

# FlowFormer: A Transformer Architecture and Its Masked Cost Volume Autoencoding for Optical Flow

Zhaoyang Huang\*, Xiaoyu Shi\*, Chao Zhang, Qiang Wang, Yijin Li, Hongwei Qin,  
Jifeng Dai, Xiaogang Wang, and Hongsheng Li

**Abstract**—This paper introduces a novel transformer-based network architecture, FlowFormer, along with the Masked Cost Volume AutoEncoding (MCVA) for pretraining it to tackle the problem of optical flow estimation. FlowFormer tokenizes the 4D cost-volume built from the source-target image pair and iteratively refines flow estimation with a cost-volume encoder-decoder architecture. The cost-volume encoder derives a cost memory with alternate-group transformer (AGT) layers in a latent space and the decoder recurrently decodes flow from the cost memory with dynamic positional cost queries. On the Sintel benchmark, FlowFormer architecture achieves 1.16 and 2.09 average end-point-error (AEPE) on the clean and final pass, a 16.5% and 15.5% error reduction from the GMA (1.388 and 2.47). MCVA enhances FlowFormer by pretraining the cost-volume encoder with a masked autoencoding scheme, which further unleashes the capability of FlowFormer with unlabeled data. This is especially critical in optical flow estimation because ground truth flows are more expensive to acquire than labels in other vision tasks. MCVA improves FlowFormer all-sided and FlowFormer+MCVA ranks 1st among all published methods on both Sintel and KITTI-2015 benchmarks and achieves the best generalization performance. Specifically, FlowFormer+MCVA achieves 1.07 and 1.94 AEPE on the Sintel benchmark, leading to 7.76% and 7.18% error reductions from FlowFormer.

**Index Terms**—Optical Flow, Masked Autoencoding, Transformer.

## 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 [1], [2], [3], video inpainting [4], [5], [6], video super-resolution [7], [8], [9], [10], and frame interpolation [11], [12], [13], [14], [15], 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. Most successful architecture designs in optical flow are achieved

via better designs of cost encoding and decoding. PWC-Net [16] and RAFT [17] are two recent representative deep learning-based methods. PWC-Net [16] builds hierarchical local cost volumes with warped features and progressively estimates flows from such local costs. RAFT [17] forms an  $H \times W \times H \times W$  4D cost volume that measures similarities between all pairs of pixels of the pair of  $H \times W$  images and iteratively retrieves local costs within local windows for regressing flow residuals.

Recently, transformers have attracted much attention for their ability to model long-range relations. Network pre-training further unleashes the capacity of transformers with unlabeled data, which is fascinating for optical flow because the ground truth collection for optical flow is expensive. Perceiver IO [18] 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 cost volumes or maps for flow estimation. It thus requires a large number of parameters and training examples to capture the desired input-output mapping. Besides, how to pretrain Perceiver IO still remains a challenge. We therefore raise a question: can we design a transformer architecture along with a pertaining technique for optical flow? We propose FlowFormer as an answer to this question. FlowFormer contains two main novel techniques: 1) an optical Flow Transformer (FlowFormer) architecture, and 2) a Masked Cost Volume Autoencoding (MCVA) technique to pretrain the proposed FlowFormer.

FlowFormer directly adopts an ImageNet-pretrained Twins [19] transformer to encode image features, and then

---

- Zhaoyang Huang and Xiaoyu Shi assert equal contributions,
- Zhaoyang Huang, Xiaoyu Shi, Xiaogang Wang, and Hongsheng Li are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong SAR. E-mail: {drinkingcoder@link, xiaoyushi@link, xgwang@ee, hsl@ee}.cuhk.edu.hk,
- Chao Zhang and Qiang Wang are with Samsung Telecommunication Research
- Yijin Li is with the State Key Lab of CAD&CG in Zhejiang University.
- Hongwei Qin is with SenseTime Research, Shenzhen, China.
- Jifeng Dai is with the Department of Electronic Engineering, Tsinghua University. Email: daijifeng@tsinghua.edu.cn

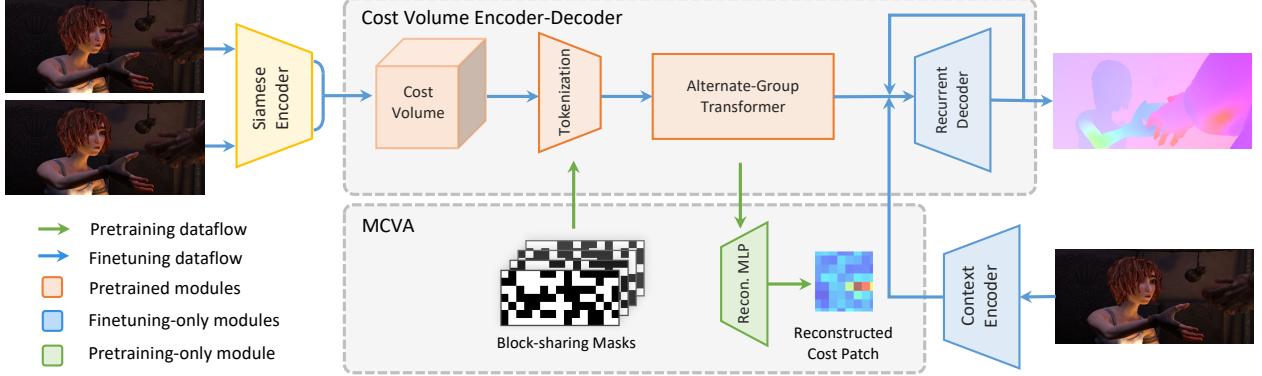


Fig. 1: **Overview of FlowFormer with Masked Cost Volume Autoencoding (MCVA).** We propose a transformer-based neural network architecture, dubbed as FlowFormer, along with a Masked Cost Volume Autoencoding (MCVA) for optical flow estimation. During pretraining, a portion of cost values are masked and the cost-volume encoder is required to reconstruct masked cost patches. In the supervised finetuning stage, FlowFormer recurrently decodes flow from the encoded cost features.

uses an asymmetric encoder-decoder architecture for cost-volume encoding and decoding (Fig. 1). After building a 4D cost volume from image features, 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 lightweight 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 interleaving 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 [20] 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.

We also propose masked cost-volume autoencoding (MCVA), a self-supervised pretraining scheme to enhance the cost-volume encoding on top of the FlowFormer framework. We are inspired by the recent success of masked autoencoding, such as BERT [21] in NLP and MAE [22] in computer vision. The key idea of masked autoencoding is masking a portion of input data, and requiring networks to learn high-level representation for masked contents reconstruction. Following this paradigm, we pretrain the cost-volume encoder by reconstructing the masked cost volume with another reconstruction head, as shown in Fig. 1. However, it is non-trivial to adapt the masked autoencoding strategy to learn a better cost-volume encoder for optical flow estimation, because of the two following reasons. Firstly, the cost volume might contain redundancy and the cost maps (cost values between a source-image pixel to all target-image pixels) of neighboring source-image pixels are highly correlated. Randomly masking cost values [22] can lead to information leakage and makes the model biased towards aggregating local information. Secondly, existing masked autoencoding methods target at reconstructing masked content randomly selected from fixed locations. This suffices to pretrain general-purpose single-image encoder in other fields. However, the cost-volume encoder of FlowFormer is deeply coupled with the follow-up recurrent decoder, which demands cost information of long range at flexible locations.

To tackle the aforementioned issues, we introduce two task-specific designs. Firstly, instead of randomly masking the cost volume, we partition source pixels into blocks

and let source pixels within the same block share a common mask on their cost maps. This strategy, termed block-sharing masking, prevents the cost-volume encoder from reconstructing masked cost values by simply copying from neighboring source pixels' cost maps. Secondly, to mimic the decoding process in finetuning and thus avoid pretraining-finetuning discrepancy, we propose a novel pre-text reconstruction task as shown in Fig. 1: small cost patches (of shape  $9 \times 9$ ) are randomly cropped from the cost maps to retrieve features from the cost-volume encoder, aiming to reconstruct larger cost map patches (of shape  $15 \times 15$ ) centered at the same locations. This is in line with the decoding process of FlowFormer in the finetuning stage. Besides, we empirically show that the image encoder, upon which the cost volume is built, should be frozen during pretraining to avoid training collapse.

In essence, the proposed masked cost-volume autoencoding (MCVA) has unique designs compared with conventional MAE methods, which encourages the cost-volume encoder 1) to construct high-level holistic representation of the cost volume, more effectively encoding long-range information, 2) to reason about occluded (*i.e.*, masked) information by aggregating faithful unmasked costs, and 3) to decode task-specific feature (*i.e.*, larger cost patches at required locations) to better align the pretraining process with that of the finetuning.

Our contributions can be summarized as sixfold. 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) We propose the masked cost-volume autoencoding scheme to better pretrain the cost-volume encoder of FlowFormer. 5) We propose task-specific masking strategy and reconstruction pre-text task to mitigate pretraining-finetuning discrepancy, fully taking advantage of the learned representations from pretraining. 6) With the proposed pretraining technique, FlowFormer+MCVA obtains all-sided improvements over FlowFormer, setting new state-of-the-art performance on public benchmarks.

## 2 RELATED WORKS

**Optical Flow.** Traditionally, optical flow was modeled as an optimization problem that maximizes visual similarity between image pairs with regularizations [23], [24], [25], [26]. 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 [27] was the first end-to-end convolutional network for optical flow estimation. Its successive work, FlowNet2.0 [28], 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 [29], PWC-Net [16], [30], LiteFlowNet [31], [32] and VCN [33], 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 [17] proposed RAFT [17], 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 [19], [34], [35], [36], [37] 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 [38], foggy [39], and lighting variations [40].

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 [41]. Perceiver IO [18] pioneered the use of transformers [20], [41], [42] that is able to establish long-range relationships 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. There are also other optical flow methods [43], [44] enhancing image feature encoder with transformers but their performance is not competitive.

**Transformers for Computer Vision.** Transformers achieved great success in Natural Language Processing [21], [41], [45], which inspired the development of self-attention for image classification [42], [46], [47]. Since then, transformer-based architectures have been introduced into many other vision tasks, such as detection [20], point cloud processing [48], [49], image restoration [50], [51], video inpainting [52], [53], 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 [47] 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 [18] proposed a general transformer backbone, which although requires a large amount of parameters, achieves state-of-the-art optical flow performance. Visual correspondence tasks [54], [55], [56], [57] is a main stream in computer vision. Recently, transformers also lead a trend in such tasks, which is more related to ours. For the sparse matching problem, SuperGlue [58] introduced a flexible context aggregation mechanism based on attention. LoFTR [54] utilized transformer to remove feature detector and further improved performance. For semantic correspondence task, CATs [56] applied transformer over the cost volume to explore global consensus. However, it directly applied attention on raw cost maps, making it

restricted to low-resolution input images. In contrast, our FlowFormer projects the cost volume into a latent space that is more compact and efficient. COTR [57] reformulated dense correspondence problem as functional mapping and took coordinates as input. But its accuracy highly relied on repeated "zoom-in" query operation, which hinders its use for full-image query as required by optical flow estimation.

**Masked Autoencoding (MAE).** As a self-supervised learning technique, MAE, e.g., BERT [21], achieved great success in NLP. Based on transformers, they mask a portion of the input tokens and require the models to predict the missing content from the reserved tokens. Pretraining with MAE encourages transformers to build effective long-range feature relationships. Recently, transformers also stream into the computer vision area, such as image recognition [42], [46], [47], video inpainting [52], [53], optical flow [43], point cloud recognition [48], [49]. By breaking the limitations that convolution can only model local features, transformers present a significant performance gap compared to the previous counterparts. Pretraining with MAE is also introduced to these modalities, e.g., image [22], [59], [60], [61], video [62], point cloud [63], [64], [65], [66]. These works show that MAE effectively releases the transformer power and do not require extra labeled data. FlowFormer presents a transformer-based cost-volume encoder and achieves state-of-the-art accuracy. We further propose the masked cost-volume autoencoding to pretrain the cost-volume encoder on a video dataset, which further unleashes the power of the transformer-based cost-volume encoder.

### 3 METHOD

Our FlowFormer contains two main parts: an optical Flow Transformer (FlowFormer) and a Masked Cost Volume Autoencoding (MCVA) technique to pretrain FlowFormer. We will elaborate them in this section.

#### 3.1 FlowFormer Architecture

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. 2, 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.

##### 3.1.1 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.

##### 3.1.2 Cost-Volume Encoder

To estimate optical flows, the corresponding positions of source images in the target images 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 the 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 pathification, 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  $\mathbf{M}_{\mathbf{x}} \in \mathbb{R}^{H \times W}$  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  $\mathbf{F}_{\mathbf{x}} \in \mathbb{R}^{\lceil H/8 \rceil \times \lceil W/8 \rceil \times D_p}$ . Each feature in the feature map stands for an  $8 \times 8$  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  $\mathbf{F}_{\mathbf{x}}$  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  $\mathbf{T}_{\mathbf{x}}$  for summarizing  $\mathbf{F}_{\mathbf{x}}$  are obtained as

$$\begin{aligned}\mathbf{K}_{\mathbf{x}} &= \text{Conv}_{1 \times 1}(\text{Concat}(\mathbf{F}_{\mathbf{x}}, \text{PE})), \\ \mathbf{V}_{\mathbf{x}} &= \text{Conv}_{1 \times 1}(\text{Concat}(\mathbf{F}_{\mathbf{x}}, \text{PE})), \\ \mathbf{T}_{\mathbf{x}} &= \text{Attention}(\mathbf{C}, \mathbf{K}_{\mathbf{x}}, \mathbf{V}_{\mathbf{x}}).\end{aligned}\quad (1)$$

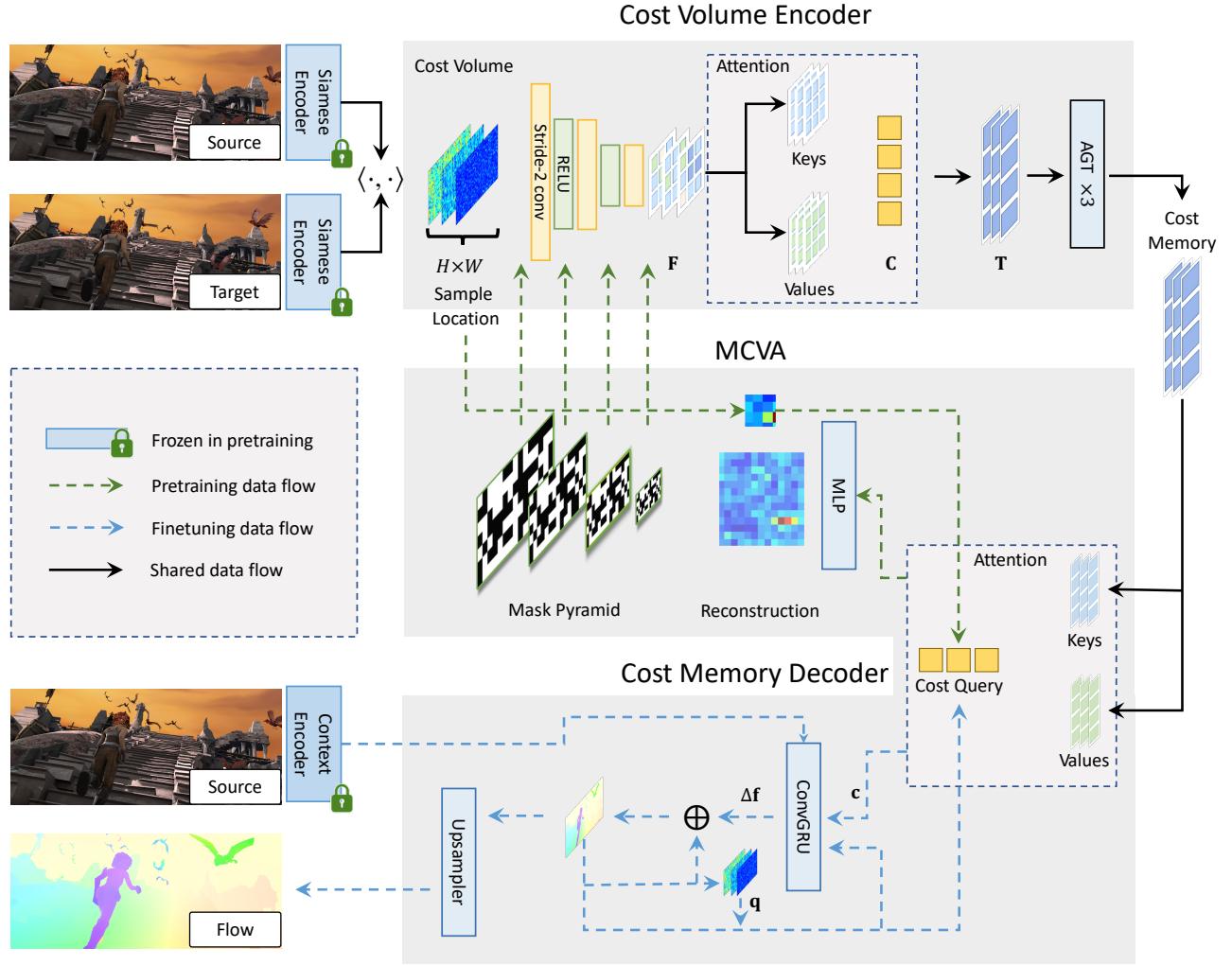


Fig. 2: **FlowFormer with MCVA.** FlowFormer estimates optical flow in three steps: 1) building a 4D cost volume from image features; 2) A cost-volume encoder that encodes the cost volume into the cost memory; 3) A recurrent transformer decoder that decodes the cost memory with the source image context features into flows. During MCVA pertaining, FlowFormer freezes the image and context encoders, block-wisely masks the cost volume, and learns to reconstruct larger cost patches from small cost patches to pretrain cost-volume encoder.

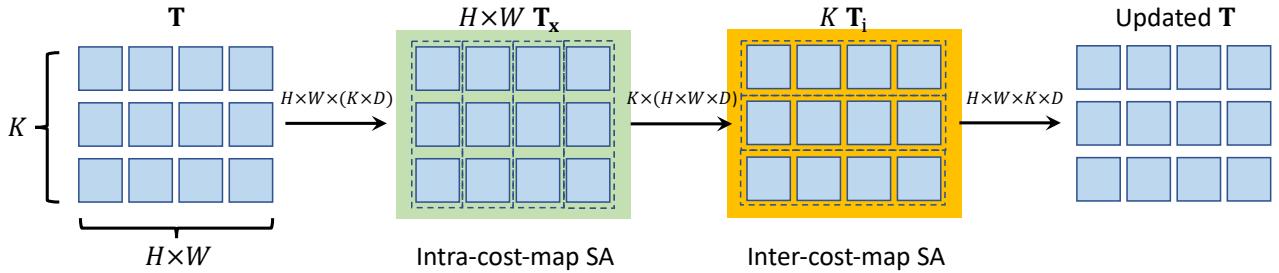


Fig. 3: **Alternate-Group Transformer Layer.** The alternate-group transformer layer (AGT) alternatively groups tokens in  $\mathbf{T}$  into  $H \times W$  groups that contains  $K$  tokens ( $\mathbf{T}_x$ ) and  $K$  groups that contains  $H \times W$  tokens ( $\mathbf{T}_i$ ), and encode tokens inside groups via self-attention and ss self-attention [47] respectively.

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  $\mathbf{PE} \in \mathbb{R}^{[H/8] \times [W/8] \times D_p}$ . Given a 2D position  $\mathbf{p}$ , we encode it

into a positional embedding of length  $D_p$  following COTR [57]. Finally, the cost map of the source pixel  $x$  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  $x$ . 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. 3, we propose an alternate-group transformer layer (AGT) that groups the tokens in two mutually orthogonal manners and apply self-attentions in the two groups alternatively, which reduces the computational cost of self-attention while still being able to propagate information between 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.

$$\mathbf{T}_x = \text{FFN}(\text{Self-Attention}(\mathbf{T}_x(1), \dots, \mathbf{T}_x(K))) \quad (2)$$

for all  $x$  in  $I_s$ ,

where  $\mathbf{T}_x(i)$  denotes the  $i$ -th latent representation for encoding the source pixel  $x$ 's cost map. After the self-attention is conducted between all  $K$  latent tokens for each source pixel  $x$ , 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 [47],

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

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 [19], [56]. 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 is responsible to propagate information of cost volume across different source pixels.

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.

### 3.1.3 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 [17]. 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 [17], 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  $x$ , we first compute its corresponding location in the target image given its current estimated flow  $\mathbf{f}(x)$  as  $\mathbf{p}_x = \mathbf{x} + \mathbf{f}(x)$ . We then retrieve a local  $9 \times 9$  cost-map patch  $\mathbf{q}_x = \text{Crop}_{9 \times 9}(\mathbf{M}_x, \mathbf{p}_x)$  by cropping costs inside the  $9 \times 9$  local window centered at  $\mathbf{p}_x$  on the cost map  $\mathbf{M}_x$ . The cost query  $\mathbf{Q}_x$  is then formulated based on the features  $\text{FFN}(\mathbf{q}_x)$  that encoded from the local costs  $\mathbf{q}_x$  and  $\mathbf{p}_x$ 's positional embedding  $\text{PE}(\mathbf{p}_x)$ , which can aggregate information from source pixel  $x$ 's cost memory  $\mathbf{T}_x$  via cross-attention,

$$\begin{aligned} \mathbf{Q}_x &= \text{FFN}(\text{FFN}(\mathbf{q}_x) + \text{PE}(\mathbf{p}_x)), \\ \mathbf{K}_x &= \text{FFN}(\mathbf{T}_x), \quad \mathbf{V}_x = \text{FFN}(\mathbf{T}_x), \\ \mathbf{c}_x &= \text{Attention}(\mathbf{Q}_x, \mathbf{K}_x, \mathbf{V}_x). \end{aligned} \quad (4)$$

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  $\Delta\mathbf{f}(x)$  to refine the flow of each source pixel  $x$  as  $\mathbf{f}(x) \leftarrow \mathbf{f}(x) + \Delta\mathbf{f}(x)$ . We adopt a ConvGRU module and follow the similar design to that in GMA-RAFT [19] 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,

$$\Delta\mathbf{f}(x) = \text{ConvGRU}(\text{Concat}(\mathbf{c}_x, \mathbf{q}_x), \mathbf{t}_x, \mathbf{f}(x)). \quad (5)$$

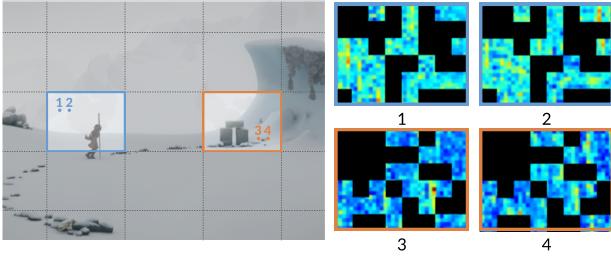


Fig. 4: **Block-sharing masking.** Source pixels within the same block share a common mask.

The flows generated at each iteration are unsampled to the size of the source image via a convex upsampler following [17] and supervised by ground-truth flows at all recurrent iterations with increasing weights.

### 3.2 Masked Cost Volume Autoencoding

To better unleash the potential of FlowFormer architecture, as presented in Fig. 2, we propose a masked cost-volume autoencoding (MCVA) scheme to pretrain the cost-volume encoder of FlowFormer framework. The key of general masked autoencoding methods is to mask a portion of data and encourage the network to reconstruct the masked tokens from visible ones. Due to the redundant nature of the cost volume and the original FlowFormer architecture being incompatible with masks, naively adopting this paradigm to pretrain the cost-volume encoder leads to inferior performance. Our proposed MCVA tackles the challenge and conducts masked autoencoding with three key components: a proper masking strategy on the cost volume, modifying FlowFormer architecture to accommodate masks, and a novel pre-text reconstruction task supervising the pretraining process.

In this subsection, we first introduce the masking strategy, dubbed as block-sharing masking, and then show the masked cost-volume tokenization that makes the cost-volume encoder compatible with masks. Coupling these two designs prevents the masked autoencoding from being hindered by information leakage in pretraining. Finally, we present the pre-text cost reconstruction task, mimicing the decoding process in finetuning to pretrain the cost-volume encoder.

#### 3.2.1 Block-sharing Cost Volume Masking

A properly designed masking scheme is required to conduct autoencoding of the masked cost volume. For each source pixel  $\mathbf{x}$ , we need to create a binary mask  $\mathbf{B}_{\mathbf{x}} \in \{0, 1\}^{H \times W}$  to its cost map  $\mathbf{A}_{\mathbf{x}} \in \mathbb{R}^{H \times W}$ , where 0 indicates masking (*i.e.*, removing) cost values from the masked locations. Naturally, neighboring source pixels' cost maps are highly correlated. Randomly masking neighboring pixels' cost maps might cause information leakage, *i.e.*, masked cost values might be easily reconstructed by copying the cost values from neighboring source pixels' cost maps.

To prevent such an over-simplified learning process, we propose a block-sharing masking strategy (Fig. 4). We

partition source pixels into non-overlapping blocks in each iteration. All source pixels belonging to the same block share a common mask for masked region reconstruction. In this way, neighboring source pixels are unlikely to copy each other's cost maps to over-simplify the autoencoding process. Besides, the size of block is designed to be large (height and width of blocks are of  $32 \sim 120$  pixels) and randomly changes in each iteration, and thus encouraging the cost-volume encoder to aggregate information from long-range context and to filter noises of cost values. The details of the mask generation algorithm are provided in supplementary.

Specifically, for each source pixel's cost map, we first generate the mask map  $\mathbf{B}_{\mathbf{x}}^3 \in \{0, 1\}^{\frac{H}{8} \times \frac{W}{8}}$  at  $\frac{1}{8}$  resolution, and then up-sample it  $2 \times$  for three times to obtain a pyramid of mask maps  $\mathbf{B}_{\mathbf{x}}^i \in \{0, 1\}^{\frac{H}{2^i} \times \frac{W}{2^i}}$ , where  $i \in \{0, 1, 2\}$ , which are used for the down-sampling encoding process and will be discussed later in Sec. 3.2.2.

Another key design is that, in pretraining, we **freeze** the ImageNet-pretrained Twins-SVT backbone to build the cost volume from the pair of input images. Freezing the image encoder ensures the reconstruction targets (*i.e.*, raw cost values) to maintain static and avoids training collapse.

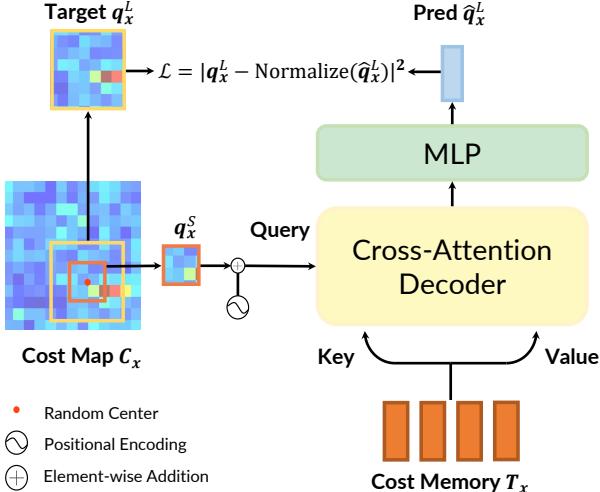
#### 3.2.2 Masked Cost-volume Tokenization

Given the above generated mask for each source pixel  $\mathbf{x}$ 's cost map, FlowFormer adopts a two-step cost-volume tokenization before the cost encoder. To prevent the masked costs from leaking into subsequent cost aggregation layers, the intermediate embeddings of the cost map need to be properly masked in the cost-volume tokenization process. We propose the masked cost-volume tokenization, which prevents mixing up masked and visible features. Firstly, FlowFormer patchifies the raw cost map  $\mathbf{A}_{\mathbf{x}} \in \mathbb{R}^{H \times W}$  of each source pixel  $\mathbf{x}$  (which is obtained by computing dot-product similarities between the source pixel  $\mathbf{x}$  and all target pixels) via 3 stacked stride-2 convolutions. We denote the feature maps after each of the 3 convolutions as  $\mathbf{F}_{\mathbf{x}}^i$ , which have spatial sizes of  $\frac{H}{2^i} \times \frac{W}{2^i}$  for  $i \in \{0, 1, 2\}$ . We propose to replace the vanilla convolutions used in the FlowFormer with masked convolutions [61], [67], [68]:

$$\mathbf{F}_{\mathbf{x}}^{i+1} = \text{Conv}_{\text{stride}2} (\text{ReLU}(\mathbf{F}_{\mathbf{x}}^i \odot \mathbf{B}_{\mathbf{x}}^i)), \quad (6)$$

where  $\odot$  indicates element-wise multiplication,  $i \in \{0, 1, 2\}$ , and  $\mathbf{F}_{\mathbf{x}}^0$  is the raw cost map  $\mathbf{A}_{\mathbf{x}}$ . The masked convolutions with the three binary mask maps remove all cost features in the masked regions in pretraining. Secondly, FlowFormer further projects the patchified cost-map features into the latent space via cross-attention. We thus remove the tokens in  $\mathbf{F}_{\mathbf{x}}^3$  indicated by the mask map  $\mathbf{B}_{\mathbf{x}}^3$  and then only project the remaining tokens into the latent space via the same cross-attention. During finetuning, the mask maps are removed to utilize all cost features, which converts the masked convolution to the vanilla convolution but the pretrained parameters in the convolution kernels and cross-attention layer are maintained.

The masked cost-volume tokenization completes two tasks. Firstly, it ensures the subsequent cost-volume encoder only processes visible features in pertaining. Secondly, the network structure is consistent with the standard tokenization of FlowFormer and can directly be used for finetuning



**Fig. 5: Pre-text reconstruction for cost memory decoding.** For each source pixel  $x$  and its corresponding cost map  $C_x$ , a small cost patch  $q_x^S$  is randomly cropped from the cost map to retrieve features from the cost memory  $T_x$ , aiming to reconstruct larger cost patch  $q_x^L$  centered at the same location.

so that the pretrained parameters have the same semantic meanings. After the masked cost-volume tokenization, the cost aggregation layers (*i.e.*, AGT layers) take visible features as input which also don't need to be modified in finetuning. The latent features interact with those of other source pixels in AGT layers and are transformed to the cost memory  $T_x$ . We explain how to decode the cost memory to estimate flows in following section.

### 3.2.3 Reconstruction Target for Cost Memory Decoding

With the masked cost-volume tokenization, the cost encoder encodes the unmasked cost volume into the cost memory. The next step is decoding and reconstructing the masked regions from the cost memory. We now formulate the pre-text reconstruction targets, which supervises the decoding process as well as aforementioned embedding and aggregation layers.

**Pre-text Reconstruction.** FlowFormer adopts recurrent flow prediction. In each iteration of the recurrent process, the flow of source pixel  $x$  is decoded from cost memory  $T_x$ , conditioned on current predicted flow, to update the flow prediction. Specifically, current predicted corresponding location in the target image  $p_x$  is computed as  $p_x = x + f(x)$ , where  $f(x)$  is current predicted flow. A local cost patch  $q_x$  is then cropped from the  $9 \times 9$  window centered at  $p_x$  on the raw cost map  $A_x$ . FlowFormer utilizes this local cost patch (with positional encoding) as the query feature to retrieve aggregated cost feature  $c_x$  via cross-attention operation (Eq. 4). Intuitively,  $c_x$  should contain long-range cost information for better optical flow estimation and it is conditioned on local cost patch  $q_x$ , which indicates the interested location on the cost map. We design a pre-text reconstruction task in line with these two characteristics to pretrain the cost-volume encoder as shown in Fig. 5: small cost-map patches are randomly cropped from the cost maps to retrieve cost features from the cost memory, targeting at reconstructing larger cost-map patches centered at the same locations.

Specifically, for each source pixel  $x$ , we randomly sample a location  $o_x$ , which is analogous to  $p_x$  in finetuning. Taking this location as center, we crop a small cost-map patch  $q_x^S = \text{Crop}_{9 \times 9}(C_x, o_x)$  of shape  $9 \times 9$ . Then we perform the decoding process shown in Eq. 4 to obtain the cost feature  $c_x$ , except that  $p_x$  and  $q_x$  are replaced by  $o_x$  and  $q_x^S$  in pretraining, respectively. To encourage the extracted cost feature  $c_x$  to carry long-range cost information conditioned on  $o_x$ , we take larger cost-map patch as supervision. Specifically, we crop another larger cost-map patch  $q_x^L = \text{Crop}_{15 \times 15}(C_x, o_x)$  of shape  $15 \times 15$  centered at the same location  $o_x$ . We choose a light-weight MLP as prediction head. The MLP takes as input  $c_x$  and its output  $\hat{q}_x^L = \text{MLP}(c_x)$  is supervised by normalized  $q_x^L$ . We take mean squared error (MSE) as loss function.

$$\mathcal{L} = \frac{1}{|\Omega|} \sum_{x \in \Omega} \left| q_x^L - \text{Normalize}(\hat{q}_x^L) \right|^2, \quad (7)$$

where  $\Omega$  is the set of source pixels.

**Discussion.** The key of pretraining is to maintain consistent with finetuning, in terms of both network architecture and prediction target. To this end, we keep the cross-attention decoding layer unchanged and construct inputs with the same semantic meaning (*e.g.*, replacing dynamically predicted  $p_x$  with randomly sampled  $o_x$ ); we supervise the extracted feature  $c_x$  with long-range cost values to encourage the cost-volume encoder to aggregate global information for better optical flow estimation. What's more, our scheme only takes an extra light-weight MLP as prediction head, which is unused in finetuning. Compared with previous methods that use a stack of self-attention layers, it is much more computationally efficient.

## 4 EXPERIMENTS

We denote MCVA pretrained FlowFormer as "FlowFormer+MCVA". We evaluate FlowFormer and FlowFormer+MCVA on the Sintel [69] and KITTI-2015 [70] benchmarks. Following previous works, FlowFormer is trained from scratch on FlyingChairs [27] and FlyingThings [71], and then respectively finetune it on the Sintel and KITTI-2015 benchmarks. FlowFormer+MCVA is pretrained with Masked Cost-volume Autoencoding on YouTube-VOS [72] dataset, and then supervised fine-tuning, following the FlowFormer training. FlowFormer sets new state-of-the-art on the Sintel benchmark and achieves the best generalization performance. FlowFormer+MCVA further obtains all-sided improvements over FlowFormer, ranking 1st on both benchmarks.

**Experimental Setup.** We adopt the commonly-used average end-point-error (AEPE) as the evaluation metric. It measures the average  $l_2$  distance between predictions and ground truth. For the KITTI-2015 dataset, we additionally use the F1-all (%) metric, which refers to the percentage of pixels whose flow error is larger than 3 pixels or over 5% of the length of ground truth flows. YouTube-VOS is a large-scale dataset containing video clips from YouTube website. The Sintel dataset is rendered from the same movie in two passes: the clean pass is rendered with easier smooth shading and specular reflections, while the final pass includes motion blur, camera depth-of-field blur and atmospheric

effects. The motions in the Sintel dataset are relatively large and complicated. The KITTI-2015 dataset constitutes of real-world driving scenarios with sparse ground truth.

**Implementation Details.** The image feature encoder of our final FlowFormer is chosen as the first two stages of ImageNet-pretrained Twins-SVT [47], 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 [19], we train FlowFormer on FlyingChairs [27] for 120k iterations with a batch size of 8, and on FlyingThings [71] for 120k iterations with a batch size of 6 (denoted as ‘C+T’). Then, we finetune FlowFormer on the data combined from FlyingThings, Sintel, KITTI-2015, and HD1K [73] (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  $2.5 \times 10^{-4}$  on FlyingChairs and  $1.25 \times 10^{-4}$  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 [18]. We use the tile technique for evaluating optical flow on KITTI because the size of images in KITTI is quite different from training image size. We use fixed Gaussian weights for tile. FlowFormer+MCVA uses the same architecture as FlowFormer. The image feature encoder and context feature encoder are frozen in pretraining. We pretrain FlowFormer+MCVA on YouTube-VOS for 50k iterations with a batch size of 24. The highest learning rate is set as  $5 \times 10^{-4}$ . During finetuning, we follow the same training procedure of FlowFormer.

**Gaussian tile.** Since positional encodings used in transformers are sensitive to image size and the size of an image pair for test ( $H_{test} \times W_{test}$ ) might be different from those of the training images, ( $H_{train} \times W_{train}$ ), we crop the test image pair according to the training size and estimate flows for patch pairs separately, and then tile the flows to obtain a complete flow map following a similar strategy proposed in Perceiver IO [18]. Specifically, we crop the image pair into four evenly-spaced tiles, i.e.,  $H_{train} \times W_{train}$  image tiles starting at  $(0, 0)$ ,  $(0, W_{test} - W_{train})$ ,  $(H_{test} - H_{train}, 0)$ , and  $(H_{test} - H_{train}, W_{test} - W_{train})$ , respectively. For each pixel that is covered by several tiles, we compute its output flow  $\mathbf{f}$  by blending the predicted flows  $\mathbf{f}_i$  with weighted averaging:

$$\mathbf{f} = \frac{\sum_i w_i \mathbf{f}_i}{\sum_i w_i}, \quad (8)$$

where  $w_i$  is the weight of the  $i$ -th tile for the pixel. We compute the  $H_{train} \times W_{train}$  weight map according to pixels' normalized distances  $d_i$  to the tile center:

$$\begin{aligned} d_i &= \|(\frac{u_i}{H_{train}} - 0.5, \frac{v_i}{W_{train}} - 0.5)\|_2, \\ w_i &= g(d_i; \mu = 0, \sigma = 0.05), \end{aligned} \quad (9)$$

where  $(u_i, v_i)$  denote a pixel  $i$ 's 2D coordinate. We use a Gaussian-like  $g$  as the weighting function to obtain

smoothly blended results. We use this weight map for all the tiles.

#### 4.1 Quantitative Comparison

As shown in Tab. 1, we evaluate FlowFormer and FlowFormer+MCVA on the Sintel and KITII-2015 benchmarks. Following previous methods, we evaluate the generalization performance of models on the training sets of Sintel and KITII-2015 (denoted as ‘C+T’). We also compare the dataset-specific accuracy of optical flow models after dataset-specific finetuning (denoted as ‘C+T+S+K+H’). Autoflow [78] is a synthetic dataset of complicated visual disturbance, while its training code is unreleased.

**Generalization Performance.** The ‘C+T’ setting in Tab. 1 reflects the generalization capacity of models. FlowFormer outperforms all other methods except FlowFormer+MCVA, which reveals the superiority of the proposed transformer architecture. FlowFormer+MCVA ranks 1st on both benchmarks among published methods. It achieves 0.90 and 2.30 on the clean and final pass of Sintel. Compared with FlowFormer, it further achieves 4.26% error reduction on Sintel clean pass. On the KITII-2015 training set, FlowFormer+MCVA achieves 3.93 F1-epc and 14.13 F1-all, improving FlowFormer by 0.16 and 0.59, respectively. These results show that our proposed MCVA promotes the generalization capacity of FlowFormer.

**Sintel Benchmark.** After training the models in the ‘C+T+S+K+H’ setting, we evaluate its performance on the Sintel online benchmark. FlowFormer sets new state-of-the-art performance on the Sintel benchmark, i.e. 1.16 on the clean pass and 2.09 on the final pass. FlowFormer+MCVA further improves the performance and achieves 1.07 and 1.94 on the two passes.

**KITII-2015 Benchmark.** We further finetune our models on the KITII-2015 training set after the Sintel stage and evaluate its performance on the KITII online benchmark. FlowFormer achieves 4.68, ranking 2nd on the KITII-2015 benchmark. S-Flow [36] obtains slightly smaller error than FlowFormer on KITII ( $-0.85\%$ ), 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 KITII 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. Even though, FlowFormer+MCVA achieves 4.52 F1-all, improving FlowFormer by 0.16 while also outperforming the previous best model S-Flow by 0.12.

To conclude, FlowFormer outperforms all published works on all tracks except KITII-2015 test due to the well-designed transformer architecture. FlowFormer+MCVA further unleashes the FlowFormer capacity by enhancing the cost-volume encoder with the MCVA pretraining.

#### 4.2 Qualitative Comparison

**GMA v.s. FlowFormer** We visualize flows that estimated by our FlowFormer and GMA of three examples in Fig. 6 to qualitatively show how FlowFormer outperforms GMA.

Training Data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)	
		Clean	Final	F1-epe	F1-all	Clean	Final	F1-all	
A	Perceiver IO [18]	1.81	2.42	4.98	-	-	-	-	-
	PWC-Net [16]	2.17	2.91	5.76	-	-	-	-	-
	RAFT [17]	1.95	2.57	4.23	-	-	-	-	-
C+T	HD3 [74]	3.84	8.77	13.17	24.0	-	-	-	-
	LiteFlowNet [31]	2.48	4.04	10.39	28.5	-	-	-	-
	PWC-Net [16]	2.55	3.93	10.35	33.7	-	-	-	-
	LiteFlowNet2 [32]	2.24	3.78	8.97	25.9	-	-	-	-
	S-Flow [36]	1.30	2.59	4.60	15.9	-	-	-	-
	RAFT [17]	1.43	2.71	5.04	17.4	-	-	-	-
	FM-RAFT [35]	1.29	2.95	6.80	19.3	-	-	-	-
	GMA [19]	1.30	2.74	4.69	17.1	-	-	-	-
	GMFlow [43]	1.08	2.48	-	-	-	-	-	-
	GMFlowNet [75]	1.14	2.71	4.24	15.4	-	-	-	-
	CRAFT [44]	1.27	2.79	4.88	17.5	-	-	-	-
	SKFlow [76]	1.22	2.46	4.47	15.5	-	-	-	-
	FlowFormer (Ours)	0.94 <sup>†</sup>	2.33 <sup>†</sup>	4.09 <sup>†</sup>	14.72 <sup>†</sup>	-	-	-	-
	FlowFormer+MCVA (Ours)	0.90 <sup>†</sup>	2.30 <sup>†</sup>	3.93 <sup>†</sup>	14.13 <sup>†</sup>	-	-	-	-
C+T+S+K+H	LiteFlowNet2 [32]	(1.30)	(1.62)	(1.47)	(4.8)	3.48	4.69	7.74	
	PWC-Net+ [30]	(1.71)	(2.34)	(1.50)	(5.3)	3.45	4.60	7.72	
	VCN [33]	(1.66)	(2.24)	(1.16)	(4.1)	2.81	4.40	6.30	
	MaskFlowNet [77]	-	-	-	-	2.52	4.17	6.10	
	S-Flow [36]	(0.69)	(1.10)	(0.69)	(1.60)	1.50	2.67	4.64	
	RAFT [17]	(0.76)	(1.22)	(0.63)	(1.5)	1.94	3.18	5.10	
	RAFT* [17]	(0.77)	(1.27)	-	-	1.61	2.86	5.10	
	FM-RAFT [35]	(0.79)	(1.70)	(0.75)	(2.1)	1.72	3.60	6.17	
	GMA [19]	-	-	-	-	1.40	2.88	5.15	
	GMA* [19]	(0.62)	(1.06)	(0.57)	(1.2)	1.39	2.47	5.15	
	GMFlow [43]	-	-	-	-	1.74	2.90	9.32	
	GMFlowNet [75]	(0.59)	(0.91)	(0.64)	(1.51)	1.39	2.65	4.79	
	CRAFT [44]	(0.60)	(1.06)	(0.57)	(1.20)	1.45	2.42	4.79	
	SKFlow* [76]	(0.52)	(0.78)	(0.51)	(0.94)	1.28	2.23	4.84	
	FlowFormer (Ours)	(0.48)	(0.74)	(0.53)	(1.11)	1.16	2.09	4.68 <sup>†</sup>	
	FlowFormer+MCVA (Ours)	(0.40)	(0.60)	(0.57)	(1.16)	1.07	1.94	4.52 <sup>†</sup>	

TABLE 1: Experiments on Sintel [69] and KITTI [70] datasets. ‘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. \* denotes that the methods use the warm-start strategy [17], which relies on previous image frames in a video, while other methods use two frames only. † denotes the result is obtained via the tile technique. Our FlowFormer+MCVA achieves the best generalization performance (C+T) and ranks 1st on both the Sintel and the KITTI-15 benchmarks (C+T+S+K+H).

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 estimated by FlowFormer on boundaries of the bamboo and the human body are more precise and clear.

**FlowFormer v.s. FlowFormer+MCVA** MCVA pretraining encourages the cost-volume encoder to aggregate cost information in a long range. We visualize flow predictions by our FlowFormer and FlowFormer+MCVA on KITTI test sets in Fig. 6 to qualitatively show how MCVA pretraining improves FlowFormer. The red arrows highlight that FlowFormer+MCVA preserves clearer details than FlowFormer and shows greater global aggregation capacity indicated by blue boxes.

### 4.3 Ablation Study on FlowFormer Architecture

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 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 ( $T_x$  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 further 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 [27] with 100k iterations and FlyingThings [71] with 60k iterations, and then evaluated on the training set of Sintel [69] and KITTI-2015 [70].

**MCR → LCT+MCD.** The number of latent tokens  $K$  and

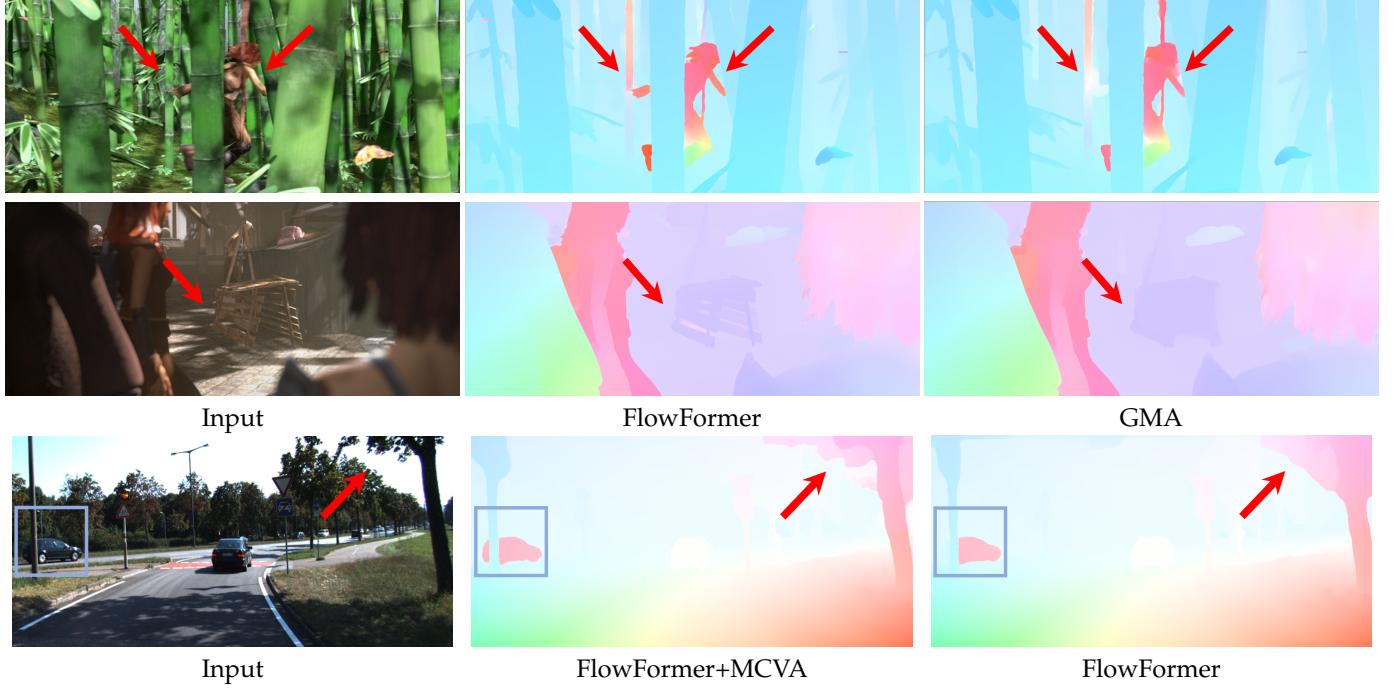


Fig. 6: **Qualitative comparison.** As pointed by red arrows, FlowFormer greatly reduces the flow leakage around object boundaries and clearer details. FlowFormer+MCVA further improves the flow details and maintains better global consistency (indicated by the blue box).

Experiment	Method	Sintel (train)		KITTI-15 (train)		Params.
		Clean	Final	F1-epc	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 on the FlowFormer architecture.** 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.

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.

**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.

**Transformer v.s. MLP&Conv.** As shown in Table 4, we con-

Method	Sintel (train)		KITTI-15 (train)		parameters
	Clean	Final	F1-epc	F1-all	
GMA [19]	1.30 (+30%)	2.74 (+12%)	4.69 (+15%)	17.1 (+16%)	5.9M
FlowFormer (small)	1.20 (+20%)	2.64 (+8%)	4.57 (+12%)	16.62 (+13%)	6.2M
GMA-L [19]	1.33 (+33%)	2.56 (+4%)	4.40 (+8%)	15.93 (+8%)	17.0M
GMA-Twins [19]	1.15 (+15%)	2.73 (+11%)	4.98 (+22%)	16.82 (+14%)	14.2M
FlowFormer	<b>1.00</b>	<b>2.45</b>	<b>4.09</b>	<b>14.72</b>	18.2M

TABLE 3: **FlowFormer v.s. GMA architectures.** 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.

Intra.	Inter.	Sintel (train)		KITTI (train)		Params.
		Clean	Final	F1-epc	F1-all	
Trans	Trans	<b>1.20</b>	2.85	<b>4.57</b>	<b>15.46</b>	15.2M
MLP	Trans	1.20	<b>2.67</b>	5.01	16.81	15.2M
Trans	Conv	1.23	2.72	4.73	15.87	15.1M
MLP	Conv	1.22	2.71	4.88	17.23	15.1M

TABLE 4: **Transformer v.s. MLP&Conv in AGT.** For intra-cost-map aggregation layer (Intra.), we replace transformer (Trans) with MLP-Mixer [79] block (MLP). For inter-cost-map aggregation layer (Inter.), we replace transformer with ConvNeXt [80] block (Conv).

Sintel (train)	clean		final	
	w/ tile	w/o tile	with tile	w/o tile
FlowFormer	<b>0.94</b>	1.01	<b>2.33</b>	2.40

TABLE 5: **FlowFormer with v.s. without tile.** We tile test images with the training image size. Tiling slightly improves performance.

Method	Sintel		KITTI-15	
	Clean↓	Final↓	F1-epc↓	F1-all↓
FlowFormer	<b>1.10</b>	<b>2.57</b>	<b>4.45</b>	<b>15.15</b>
w/o AGT PE	1.18	2.63	4.78	16.19
w/o token PE	1.19	2.59	4.63	16.23

TABLE 6: **FlowFormer with v.s. without positional encoding (PE).** Removing PE at either the AGT stage or the tokenization stage hampers FlowFormer.

duct ablation experiments on the alternative-group transformer (AGT) layer. For intra-cost-map aggregation layer, since the number and dimension of latent cost tokens are fixed, we test on replacing our design with MLP-Mixer [79] (2nd row), which is a state-of-the-art MLP-based architecture. We also substitute ConvNeXt [80] for transformer in inter-cost-map aggregation (3rd row). Furthermore, we replace both transformers with MLP and ConvNext (4th row). Replacing transformer layers leads to slightly better performance on Sintel final pass, while brings a clear drop on KITTI. Therefore, we adopt the proposed full transformer architecture as our final model.

**Optical flow tiling.** Transformers are sensitive to image size due to the use of positional encodings. We use the Gaussian tile technique to avoid image size misalignment between training and test stages. Tab. 5 shows that the tiling technique slightly improves performance.

**FlowFormer with v.s. without positional encoding (PE).** Transformers encode tokens with PE to obtain position information. We add PE to cost tokens in two stages: the

MLP Depth	Sintel (train)		KITTI-15 (train)	
	Clean	Final	F1-epc	F1-all
1	0.92	<b>2.30</b>	4.10	14.66
3	0.90	<b>2.30</b>	<b>3.93</b>	<b>14.13</b>
5	0.94	2.35	4.08	14.64

TABLE 7: **Reconstruction head depth.** We use an MLP as the reconstruction head during MCVAE pretraining. The three layer MLP yields the best result.

Strategy	Ratio	Sintel (train)		KITTI-15 (train)	
		Clean	Final	F1-epc	F1-all
Block-sharing	20%	0.97	2.41	<b>3.91</b>	14.27
Block-sharing	50%	<b>0.90</b>	<b>2.30</b>	3.93	14.13
Block-sharing	80%	0.94	2.35	4.00	<b>14.07</b>
Random	20%	0.98	2.39	4.12	14.89
Random	50%	0.93	2.35	4.03	14.30
Random	80%	0.92	2.38	4.10	14.51

TABLE 8: **Cost Volume Masking Strategy.** Masked cost volume autoencoding with different masking strategies and ratios. Block-sharing masking brings greater performance gain than random masking. Masking 50% cost values yields the best overall results.

tokenization stage and the AGT cost encoding stage. Tab. 6 reveals PE is essential in both stages. Removing either PE leads to a performance drop.

**FlowFormer vs. 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  $\times 1$ . 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 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) [17], GMA-L enlarges GMA by doubling feature channels, which has 17M parameters, comparable to FlowFormer. However,

Location	Query Feature	Sintel (train)		KITTI-15 (train)	
		Clean	Final	F1-epc	F1-all
Fixed	PE	0.99	2.40	4.35	15.33
Random	PE	0.95	2.42	3.99	14.47
Random	PE + Cropped patch	0.90	2.30	3.93	14.13

TABLE 9: **Pre-text reconstruction design.** Our pre-text reconstruction task leads to better performance over conventional MAE task (first row) and its improved version with random reconstruction locations (second row).

Size of $q_x^L$	Sintel (train)		KITTI-15 (train)	
	Clean	Final	F1-epc	F1-all
15 × 15	0.90	2.30	3.93	14.13
25 × 25	0.92	2.28	3.87	14.54

TABLE 10: **Size of  $q_x^L$ .**  $q_x^L$  is the cost-map patch to be reconstructed in MVCA. Further increasing the size of  $q_x^L$  does not bring clear gain.

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 F1-epc. In conclusion, the performance improvement of FlowFormer is not derived from more parameters but the novel design of the architecture.

#### 4.4 Ablation Study on MCVA Pretraining

We conduct a set of ablation studies to show the effectiveness of designs in the Masked Cost-volume Autoencoding (MCVA). All models in the experiments are first pretrained and then finetuned on ‘C+T’. We report the test results on Sintel and KITTI training sets.

**The Depth of the Reconstruction Head.** We use a light weight MLP as the reconstruction head. As shown in Tab. 7, the best performance is achieved when the depth is 3, so we use a 3-layer MLP as the reconstruction head when pretraining FlowFormer with MCVA. Using a shallow single layer as the reconstruction head makes the reconstruction too challenging and leads to inferior performance. A 5-layer MLP also shows to worse performance because the effective knowledge may be shifted from the AGT to the reconstruction head while the reconstruction head will be dropped in finetuning.

**Masking Strategy.** Masking strategy is one important design of our MCVA. As shown in Tab. 8, pretraining FlowFormer with random masking already improves the performance on three of the four metrics. But the proposed block-sharing masking strategy brings even larger gain, which demonstrates the effectiveness of this design. Besides, we observe higher pretraining loss with block-sharing masking than that with random masking, validating that the block-sharing masking makes the pretraining task harder.

**Masking Ratio.** Masking ratio influences the difficulty of the pre-text reconstruction task. We empirically find that the mask ratio of 50% yields the best overall performance (Tab. 8).

Freeze		Sintel (train)		KITTI-15 (train)	
IE	CE	Clean	Final	F1-epc	F1-all
✗	✗	-	-	-	-
✗	✓	-	-	-	-
✓	✗	0.92	2.34	3.90	14.23
✓	✓	0.90	2.30	3.93	14.13

TABLE 11: **Freezing the image encoder (IE) and the context encoder (CE) in pretraining.** Freezing the ImageNet pretrained image encoder ensures the reconstruction targets (*i.e.*, raw cost values) maintain static, otherwise the model diverges. Freezing the context encoder leads to better overall performance.

Methods	Sintel (train)		KITTI-15 (train)	
	Clean	Final	F1-epc	F1-all
Unsupervised Baseline	0.99	2.54	4.38	15.22
MCVA (ours)	0.90	2.30	3.93	14.13

TABLE 12: **Comparisons with unsupervised methods.** We use the conventional unsupervised algorithm [81], [82], [83] (using photometric loss and smooth loss) to pretrain FlowFormer for comparison. Our MCVA outperforms the unsupervised counterpart.

**Pre-text Reconstruction Design.** The conventional MAE methods aim to reconstruct input data at fixed locations and use the positional encodings as query features to absorb information for reconstruction (the first row of Tab. 9). To keep consistent with the dynamic positional query of FlowFormer architecture, we propose to reconstruct contents at random locations (the second row of Tab. 9) and additionally use local patches as query features (the third row of Tab. 9). The results validate the necessity of ensuring semantic consistency between pretraining and finetuning.

**Size of  $q_x^L$ .** During MCVA pertaining, FlowFormer reconstructs a larger cost-map patch  $q_x^L$  with features retrieved from the cost memory. Increasing the size of the  $q_x^L$  will increase the reconstruction difficulty. We show that  $15 \times 15$  is a good choice for MCVA.

**Freezing Image and Context Encoders.** The FlowFormer architecture has an image encoder to encode visual appearance features for constructing the cost volume, and a context encoder to encode context features for flow prediction. As shown in Tab. 11, freezing the image encoder is necessary, otherwise the model diverges. We hypothesize that the frozen image encoder ensures the reconstruction targets (*i.e.*, raw cost values) to keep static. Freezing the context encoder leads to better overall performance.

**Comparisons with Unsupervised Methods.** We also use conventional unsupervised methods to pretrain FlowFormer with photometric loss and smooth loss following [81], [82], [83] and then finetune it in the ‘C+T’ setting as MVCA-FlowFormer. As shown in Tab. 12, our MCVA outperforms the unsupervised counterpart for pretraining FlowFormer.

**FlowFormer with v.s. without MCVA on FlyingChairs.** We show the training and validating loss of the training process on FlyingChairs [27] in Fig. 7. FlowFormer+MCVA presents faster convergence during training and better validation loss at the end, which reveals that FlowFormer learns effective feature relationships during MCVA pretraining and benefits

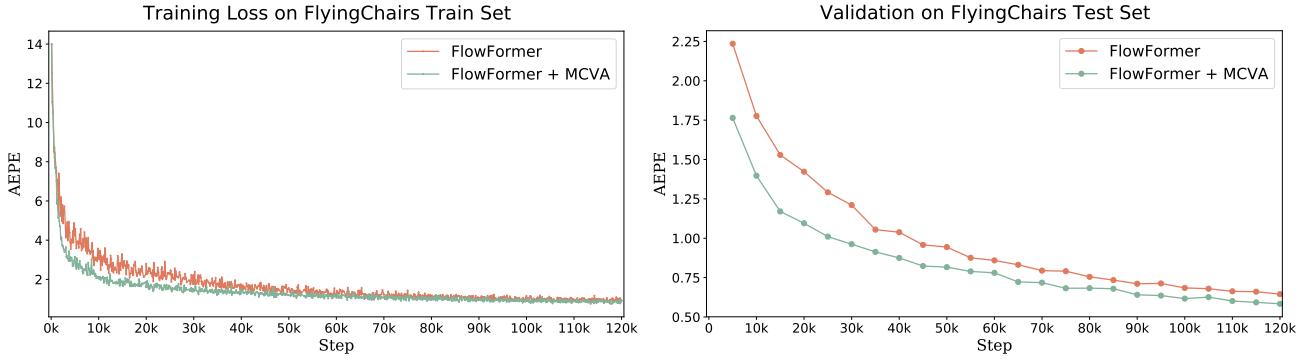


Fig. 7: AEPE vs. training-step curves of FlowFormer and FlowFormer+MCVA. FlowFormer+MCVA converges faster and achieves lower validation error.

the supervised finetuning.

## 5 CONCLUSION

In this paper, we propose an optical Flow Transformer (FlowFormer) and Masked Cost Volume Autoencoding (MCVA) to enhance the cost-volume encoder of FlowFormer by pertaining. MCVA-FlowFormer is the first work that introduces the transformers with MAE pretraining paradigm to optical flow and sheds light on improving optical flow estimation with unlabeled data. We show that the naive adaptation of MAE scheme to cost volume does not work due to the redundant nature of cost volumes and the incurred pretraining-finetuning discrepancy. We tackle these issues with a specially designed block-sharing masking strategy and the novel pre-text reconstruction task. These designs ensure semantic integrity between pretraining and finetuning and encourage the cost-volume to aggregate information in a long range. Experiments demonstrate clear generalization and dataset-specific performance improvements. **Limitations.** FlowFormer is a transformer-based network architecture, which is slightly sensitive to image size due to the positional encoding. Therefore, FlowFormer needs the tile technique for images that have a quite different size compared to training images.

## REFERENCES

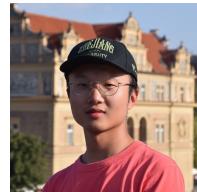
- [1] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, "Optical flow guided feature: A fast and robust motion representation for video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1390–1399.
- [2] A. Piergiovanni and M. S. Ryoo, "Representation flow for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9945–9953.
- [3] Y. Zhao, K. L. Man, J. Smith, K. Siddique, and S.-U. Guan, "Improved two-stream model for human action recognition," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–9, 2020.
- [4] D. Kim, S. Woo, J.-Y. Lee, and I. S. Kweon, "Deep video inpainting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5792–5801.
- [5] R. Xu, X. Li, B. Zhou, and C. C. Loy, "Deep flow-guided video inpainting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3723–3732.
- [6] C. Gao, A. Saraf, J.-B. Huang, and J. Kopf, "Flow-edge guided video completion," in *European Conference on Computer Vision*. Springer, 2020, pp. 713–729.
- [7] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 624–632.
- [8] L. Wang, Y. Guo, L. Liu, Z. Lin, X. Deng, and W. An, "Deep video super-resolution using hr optical flow estimation," *IEEE Transactions on Image Processing*, vol. 29, pp. 4323–4336, 2020.
- [9] K. C. Chan, X. Wang, K. Yu, C. Dong, and C. C. Loy, "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*, 2021, pp. 4947–4956.
- [10] M. S. Sajjadi, R. Vemulapalli, and M. Brown, "Frame-recurrent video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6626–6634.
- [11] X. Xu, L. Siyao, W. Sun, Q. Yin, and M.-H. Yang, "Quadratic video interpolation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [12] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3703–3712.
- [13] X. Liu, H. Liu, and Y. Lin, "Video frame interpolation via optical flow estimation with image inpainting," *International Journal of Intelligent Systems*, vol. 35, no. 12, pp. 2087–2102, 2020.
- [14] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5437–5446.
- [15] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, "Rife: Real-time intermediate flow estimation for video frame interpolation," *arXiv preprint arXiv:2011.06294*, 2020.
- [16] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.
- [17] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*. Springer, 2020, pp. 402–419.
- [18] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.
- [19] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," *arXiv preprint arXiv:2104.02409*, 2021.
- [20] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Springer, 2020, pp. 213–229.
- [21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [22] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

- [23] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.
- [24] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *1993 (4th) International Conference on Computer Vision*. IEEE, 1993, pp. 231–236.
- [25] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International journal of computer vision*, vol. 61, no. 3, pp. 211–231, 2005.
- [26] D. Sun, S. Roth, and M. J. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," *International Journal of Computer Vision*, vol. 106, no. 2, pp. 115–137, 2014.
- [27] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.
- [28] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.
- [29] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.
- [30] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Models matter, so does training: An empirical study of cnns for optical flow estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 6, pp. 1408–1423, 2019.
- [31] T.-W. Hui, X. Tang, and C. C. Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8981–8989.
- [32] ———, "A lightweight optical flow cnn—revisiting data fidelity and regularization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 8, pp. 2555–2569, 2020.
- [33] G. Yang and D. Ramanan, "Volumetric correspondence networks for optical flow," *Advances in neural information processing systems*, vol. 32, pp. 794–805, 2019.
- [34] H. Xu, J. Yang, J. Cai, J. Zhang, and X. Tong, "High-resolution optical flow from 1d attention and correlation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 498–10 507.
- [35] S. Jiang, Y. Lu, H. Li, and R. Hartley, "Learning optical flow from a few matches," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 592–16 600.
- [36] F. Zhang, O. J. Woodford, V. A. Prisacariu, and P. H. Torr, "Separable flow: Learning motion cost volumes for optical flow estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 807–10 817.
- [37] M. Hofinger, S. R. Bulò, L. Porzi, A. Knapsch, T. Pock, and P. Kontschieder, "Improving optical flow on a pyramid level," in *European Conference on Computer Vision*. Springer, 2020, pp. 770–786.
- [38] Y. Zheng, M. Zhang, and F. Lu, "Optical flow in the dark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6749–6757.
- [39] W. Yan, A. Sharma, and R. T. Tan, "Optical flow in dense foggy scenes using semi-supervised learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 259–13 268.
- [40] Z. Huang, X. Pan, R. Xu, Y. Xu, G. Zhang, H. Li *et al.*, "Life: Lighting invariant flow estimation," *arXiv preprint arXiv:2104.03097*, 2021.
- [41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [43] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "Gmflow: Learning optical flow via global matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8121–8130.
- [44] X. Sui, S. Li, X. Geng, Y. Wu, X. Xu, Y. Liu, R. Goh, and H. Zhu, "Craft: Cross-attentional flow transformer for robust optical flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 602–17 611.
- [45] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.
- [46] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.
- [47] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting spatial attention design in vision transformers," *arXiv preprint arXiv:2104.13840*, 2021.
- [48] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.
- [49] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.
- [50] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 299–12 310.
- [51] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1833–1844.
- [52] Y. Zeng, J. Fu, and H. Chao, "Learning joint spatial-temporal transformations for video inpainting," in *European Conference on Computer Vision*. Springer, 2020, pp. 528–543.
- [53] R. Liu, H. Deng, Y. Huang, X. Shi, L. Lu, W. Sun, X. Wang, J. Dai, and H. Li, "Fuseformer: Fusing fine-grained information in transformers for video inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 040–14 049.
- [54] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8922–8931.
- [55] Z. Huang, H. Zhou, Y. Li, B. Yang, Y. Xu, X. Zhou, H. Bao, G. Zhang, and H. Li, "Vs-net: Voting with segmentation for visual localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6101–6111.
- [56] S. Cho, S. Hong, S. Jeon, Y. Lee, K. Sohn, and S. Kim, "Cats: Cost aggregation transformers for visual correspondence," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [57] W. Jiang, E. Trulls, J. Hosang, A. Tagliasacchi, and K. M. Yi, "Cotr: Correspondence transformer for matching across images," *arXiv preprint arXiv:2103.14167*, 2021.
- [58] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.
- [59] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *International conference on machine learning*. PMLR, 2020, pp. 1691–1703.
- [60] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, "Simmim: A simple framework for masked image modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9653–9663.
- [61] P. Gao, T. Ma, H. Li, J. Dai, and Y. Qiao, "Convmae: Masked convolution meets masked autoencoders," *arXiv preprint arXiv:2205.03892*, 2022.
- [62] Z. Tong, Y. Song, J. Wang, and L. Wang, "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training," *arXiv preprint arXiv:2203.12602*, 2022.
- [63] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Pointbert: Pre-training 3d point cloud transformers with masked point modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 313–19 322.
- [64] Y. Pang, W. Wang, F. E. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," *arXiv preprint arXiv:2203.06604*, 2022.
- [65] G. Hess, J. Jaxing, E. Svensson, D. Hagerman, C. Petersson, and L. Svensson, "Masked autoencoder for self-supervised pre-training on lidar point clouds," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 350–359.

- [66] R. Zhang, Z. Guo, P. Gao, R. Fang, B. Zhao, D. Wang, Y. Qiao, and H. Li, "Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training," *arXiv preprint arXiv:2205.14401*, 2022.
- [67] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.
- [68] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "Sbnnet: Sparse blocks network for fast inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8711–8720.
- [69] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European conference on computer vision*. Springer, 2012, pp. 611–625.
- [70] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [71] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4040–4048.
- [72] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang, "Youtube-vos: A large-scale video object segmentation benchmark," *arXiv preprint arXiv:1809.03327*, 2018.
- [73] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrlulis, A. Brock, B. Gussefeld, M. Rahimimoghadam, S. Hofmann, C. Brenner *et al.*, "The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 19–28.
- [74] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6044–6053.
- [75] S. Zhao, L. Zhao, Z. Zhang, E. Zhou, and D. Metaxas, "Global matching with overlapping attention for optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17592–17601.
- [76] S. Sun, Y. Chen, Y. Zhu, G. Guo, and G. Li, "Skflow: Learning optical flow with super kernels," *arXiv preprint arXiv:2205.14623*, 2022.
- [77] S. Zhao, Y. Sheng, Y. Dong, E. I. Chang, Y. Xu *et al.*, "Maskflownet: Asymmetric feature matching with learnable occlusion mask," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6278–6287.
- [78] D. Sun, D. Vlasic, C. Herrmann, V. Jampani, M. Krainin, H. Chang, R. Zabih, W. T. Freeman, and C. Liu, "Autoflow: Learning a better training set for optical flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10093–10102.
- [79] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," 2021.
- [80] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022.
- [81] A. Stone, D. Maurer, A. Ayvaci, A. Angelova, and R. Jonschkowski, "Smurf: Self-teaching multi-frame unsupervised raft with full-image warping," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3887–3896.
- [82] S. Liu, K. Luo, N. Ye, C. Wang, J. Wang, and B. Zeng, "Oiflow: Occlusion-inpainting optical flow estimation by unsupervised learning," *IEEE Transactions on Image Processing*, vol. 30, pp. 6420–6433, 2021.
- [83] K. Luo, C. Wang, S. Liu, H. Fan, J. Wang, and J. Sun, "Upflow: Upsampling pyramid for unsupervised optical flow learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 1045–1054.

## ACKNOWLEDGMENTS

This work is supported in part by Centre for Perceptual and Interactive Intelligence Limited, in part by the General Research Fund through the Research Grants Council of Hong Kong No. 14204021.



**Zhaoyang Huang** Zhaoyang Huang is currently a Ph.D. candidate in electronic engineering from The Chinese University of Hong Kong, advised by Hongsheng Li. He received his bachelor's and master's degree in computer science from Zhejiang University. His research focuses on image correspondence learning, video editing, and generative neural radiance fields.



**Xiaoyu Shi** Xiaoyu Shi is currently a Ph.D. candidate in electronic engineering from The Chinese University of Hong Kong, advised by Hongsheng Li. He received his bachelor's degrees in computer science from Zhejiang University and Simon Fraser University. His research focuses on optical flow estimation.



**Chao Zhang** Chao Zhang received the BS degree in Electrical Engineering from the Tianjin University, Tianjin, China, in 2009. And the PhD degree in computer science from The University of New South Wales, NSW, Australia in 2014. He is currently a staff engineer at the SAIT China Lab, Samsung Research China, Beijing. His research interests include computer vision, camera ISP, and machine learning. Chao Zhang received the BS degree in Electrical Engineering from the Tianjin University, Tianjin, China, in 2009. And the PhD degree in computer science from The University of New South Wales, NSW, Australia in 2014. He is currently a staff engineer at the SAIT China Lab, Samsung Research China, Beijing. His research interests include computer vision, camera ISP, and machine learning.



**Qiang Wang** Qiang Wang received the BS, MS and Ph.D degrees in computer science from Tsinghua University, Beijing, China, in 1998, 2001 and 2004, respectively. He is currently a principal researcher at Samsung Research China - Beijing, Samsung Electronics. His research interests include computer vision, machine learning, and computer graphics.



**Yijin Li** Yijin Li is currently a Ph.D. candidate in computer science from Zhejiang University, advised by Guofeng Zhang. He received his bachelor's degree in software engineering from Sun Yat-sen University. His research focuses on event-based vision, 3D reconstruction, and augmented reality. Yijin Li is currently a Ph.D. candidate in computer science from Zhejiang University, advised by Guofeng Zhang. He received his bachelor's degree in software engineering from Sun Yat-sen University. His research focuses on event-based vision, 3D reconstruction, and augmented reality.



**Hongwei Qin** Hongwei Qin leads a team that focuses on AI ISP and AI Video Codec (including software and silicon) for next generation AI-Camera. Before that, he got his PhD and Bachelor of Engineering from Tsinghua University in Jun. 2017 and Jun. 2012 respectively. He works in SenseTime from 2015 till now.

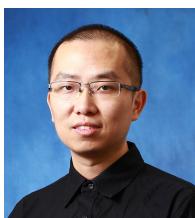


**Jifeng Dai** Jifeng Dai is an Associate Professor at the Department of Electronic Engineering of Tsinghua University. His current research focus is on deep learning for high-level vision. Prior to that, He was an Executive Research Director at SenseTime Research, headed by Professor Xiaogang Wang, between 2019 and 2022. He was a Principle Research Manager in Visual Computing Group at Microsoft Research Asia (MSRA) between 2014 and 2019, headed by Dr. Jian Sun. He got his Ph.D. degree from the

Department of Automation, Tsinghua University in 2014, under the supervision of Professor Jie Zhou.



**Xiaogang Wang** Xiaogang Wang received his Bachelor degree in Electrical Engineering and Information Science from the Special Class of Gifted Young at the University of Science and Technology of China, MPhil. degree in Information Engineering from the Chinese University of Hong Kong, and Ph.D. degree in Computer Science from Massachusetts Institute of Technology. He is an assistant professor in the Department of Electronic Engineering at the Chinese University of Hong Kong since August 2009. He was the Area Chair of IEEE International Conference on Computer Vision (ICCV) 2011. He received the Outstanding Young Researcher in Automatic Human Behaviour Analysis award in 2011, and Hong Kong Early Career Award in 2012.



**Hongsheng Li** Hongsheng Li (Member, IEEE) received the BS degree in automation from the East China University of Science and Technology, Shanghai, China, in 2006, and the MS and PhD degrees in computer science from Lehigh University, Bethlehem, Pennsylvania, in 2010 and 2012, respectively. He is currently an associate professor at the Department of Electronic Engineering, The Chinese University of Hong Kong. His research interests include computer vision, medical image analysis, and machine learning.