 1: Architecture of FlowFormer. FlowFormer estimates optical flow in three steps: 1) building a 4D cost volume from image features. 2) A cost volume encoder that encodes the cost volume into the cost memory. 3) A recurrent transformer decoder that decodes the cost memory with the source image context features into flows.

3.1 Building the 4D Cost Volume

A backbone vision network is used to extract an $H \times W \times D_f$ feature map from an input $H_I \times W_I \times 3$ RGB image, where typically we set $(H, W) = (H_I/8, W_I/8)$. After extracting the feature maps of the source image and the target image, we construct an $H \times W \times H \times W$ 4D cost volume by computing the dot-product similarities between all pixel pairs between the source and target feature maps.

3.2 Cost Volume Encoder

To estimate optical flows, the corresponding positions in the target image of source pixels need to be identified based on source-target visual similarities encoded in the 4D cost volume. The built 4D cost volume can be viewed as a series of 2D cost maps of size $H \times W$, each of which measures visual similarities between a single source pixel and all target pixels. We denote source pixel $\mathbf{x}$'s cost map as $\mathbf{M}_\mathbf{x} \in \mathbb{R}^{H \times W}$. Finding corresponding positions in such cost maps is generally challenging, as there might exist repeated patterns and non-discriminative regions in the two images. The task becomes even more challenging when only considering costs from a local window of the map, as previous CNN-based optical flow estimation methods do. Even for estimating a single source pixel's accurate displacement, it is beneficial to take its contextual source pixels' cost maps into consideration.

To tackle this challenging problem, we propose a transformer-based cost volume encoder that encodes the whole cost volume into a *cost memory*. Our cost

volume encoder consists of three steps: 1) cost map patchification, 2) cost patch token embedding, and 3) cost memory encoding. We elaborate the details of the three steps as follows.

Cost map patchification. Following existing vision transformers, we patchify the cost map $\mathbf{M_x} \in \mathbb{R}^{H\times W}$ of each source pixel $\mathbf{x}$ with strided convolutions to obtain a sequence of cost patch embeddings. Specifically, given an $H \times W$ cost map, we first pad zeros at its right and bottom sides to make its width and height multiples of 8. The padded cost map is then transformed by a stack of three stride-2 convolutions followed by ReLU into a feature map $\mathbf{F_x} \in \mathbb{R}^{\lceil H/8\rceil\times\lceil W/8\rceil\times D_p}$. Each feature in the feature map stands for an 8×8 patch in the input cost map. The three convolutions have output channels of $D_p/4$, $D_p/2$, D_p, respectively.

Patch Feature Tokenization via Latent Summarization. Although the patchification results in a sequence of cost patch feature vectors for each source pixel, the number of such patch features is still large and hinders the efficiency of information propagation among different source pixels. Actually, a cost map is highly redundant because only a few high costs are most informative. To obtain more compact cost features, we further summarize the patch features $\mathbf{F_x}$ of each source pixel $\mathbf{x}$ via K latent codewords $\mathbf{C} \in \mathbb{R}^{K\times D}$. Specifically, the latent codewords query each source pixel's cost-patch features to further summarize each cost map into K latent vectors of D dimensions via the dot-product attention mechanism. The latent codewords $\mathbf{C} \in \mathbb{R}^{K\times D}$ are randomly initialized, updated via back-propagation, and shared across all source pixels. The latent representations $\mathbf{T_x}$ for summarizing $\mathbf{F_x}$ are obtained as

$$\begin{aligned}\mathbf{K_x} &= \mathrm{Conv}_{1\times1}\left(\mathrm{Concat}(\mathbf{F_x}, \mathrm{PE})\right),\\ \mathbf{V_x} &= \mathrm{Conv}_{1\times1}\left(\mathrm{Concat}(\mathbf{F_x}, \mathrm{PE})\right),\\ \mathbf{T_x} &= \mathrm{Attention}(\mathbf{C}, \mathbf{K_x}, \mathbf{V_x}).\end{aligned} \tag{1}$$

Before projecting the cost-patch features $\mathbf{F_x}$ to obtain keys $\mathbf{K_x}$ and values $\mathbf{V_x}$, the patch features are concatenated with a sequence of positional embeddings $\mathrm{PE} \in \mathbb{R}^{\lceil H/8\rceil\times\lceil W/8\rceil\times D_p}$. Given a 2D position $\mathbf{p}$, we encode it into a positional embedding of length D_p following COTR [27]. Finally, the cost map of the source pixel $\mathbf{x}$ can be summarized into K latent representations $\mathbf{T_x} \in \mathbb{R}^{K\times D}$ by conducting multi-head dot-product attention with the queries, keys, and values. Generally, $K \times D \ll H \times W$ and the latent summarizations $\mathbf{T_x}$ therefore provides more compact representations than each $H \times W$ cost map for each source pixel $\mathbf{x}$. For all source pixels in the image, there are a total of $(H \times W)$ 2D cost maps. Their summarized representations can consequently be converted into a latent 4D cost volume $\mathbf{T} \in \mathbb{R}^{H\times W\times K\times D}$.

Attention in the latent cost space. The aforementioned two stages transform the original 4D cost volume into a latent and compact 4D cost volume $\mathbf{T}$. However, it is still too expensive to directly apply self-attention over all the vectors in the 4D volume because the computational cost quadratically increases with the number of tokens. As shown in Fig. 2, we propose an alternate-group transformer layer (AGT) that groups the tokens in two mutually orthogonal manners

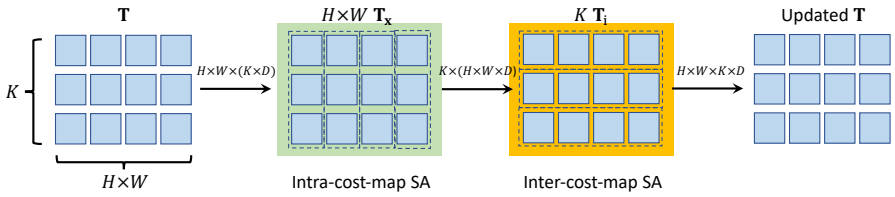

Fig. 2: Alternate-Group Transformer Layer. The alternate-group transformer layer (AGT) alternatively groups tokens in $\mathbf{T}$ into $H \times W$ groups that contains K tokens ($\mathbf{T_x}$) and K groups that contains $H \times W$ tokens ($\mathbf{T}_i$), and encode tokens inside groups via self-attention and ss self-attention [8] respectively.

and apply attentions in the two groups alternatively, which reduces the cost of attention while still being able to propagate information among all tokens.

The first grouping is conducted for each source pixel, i.e., each $\mathbf{T_x} \in \mathbb{R}^{K \times D}$ forms a group and the self-attention is conducted within each group.

$$\mathbf{T_x} = \text{FFN}(\text{Self-Attention}(\mathbf{T_x}(1), \ldots, \mathbf{T_x}(K)) \quad \text{for all } \mathbf{x} \text{ in } \mathbf{I}_s, \tag{2}$$

where $\mathbf{T_x}(i)$ denotes the i-th latent representation for encoding the source pixel $\mathbf{x}$'s cost map. After the self-attention is conducted between all K latent tokens for each source pixel $\mathbf{x}$, updated $\mathbf{T_x}$ are further transformed by a feed-forward network (FFN) and then re-organized back to form the updated 4D cost volume $\mathbf{T}$. Both the self-attention and FFN sub-layers adopt the common designs of residual connection and layer normalization of transformers. This self-attention operation propagates the information within each cost map and we name it as intra-cost-map self-attention.

The second way groups all the latent cost tokens $\mathbf{T} \in \mathbb{R}^{H \times W \times K \times D}$ into K groups according to the K different latent representations. Each group would therefore have $(H \times W)$ tokens of dimension D for information propagation in the spatial domain via the spatially separable self-attention (SS-SelfAttention) proposed in Twins [8],

$$\mathbf{T}_i = \text{FFN}(\text{SS-SelfAttention}(\mathbf{T}_i)) \quad \text{for } i = 1, 2, \ldots, K, \tag{3}$$

where we slightly abuse the notation and denote $\mathbf{T}_i \in \mathbb{R}^{(H \times W) \times D}$ as the i-th group. The updated $\mathbf{T}_i$'s are then re-organized back to obtain the updated 4D latent cost volume $\mathbf{T}$. Moreover, visually similar source pixels should have coherent flows, which has been validated by previous methods [25, 7]. Thus, we integrate appearance affinities between different source pixels into SS-SelfAttention via concatenating the source image's context features $\mathbf{t}$ with the cost tokens when generating queries and keys. We call this layer inter-cost-map self-attention layer as it propagates information of cost volume across different source pixels. Note that these two operations are different from CATs [7], which augmented correlations 'intra' a level of cost map and 'inter' multi-level correlation layers.

The above self-attention operations' parameters are shared across different groups and they are sequentially operated to form the proposed alternate-group attention layer. By stacking the alternate-group transformer layer multiple times, the latent cost tokens can effectively exchange information across source pixels and across latent representations to better encode the 4D cost volume. In this way, our cost volume encoder transforms the $H \times W \times H \times W$ 4D cost volume to $H \times W \times K$ latent tokens of length D. We call the final $H \times W \times K$ tokens as the *cost memory*, which is to be decoded for optical flow estimation.

3.3 Cost Memory Decoder for Flow Estimation

Given the cost memory encoded by the cost volume encoder, we propose a cost memory decoder to predict optical flows. Since the original resolution of the input image is $H_I \times W_I$, we estimate optical flow at the $H \times W$ resolution and then upsample the predicted flows to the original resolution with a learnable convex upsampler [46]. However, in contrast to previous vision transformers that seek abstract semantic features, optical flow estimation requires recovering dense correspondences from the cost memory. Inspired by RAFT [46], we propose to use cost queries to retrieve cost features from the cost memory and iteratively refine flow predictions with a recurrent attention decoder layer.

Cost memory aggregation. For predicting the flows of the $H \times W$ source pixels, we generate a sequence of $(H \times W)$ cost queries, each of which is responsible for estimating the flow of a single source pixel via co-attention on the cost memory. To generate the cost query $\mathbf{Q_x}$ for a source pixel $\mathbf{x}$, we first compute its corresponding location in the target image given its current estimated flow $\mathbf{f}(\mathbf{x})$ as $\mathbf{p} = \mathbf{x} + \mathbf{f}(\mathbf{x})$. We then retrieve a local 9×9 cost-map patch $\mathbf{q_x} = \mathrm{Crop}_{9\times 9}(\mathbf{M_x}, \mathbf{p})$ by cropping costs inside the 9×9 local window centered at $\mathbf{p}$ on the cost map $\mathbf{M_x}$. The cost query $\mathbf{Q_x}$ is then formulated based on the features $\mathrm{FFN}(\mathbf{q_x})$ that encoded from the local costs $\mathbf{q_x}$ and $\mathbf{p}$'s positional embedding $\mathrm{PE}(\mathbf{p})$, which can aggregate information from source pixel $\mathbf{x}$'s cost memory $\mathbf{T_x}$ via cross-attention,

$$\begin{aligned} \mathbf{Q_x} &= \mathrm{FFN}\left(\mathrm{FFN}(\mathbf{q_x}) + \mathrm{PE}(\mathbf{p})\right), \\ \mathbf{K_x} &= \mathrm{FFN}\left(\mathbf{T_x}\right), \quad \mathbf{V_x} = \mathrm{FFN}\left(\mathbf{T_x}\right), \\ \mathbf{c_x} &= \mathrm{Attention}(\mathbf{Q_x}, \mathbf{K_x}, \mathbf{V_x}). \end{aligned} \tag{4}$$

The cross-attention summarizes information from the cost memory for each source pixel to predict its flow. As $\mathbf{Q_x}$ is dynamically updated in terms of the fed position at each iteration, we call it as dynamic positional cost query. We note that keys and values can be generated at the beginning and re-used in subsequent iterations, which saves computation as a benefit of our recurrent decoder.

Recurrent flow prediction. Our cost decoder iteratively regresses flow residuals $\Delta\mathbf{f}(\mathbf{x})$ to refine the flow of each source pixel $\mathbf{x}$ as $\mathbf{f}(\mathbf{x}) \leftarrow \mathbf{f}(\mathbf{x}) + \Delta\mathbf{f}(\mathbf{x})$. We adopt a ConvGRU module and follow the similar design to that in GMA-RAFT [25] for flow refinement. However, the key difference of our recurrent module is the use of cost queries to adaptively aggregate information from the

cost memory for more accurate flow estimation. Specifically, at each iteration, the ConvGRU unit takes as input the concatenation of retrieved cost features and cost-map patch $\text{Concat}(\mathbf{c_x}, \mathbf{q_x})$, the source-image context feature $\mathbf{t_x}$ from the context network, and the current estimated flow $\mathbf{f}$, and outputs the predicted flow residuals as follows,

$$\Delta\mathbf{f}(\mathbf{x}) = \text{ConvGRU}(\text{Concat}(\mathbf{c_x}, \mathbf{q_x}), \mathbf{t_x}, \mathbf{f}(\mathbf{x})). \tag{5}$$

The flows generated at each iteration are unsampled to the size of the source image via a convex upsampler following [46] and supervised by ground-truth flows at all recurrent iterations with increasing weights.

4 Experiment

We evaluate our FlowFormer on the Sintel [3] and the KITTI-2015 [14] benchmarks. Following previous works, we train FlowFormer on FlyingChairs [12] and FlyingThings [35], and then respectively finetune it for Sintel and KITTI benchmark. Flowformer achieves state-of-the-art performance on both benchmarks.

Experimental setup. We use the average end-point-error (AEPE) and F1-All(%) metric for evaluation. The AEPE computes mean flow error over all valid pixels. The F1-all, which refers to the percentage of pixels whose flow error is larger than 3 pixels or over 5% of length of ground truth flows. The Sintel dataset is rendered from the same model but in two passes, i.e. clean pass and final pass. The clean pass is rendered with smooth shading and specular reflections. The final pass uses full rendering settings including motion blur, camera depth-of-field blur, and atmospheric effects.

Implementation details. The image feature encoder of our final FlowFormer is chosen as the first two stages of ImageNet-pretrained Twins-SVT [8], which encodes an image into $D_f = 256$-channel feature map of 1/8 image size. The cost volume encoder patchifies each cost map to a $D_p = 64$-channel feature map and further summarizes the feature map to $N = 8$ cost tokens of $K = 128$ dimensions. Then, the cost volume encoder encodes the cost tokens with 3 AGT layers. Following previous optical flow training procedure [25], we pre-train FlowFormer on FlyingChairs [12] for 120k iterations with a batch size of 8, and on FlyingThings [35] for 120k iterations with a batch size of 6 (denoted as 'C+T'). After pre-training, we finetune FlowFormer on the data combined from FlyingThings, Sintel, KITTI-2015, and HD1K [29] (denoted as 'C+T+S+K+H') for 120k iterations with a batch size of 6. To achieve the best performance on the KITTI benchmark, we also further finetune FlowFormer on the KITTI-2015 for 50k iterations with a batch size of 6. We use the one-cycle learning rate scheduler. The highest learning rate is set as 2.5×10^{-4} on FlyingChairs and 1.25×10^{-4} on the other training sets. As positional encodings used in transformers are sensitive to image size, we crop the image pairs for flow estimation and tile them to obtain complete flows following Perceiver IO [24]. We use fixed Gaussian weights for tile, which will be detailed in the supplementary materials.

Training Data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)
		Clean	Final	F1-epe	F1-all	Clean	Final	F1-all
A+S+K+H	Perceiver IO [24]	-	-	-	-	1.81	2.42	4.98
	PWC-Net [42]	-	-	-	-	2.17	2.91	5.76
	RAFT [46]	-	-	-	-	1.95	2.57	4.23
C+T	HD3 [55]	3.84	8.77	13.17	24.0	-	-	-
	LiteFlowNet [21]	2.48	4.04	10.39	28.5	-	-	-
	PWC-Net [42]	2.55	3.93	10.35	33.7	-	-	-
	LiteFlowNet2 [22]	2.24	3.78	8.97	25.9	-	-	-
	S-Flow [57]	1.30	2.59	4.60	15.9			
	RAFT [46]	1.43	2.71	5.04	17.4	-	-	-
	FM-RAFT [26]	1.29	2.95	6.80	19.3	-	-	-
	GMA [25]	1.30	2.74	4.69	17.1	-	-	-
	Ours	**0.95**	**2.35**	**4.09**	**14.72**	-	-	-
C+T+S+K+H	LiteFlowNet2 [22]	(1.30)	(1.62)	(1.47)	(4.8)	3.48	4.69	7.74
	PWC-Net+ [43]	(1.71)	(2.34)	(1.50)	(5.3)	3.45	4.60	7.72
	VCN [53]	(1.66)	(2.24)	(1.16)	(4.1)	2.81	4.40	6.30
	MaskFlowNet [59]	-	-	-	-	2.52	4.17	6.10
	S-Flow [57]	(0.69)	(1.10)	(0.69)	(1.60)	1.50	2.67	**4.64**
	RAFT [46]	(0.76)	(1.22)	(0.63)	(1.5)	1.94	3.18	5.10
	FM-RAFT [26]	(0.79)	(1.70)	(0.75)	(2.1)	1.72	3.60	6.17
	GMA [25]	-	-	-	-	1.40	2.88	5.15
	Ours	(0.48)	(0.74)	(0.53)	(1.11)	**1.14**	**2.18**	4.68
	RAFT* [46]	(0.77)	(1.27)	-	-	1.61	2.86	-
	GMA* [25]	(0.62)	(1.06)	(0.57)	(1.2)	1.39	2.47	-

Table 1: Experiments on Sintel [3] and KITTI [14] datasets. * denotes that the methods use the warm-start strategy [46], which relies on previous image frames in a video. 'A' denotes the autoflow dataset. 'C + T' denotes training only on the FlyingChairs and FlyingThings datasets. '+ S + K + H' denotes finetuning on the combination of Sintel, KITTI, and HD1K training sets. Our FlowFormer achieves best generalization performance (C+T) and ranks 1st on the Sintel benchmark (C+T+S+K+H).

4.1 Quantitative Experiment

We evaluate FlowFormer on the well-known Sintel and KITTI benchmarks as shown in Tab. 1. GMA [25], an improved version of RAFT [46], is the most competitive flow estimation method at present. After being trained on FlyingChairs and FlyingThings, we evaluate the generalization performance of FlowFormer on the training set of Sintel and KITTI-2015. By further finetuning FlowFormer on the combination of HD1K, Sintel and KITTI training sets, we compare the dataset-specific accuracy of optical flow models. Autoflow [41] is a dataset that provides training data covering various challenging visual disturbance, but its training code is not released yet.

Generalization performance. We train FlowFormer on the FlyingChairs and FlyingThings (C+T), and evaluate it on the training set of Sintel and KITTI-2015. This settings evaluates the generalization performance of optical flow models. FlowFormer ranks 1st among all compared methods on both benchmarks.

FlowFormer achieves 0.95 and 2.35 on the clean and final pass of Sintel. On the KITTI-2015 training set, FlowFormer achieves 4.09 F1-epe and 14.72 F1-all. Compared to GMA, FlowFormer reduces 26.9% and 14.2% errors on Sintel clean and final, and 13.9% errors on KITTI-2015 F1-all, which shows its extraordinary generalization performance.

Sintel benchmark. We finetune the pretrained FlowFormer on the combination of training data of FlyingThings, HD1K, Sintel and KITTI-2015, and then evaluate it on the Sintel test set. FlowFormer achieves 1.14 and 2.18 on the Sintel clean and final, 17.6% and 11.6% lower error compared to GMA*, which ranks both 1st on the Sintel benchmark. It is noteworthy that RAFT* and GMA* use the warm-start strategy that requires image sequences while FlowFormer does not. Compared with GMA, which also does not use the warm-start, FlowFormer obtains 18.6% and 24.3% error reduction. RAFT trained on the autoflow dataset (A+S+K+H) significantly outperforms RAFT trained on the C+T+S+K+H on final pass because autoflow provides training image pairs that are more challenging. We believe training FlowFormer with autoflow can achieve better accuracy but it is not released yet.

KITTI-2015 benchmark. We further finetune the FlowFormer on the KITTI-2015 training set after the Sintel finetuning stage and evaluate it on the KITTI test set. FlowFormer achieves 4.68, ranking 2nd on the KITTI-2015 benchmark. S-Flow [57] obtains slightly smaller error than FlowFormer on KITTI (-0.85%), which, however, is significantly worse on Sintel (31.6% and 22.5% larger error on clean and final pass). S-Flow finds corresponding points by computing the coordinate expectation weighted by refined cost maps. Images in the KITTI dataset are captured in urban traffic scenes, which contains objects that are mostly rigid. Flows on rigid objects are rather simple, which is easier for cost-based coordinate expectation, but the assumption can be easily violated in non-rigid scenarios such as Sintel.

4.2 Qualitative Experiment

We visualize flows that estimated by our FlowFormer and GMA of three examples in Fig. 3 to qualitatively show how FlowFormer outperforms GMA. As transformers can encode the cost information at a large perceptive field, FlowFormer can distinguish overlapping objects via contextual information and thus reduce the leakage of flows over boundaries. Compared with GMA, the flows that are estimatd by FlowFormer on boundaries of the bamboo and the human body are more precise and clear. Besides, FlowFormer can also recover motion details that are ignored by GMA, such as the hair and the holes on the box.

4.3 Ablation Study

We conduct a series of ablation experiments in Tab. 2. We start from RAFT as the baseline, which directly regresses residual flows with the multi-level cost retrieval (MCR) decoder, and gradually replace its components with our proposed components. We first replace RAFT's MCR decoder with the latent cost

(a) Input (b) FlowFormer (Ours) (c) GMA

Fig. 3: Qualitative comparison on the Sintel test set. FlowFormer greatly reduces the flow leakage around object boundaries (pointed by red arrows) and clearer details (pointed by blue arrows).

tokenization (LCT) part of our encoder and the cost memory decoder (CMD) (denoted as 'MCR→LCT+CMD'). Note that our cost memory decoder cannot be used alone on top of the 4D cost volume of RAFT because of the too large number of tokens. It must be combined with our latent cost tokens ($\mathbf{T_x}$ from Eq. (1)). Encoding $K = 8$ latent tokens of $D = 128$ dimensions for each source pixel achieves the best performance. Based on LCT+CMD with $K = 8$ and $D = 128$, we replace RAFT's CNN image feature encoder with Twins-SVT (denoted as 'CNN→Twins'). We then further add attention layers of the proposed cost volume encoder to encode and update latent cost tokens. The proposed Alternate-Group Transformer (AGT) layer consists of two types of attention, i.e., intra-cost-map attention and inter-cost-map attention. We first add a single intra-cost-map attention layer (denoted as '+Intra.'), and then add the inter-cost-map attention (denoted as 'AGT×1 (+Intra.+Inter.)', which is equivalent to adding a single AGT layer. We then test on increasing the number of AGT layers to 2 and 3. Following RAFT, all models are trained on FlyingChairs [12] with 100k iterations and FlyingThings [35] with 60k iterations, and then evaluated on the training set of Sintel [3] and KITTI-2015 [14].

MCR → LCT+MCD. The number of latent tokens K and token dimension D determine how much cost volume information the cost tokens can encode. From $K = 4, D = 32$ to $K = 8, D = 128$, the AEPE decreases because the cost tokens summarizes more cost map information and benefits the residual flow regression. The latent cost tokens are capable of summarizing whole-image information and our MCD can absorb interested information from them through co-attention, while the MCR decoder of RAFT only retrieves multi-level costs inside flow-guided local windows. Therefore, even without our AGT layers in our encoder, LCT+MCD still shows better performance than MCR decoder of RAFT.

Experiment	Method	Sintel (train)		KITTI-15 (train)		Params.
		Clean	Final	F1-epe	F1-all	
baseline	RAFT	1.53	2.99	5.73	18.29	5.3M
MCR→LCT+CMD	$K = 4$, $D = 32$	1.66	2.93	5.60	19.67	5.5M
	$K = 8$, $D = 32$	1.58	2.90	5.50	18.71	5.5M
	$K = 8$, $D = 128$	1.44	2.80	5.22	17.64	5.6M
CNN → Twins	CNN	1.44	2.80	5.22	17.64	5.6M
	Twins from Scratch	1.44	2.86	5.38	17.58	14.0M
	Pretrained Twins	1.29	2.72	4.82	16.16	14.0M
Cost Encoding	None	1.29	2.72	4.82	16.16	14.0M
	+Intra.	1.29	2.89	4.74	15.71	14.1M
	AGT×1 (+Intra.+Inter.)	1.20	2.85	4.57	15.46	15.2M
	AGT×2	1.16	2.66	4.70	16.01	16.4M
	AGT×3	1.10	2.57	4.45	15.15	17.6M

Table 2: Ablation study. We gradually change one component of the RAFT at a time to obtain our FlowFormer model. MCR→LCT+CMD: replacing RAFT's decoder with OUR latent cost tokens + cost memory decoder. CNN→Twins: replacing RAFT's CNN encoder with Twins-SVT transformer. Cost Encoding: adding intra-cost-map and inter-cost-map to form an Alternate-Group Transformer layer in the encoder. 3 AGT layers are used in our final model.

CNN vs. Transformer Image Encoder. In the CNN→Twins experiment, the AEPE of Twins trained from scratch is marginally worse than CNN, but the ImageNet-pretraining is beneficial, because Twins is a transformer architecture with larger receptive field and model capacity, which requires more training examples for sufficient training.

Cost Encoding. In the cost volume encoder, we encode and update the latent cost tokens with an intra-cost-map attention operation and an inter-cost-map attention operation. The two operations form an Alternate-Group Transformer (AGT) layer. Then we gradually increase the number of AGT layers to 3. From no attention layer to AGT×3, the errors gradually decrease, which demonstrates that encoding latent cost tokens with our AGT layers benefits flow estimation.

FlowFormer vs. GMA. We train all the models with the settings of GMA. The full version of FlowFormer has 18.2M parameters, which is larger than GMA. One of the causes is that FlowFormer uses the first two stages of ImageNet-pretrained Twins-SVT as the image feature encoder while GMA uses a CNN. We present an experiment to compare FlowFormer and GMA with aligned settings in Tab. 3. We first provide a small version of FlowFormer using GMA's CNN image encoder and also set $K = 4$, $D = 32$, and AGT×1. Although the smaller version of FlowFormer (denoted as 'Ours (small)') has a significant performance drop compared to the full version of FlowFormer, it still outperforms GMA in terms of all metrics. We also design two enhanced GMA models and compare them with the full version of FlowFormer to show that the performance improvements are not simply derived from adding more parameters. The first one is denoted as 'GMA-L', a large version of GMA and the second one is denoted as 'GMA-Twins' which also adopts the pretrained Twins as the image

Method	Sintel (train)		KITTI-15 (train)		parameters
	Clean	Final	F1-epe	F1-all	
GMA [25]	1.30 (+30%)	2.74 (+12%)	4.69 (+15%)	17.1 (+16%)	5.9M
Ours (small)	1.20 (+20%)	2.64 (+8%)	4.57 (+12%)	16.62 (+13%)	6.2M
GMA-L [25]	1.33 (+33%)	2.56 (+4%)	4.40 (+8%)	15.93 (+8%)	17.0M
GMA-Twins [25]	1.15 (+15%)	2.73 (+11%)	4.98 (+22%)	16.82 (+14%)	14.2M
Ours	**1.00**	**2.45**	**4.09**	**14.72**	18.2M

Table 3: FlowFormer v.s. GMA. Ours (small) is a small version of FlowFormer and uses the CNN image feature encoder of GMA. GMA-L is a large version of GMA. GMA-Twins replace its CNN image feature encoder with pre-trained Twins. (+x%) indicates that this model obtains x% larger error than ours.

encoder. In this experiment, we train all models on FlyingChairs with 120k iterations and FlyingThings with 120k iterations. Similar to reducing RAFT to RAFT (small) [46], GMA-L enlarges GMA by doubling feature channels, which has 17M parameters, comparable to FlowFormer. However, its performance degrades in Sintel clean, a 33% larger error than FlowFormer. GMA-Twins replaces the CNN image encoder with the shallow Image-Net pre-trained Twins-SVT as FlowFormer does. The largest improvement of GMA-Twins upon GMA is on the Sintel clean, but it still has a 15% larger error than FlowFormer. GMA-Twins does not lead to significant error reduction on other metrics and is even worse on the KITTI-15. In conclusion, the performance improvement of FlowFormer is not derived from more parameters but the novel design of the architecture.

5 Conclusion

We have proposed FlowFormer, a Transformer-based architecture for optical flow estimation. FlowFormer summarizes the $H \times W \times H \times W$ 4D cost volume built from a pair of images as $H \times W \times K$ tokens of length D, and then efficiently and effectively encodes the cost tokens via the alternate-group transformer (AGT). Thanks to such design, the generated cost memory is able to grasp essential information over the cost volume and obtain compact cost features. Finally, the cost memory decoder absorbs cost information from the cost memory with dynamic positional cost queries, which gets rid of the limitation of local windows, for residual flow regression. To our best knowledge, FlowFormer is the first method that deeply integrates transformers with cost volumes for optical flow estimation. Thanks to the compact cost tokens and long-range relation modeling ability of transformers, FlowFormer achieves state-of-the-art accuracy and shows strong cross-dataset generalization.

Acknowledgements. Hongsheng Li is also a Principal Investigator of Centre for Perceptual and Interactive Intelligence Limited (CPII). This work is supported in part by CPII, in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants (Nos. 14204021, 14207319), in part by CUHK Strategic Fund.

References

1. Black, M.J., Anandan, P.: A framework for the robust estimation of optical flow. In: 1993 (4th) International Conference on Computer Vision. pp. 231–236. IEEE (1993)
2. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. International journal of computer vision **61**(3), 211–231 (2005)
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. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
5. Chan, K.C., Wang, X., Yu, K., Dong, C., Loy, C.C.: Basicvsr: The search for essential components in video super-resolution and beyond. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4947–4956 (2021)
6. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., Gao, W.: Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12299–12310 (2021)
7. Cho, S., Hong, S., Jeon, S., Lee, Y., Sohn, K., Kim, S.: Cats: Cost aggregation transformers for visual correspondence. Advances in Neural Information Processing Systems **34** (2021)
8. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., Shen, C.: Twins: Revisiting spatial attention design in vision transformers. arXiv preprint arXiv:2104.13840 (2021)
9. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
12. 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)
13. Gao, C., Saraf, A., Huang, J.B., Kopf, J.: Flow-edge guided video completion. In: European Conference on Computer Vision. pp. 713–729. Springer (2020)
14. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
15. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (2021)
16. 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)

17. Horn, B.K., Schunck, B.G.: Determining optical flow. Artificial intelligence **17**(1-3), 185–203 (1981)
18. Huang, Z., Pan, X., Xu, R., Xu, Y., Zhang, G., Li, H., et al.: Life: Lighting invariant flow estimation. arXiv preprint arXiv:2104.03097 (2021)
19. Huang, Z., Zhou, H., Li, Y., Yang, B., Xu, Y., Zhou, X., Bao, H., Zhang, G., Li, H.: Vs-net: Voting with segmentation for visual localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6101–6111 (2021)
20. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Rife: Real-time intermediate flow estimation for video frame interpolation. arXiv preprint arXiv:2011.06294 (2020)
21. 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)
22. 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)
23. 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)
24. Jaegle, A., Borgeaud, S., Alayrac, J.B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al.: Perceiver io: A general architecture for structured inputs & outputs. arXiv preprint arXiv:2107.14795 (2021)
25. Jiang, S., Campbell, D., Lu, Y., Li, H., Hartley, R.: Learning to estimate hidden motions with global motion aggregation. arXiv preprint arXiv:2104.02409 (2021)
26. 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)
27. Jiang, W., Trulls, E., Hosang, J., Tagliasacchi, A., Yi, K.M.: Cotr: Correspondence transformer for matching across images. arXiv preprint arXiv:2103.14167 (2021)
28. Kim, D., Woo, S., Lee, J.Y., Kweon, I.S.: Deep video inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5792–5801 (2019)
29. Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., 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. pp. 19–28 (2016)
30. Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H.: Deep laplacian pyramid networks for fast and accurate super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 624–632 (2017)
31. Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., Timofte, R.: Swinir: Image restoration using swin transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1833–1844 (2021)
32. Liu, R., Deng, H., Huang, Y., Shi, X., Lu, L., Sun, W., Wang, X., Dai, J., Li, H.: Fuseformer: Fusing fine-grained information in transformers for video inpainting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14040–14049 (2021)
33. Liu, X., Liu, H., Lin, Y.: Video frame interpolation via optical flow estimation with image inpainting. International Journal of Intelligent Systems **35**(12), 2087–2102 (2020)

34. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021)
35. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: 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. pp. 4040–4048 (2016)
36. Piergiovanni, A., Ryoo, M.S.: Representation flow for action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9945–9953 (2019)
37. 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)
38. Sajjadi, M.S., Vemulapalli, R., Brown, M.: Frame-recurrent video super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6626–6634 (2018)
39. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: Superglue: Learning feature matching with graph neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4938–4947 (2020)
40. Sun, D., Roth, S., Black, M.J.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. International Journal of Computer Vision **106**(2), 115–137 (2014)
41. Sun, D., Vlasic, D., Herrmann, C., Jampani, V., Krainin, M., Chang, H., Zabih, R., Freeman, W.T., Liu, C.: Autoflow: Learning a better training set for optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10093–10102 (2021)
42. 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)
43. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: 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)
44. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: Detector-free local feature matching with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8922–8931 (2021)
45. Sun, S., Kuang, Z., Sheng, L., Ouyang, W., Zhang, W.: Optical flow guided feature: A fast and robust motion representation for video action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1390–1399 (2018)
46. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
48. 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)
49. Xu, R., Li, X., Zhou, B., Loy, C.C.: Deep flow-guided video inpainting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3723–3732 (2019)

50. Xu, X., Siyao, L., Sun, W., Yin, Q., Yang, M.H.: Quadratic video interpolation. Advances in Neural Information Processing Systems **32** (2019)
51. Xu, Y., Lin, K.Y., Zhang, G., Wang, X., Li, H.: Rnnpose: Recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization (2022)
52. Yan, W., Sharma, A., Tan, R.T.: Optical flow in dense foggy scenes using semi-supervised learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13259–13268 (2020)
53. Yang, G., Ramanan, D.: Volumetric correspondence networks for optical flow. Advances in neural information processing systems **32**, 794–805 (2019)
54. Yang, L., Xu, Y., Yuan, C., Liu, W., Li, B., Hu, W.: Improving visual grounding with visual-linguistic verification and iterative reasoning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9499–9508 (2022)
55. Yin, Z., Darrell, T., Yu, F.: Hierarchical discrete distribution decomposition for match density estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6044–6053 (2019)
56. Zeng, Y., Fu, J., Chao, H.: Learning joint spatial-temporal transformations for video inpainting. In: European Conference on Computer Vision. pp. 528–543. Springer (2020)
57. 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)
58. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021)
59. Zhao, S., Sheng, Y., Dong, Y., Chang, E.I., Xu, Y., et al.: Maskflownet: Asymmetric feature matching with learnable occlusion mask. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6278–6287 (2020)
60. Zhao, Y., Man, K.L., Smith, J., Siddique, K., Guan, S.U.: Improved two-stream model for human action recognition. EURASIP Journal on Image and Video Processing **2020**(1), 1–9 (2020)
61. Zheng, Y., Zhang, M., Lu, F.: Optical flow in the dark. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6749–6757 (2020)