APPENDIX

A. More Details on Optical Flow Extraction

Before generating the trajectory, we need to calculate dense optical flows frame-by-frame to provide the fine-grained motion of each pixel. We follow the previous work to remove camera motion and pre-extract the optical flows of each video in the dataset offline. Details are as below.

Removing camera motion. To alleviate the influence of camera motion, we follow previous work [67] to warp the optical flow, which is also be applied in two stream video action recognition method [71]. Specifically, by matching SURF [6] interest point in two frames to estimate the camera motion using RANSC [31] algorithm. Then the camera motion is removed by rectifying the frame, and the optical flow is re-calculated using the warped frame. This also relieves the model from being disturbed by the background and makes it pay more attention to the moving objects in the foreground.

Pre-extracting optical flow. We use the *denseflow*[1] tool box to pre-extract warp optical flow before training following [67,71]. We set the upper bound of flow value to 20. The stride of flow is set to $s_{flow} = 1$ if perform motion target interpolation, otherwise is equal to the sampling stride of the input rgb frames $s_{flow} = s_{rgb}$. The whole video dataset is split into chunks and processed on 4 nodes, each node is equipped with 2 Intel Xeon CPUs (32 cores per CPU) for video codec and 8 TITAN X GPUs for optical flow calculation speed-up. The extraction process spends about 1 day on the Kinetics400 dataset.

B. More Details on Motion Trajectory Generation

As mentioned in Section 3, we use a motion trajectory to represent long-term and fine-grained motion, which consists of position features $\mathbf{z}^p$ and shape features $\mathbf{z}^s$. To extract these two features, we first need to calculate a continuous trajectory that tracks the position transition of a specific grid point. We then consider the surrounding area of this trajectory as a $\tau \times W \times W$ volume, as shown in Fig. 5. The generation process of this trajectory is described in detailed as following.

Tracking the trajectory. With the pre-extracted warp optical flow ω, we are able to track the position transition of each densely sampled grid point to produce a continuous trajectory, *i.e.*,

$$\mathbf{p}_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * \omega_t)_{(x_t, y_t)}, \quad (6)$$

[1] https://github.com/yjxiong/dense_flow

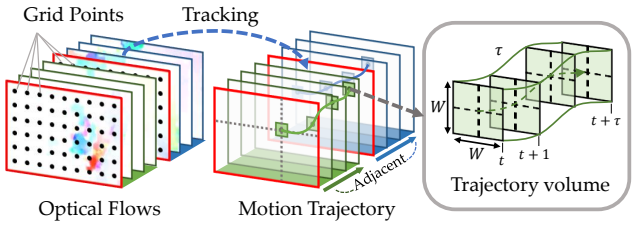

Figure 5. **The detailed generation process of motion trajectory.**

where ω_t is the optical flow at step t. M is the median filter kernel shapes 3×3 and $*$ denotes the convolution operator.

Pre-extracting the trajectory. To reduce the io overhead of loading and decoding flow images during pre-training, we pre-process the dataset to generate dense trajectories of each video and pack them in the compressed pickle files. In pre-training, we use a special dataloader to sample K trajectories with length L starting from the first frame of each input 3D patch. During pre-training, we then crop and resize the trajectories in the spatial dimension to maintain the corresponding area with the input RGB frames.

Masking inaccurate trajectory. We notice that in most of the video samples, the objects move out of the range of the camera FOV. This circumstance often occurs on the objects that are at the edge of the video. The tracked trajectories, in this case, could be inaccurate, as visualized in Fig. 6. We use a loss mask to remove these trajectories from the loss calculation:

$$\mathcal{L}^M = \frac{1}{N_p} \sum_{i=1}^{N_p} m_i \cdot \mathcal{L}_i, \quad m_i = \begin{cases} 1, \mathbf{p}_t \in \mathbf{P} \\ 0, otherwise \end{cases} \quad (7)$$

where N_p is the total number of unmasked patches, $\mathbf{P} = \{(x_t, y_t) \mid 0 < x_t < W, 0 < y_t < H\}$ represents the points in the video clip.

Figure 6. **Illustration of the inaccurate trajectories** (marked as blue), which are caused by objects moving out of the camera FOV.

C. More Details on Baseline Reconstruction Targets Extraction

In Section 4.2.2, we introduce different types of baseline reconstruction targets (*e.g.*, HOG, HOF and MBH) and investigate the effect of reconstructing these targets in the mask-and-predict task. In this section, we present the details of extracting these baseline reconstruction targets.

HOG. In our experiment, we utilize the successful application of the HOG feature [21] in masked video modeling [75] to represent appearance information. HOG is the statistic histogram of image gradients in RGB channels. To speed up the calculation, we re-implement the HOG extractor by convolution operator with Sobel kernel and the scatter method in the PyTorch [50] Tensor API, and thus the calculation process can be conducted on the GPU. With the exception of the L2 normalization, which does not work in our practice, we precisely follow the recipe in Wei's work [75] to calculate a feature map of the entire image and then fetch the 9-bins HOG feature in each RGB channel.

HOF and MBH. These two features are spatial-temporal local features that have been widely used in the action recognition task since the 2010s. Similar to the HOG feature, these two features are the histogram of optical flow [45], the only difference is that MBH is the second-order histogram of the optical flow [22]. We implement them based on the HOG extractor that deals with optical flow.

D. More Details on Pre-training and Fine-tuning Settings

We pre-train MME on the two large-scale video datasets (cf. Tab. 8) and then transfer to four action recognition benchmark datasets (cf. Tab. 9). We linearly scale the base learning rate w.r.t. the overall batch size, $lr = base\_lr \times \frac{batchsize}{256}$. More details are presented as follows:

Something-Something V2.[2] The default settings for pre-training and fine-tuning on Something-Something V2 (SSv2) are shown in Tab. 8 and Tab. 9. We take the same recipe as in [64].

Kinetics-400.[3] The default settings for pre-training and fine-tuning on Kinetics-400 (K400) are shown in Tab. 8 and Tab. 9. We take the same recipe as in [64].

UCF101.[4] We pre-train the model on the Kinetics-400 for 400 epochs and then transfer to the UCF101. The default settings of fine-tuning are shown in Tab. 9.

HMDB51.[5] The default setting is the same as in UCF101.

[2] https://developer.qualcomm.com/software/ai-datasets/something-something

[3] https://www.deepmind.com/open-source/kinetics

[4] https://www.crcv.ucf.edu/data/UCF101.php

[5] https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/

config	SSv2	K400
optimizer	AdamW [48]	
base learning rate	1.5e-4	
weight decay	0.05	
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$ [11]	
batch size	768	
learning rate schedule	cosine decay [47]	
warmup epochs	40	
flip augmentation	*no*	*yes*
augmentation	MultiScaleCrop	

Table 8. **Pre-training setting.**

config	SSv2	K400	Others
optimizer	AdamW		
base learning rate	1.5e-4		
weight decay	0.05		
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$		
layer-wise lr decay	0.75 [5]		
batch size	128		
learning rate schedule	cosine decay		
warmup epochs	5		
training epochs	30	75	100
flip augmentation	*no*	*yes*	*yes*
RandAug	(9,0.5) [20]		
label smoothing	0.1 [61]		
mixup	0.8 [83]		
cutmix	1.0 [80]		
drop path	0.1 [42]		
repeated sampling	2 [38]		

Table 9. **Fine-tuning setting.**

E. Additional Experimental Results on Ablation Studies

Training speed-up versus performance decrease. To speed up the training procedure during the ablation studies, we (1) randomly select 25% videos from the training set of Something-Something V2 dataset to pre-train the model, (2) randomly drop 50% patches following [55], specifically, we adopt a random masking strategy to select 50% input patches to reduce the computation of self-attention. We present a comparison across the speed-up and performance decrease due to these techniques in Tab. 10. Notice that with an acceptable performance drop (-3.7%), we speed up the ablation experiment by 2.5 times.

Complexity and necessity of data pre-processing steps The data pre-processing includes 4 steps: 1) pre-extracting the optical flow; 2) removing the camera motion; 3) extracting dense trajectories and 4) removing the inaccurate trajectories. We only need to pre-process the data one time, taking less than 24 hours, to get simpler and cleaner reconstruction targets. These targets can be reconstructed using a de-

Data	Input patches	Acc.@1	Acc.@5	Speed-up
100%	100%	**64.8**	**89.7**	1.0×
25%	50%	61.1	86.9	**2.5**×

Table 10. **Training speed-up versus performance decrease**. We report the total time cost of the entire pre-training and fine-tuning procedure. We use a random masking strategy to drop a portion of input patches.

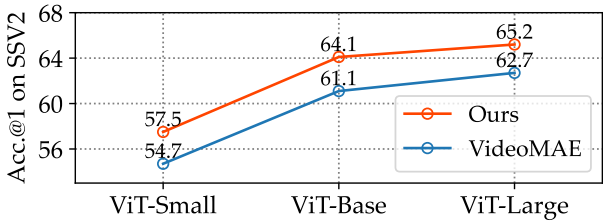

Figure 7. **Comparing MME and VideoMAE on different ViT backbone.** Our MME scales well on different ViT variants.

coder with fewer layers, leading to a faster pretraining process compared with VideoMAE (2.12 vs. 2.24 min/epoch). Besides, removing steps 2 and 4 lowers the performance on SSV2 by 4.4% and 0.7%, respectively.

Scalability on model size of MME. We conduct experiments on Something-Something V2 using different ViT variants. In Fig. 7, both our MME and VideoMAE perform better as the model size grows, and our MME outperforms VideoMAE consistently. This shows the scalability and superiority of our method.

Detailed searching results of hyper-parameters. We conduct ablation studies on MME design in detail. Specifically, we present detailed hyper-parameters searching results on masking ratio and trajectory length design, as shown in Fig. 8 and Fig. 9. Besides, we also complete the ablation results on spatial trajectory density. Extended from Tab. 7c, we increase the number of trajectories per patch from 4 to 16. Performance on SSV2 drops slightly by 0.7%, indicating that 4 trajectories are dense enough.

Comparing with the original version of dense trajectory. We introduce the dense trajectories [66] into our mask motion modeling task to explicitly represent motion by densely tracking trajectories of different object parts. This video descriptor has been successfully applied in action recognition [66]. The iDT [67] further improves the dense trajectories by removing the effects of camera motion, thus enhancing the performance of action recognition. The original version of iDT combines short-term motion targets, including HOF and MBH features to represent motion. However, we found that only using the position changes to represent objects' movement is enough, see Tab. 11.

Comparing with the MaskFeat directly. We provide a direct comparison with MaskFeat using the MViTv2-S backbone by pre-training on K400 for 300 epochs and then fine-tuning on SSvV2, following the recipe in MaskFeat. In Tab. 12, our MME outperforms the MaskFeat by 1.5% on SSv2.

Additional features	Acc.@1	Acc.@5
Traj. + MBH + HOF	63.5	88.2
Traj. (ours)	**63.5**	**88.4**

Table 11. **Combining short-term motion features in a trajectory volume**. Discarding these additional short-term motion features draws a competitive result.

Method	Backbone	SSv2 Acc.@1
MaskFeat	MViTv2-S	67.7%
Ours	MViTv2-S	**69.2% (+1.5%)**

Table 12. Comparison with MaskFeat on K400 and SSV2.

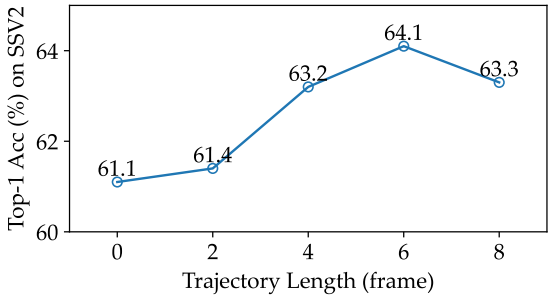

Figure 8. **Ablation study on trajectory length.**

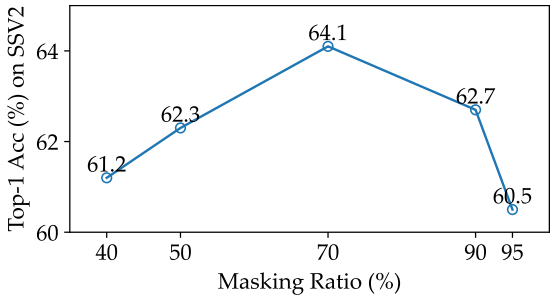

Figure 9. **Ablation study on masking ratio.**

F. More Visualization Results

We present more visualization results of the motion trajectories. Videos are sampled from the Something-Something V2 validation set. For each 3D cube patch shape $2 \times 16 \times 16$, we visualize the first frame and the trajectories start from it. All the trajectories are with length $L = 6$ as our default setting.

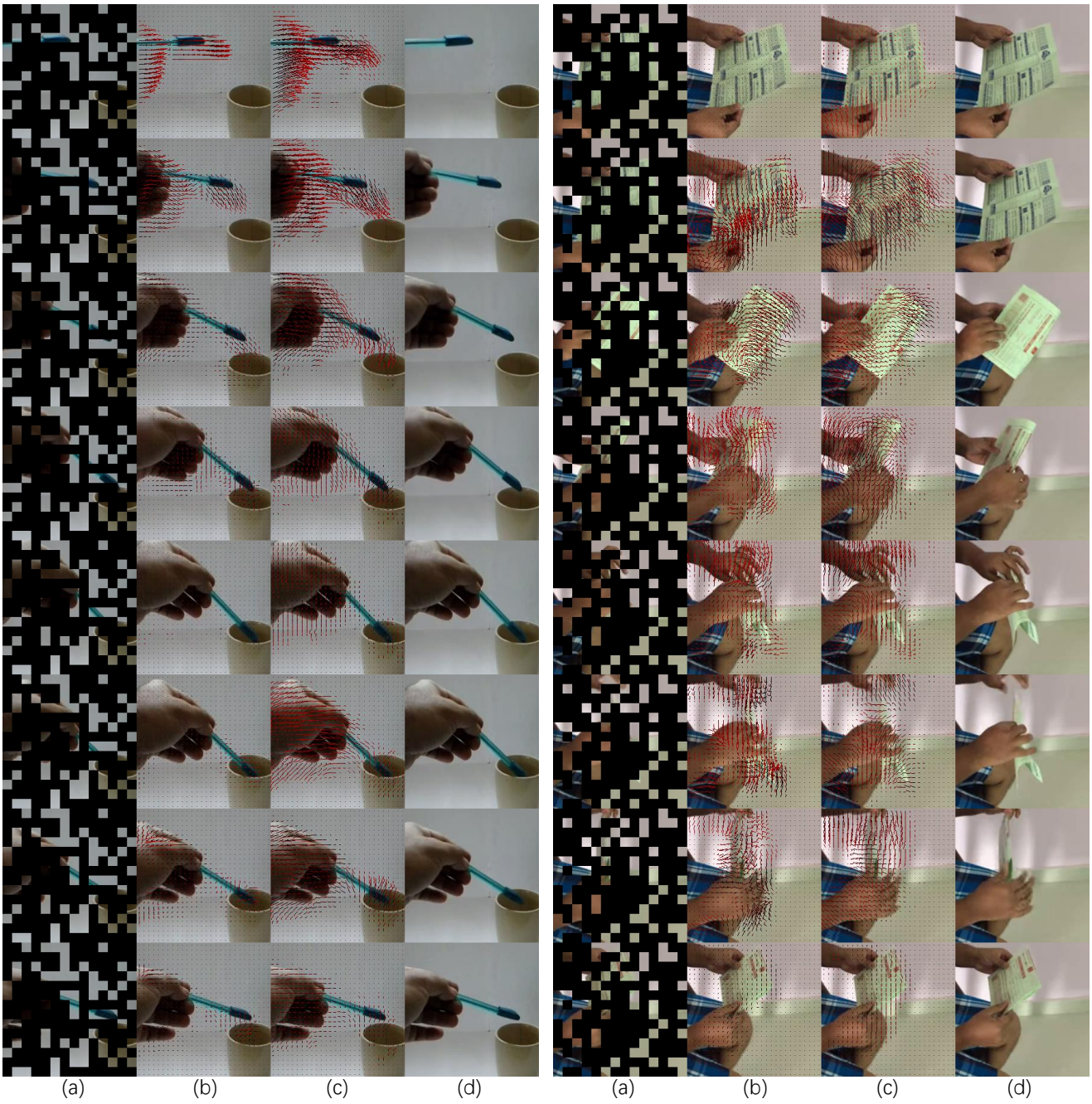

Figure 10. **Visualization of predicted motion trajectory.** Each column represents: (a) Masked frames; (b) ground truth; (c) prediction; (d) original frame. Trajectories are colored from dark to light in chronological order.

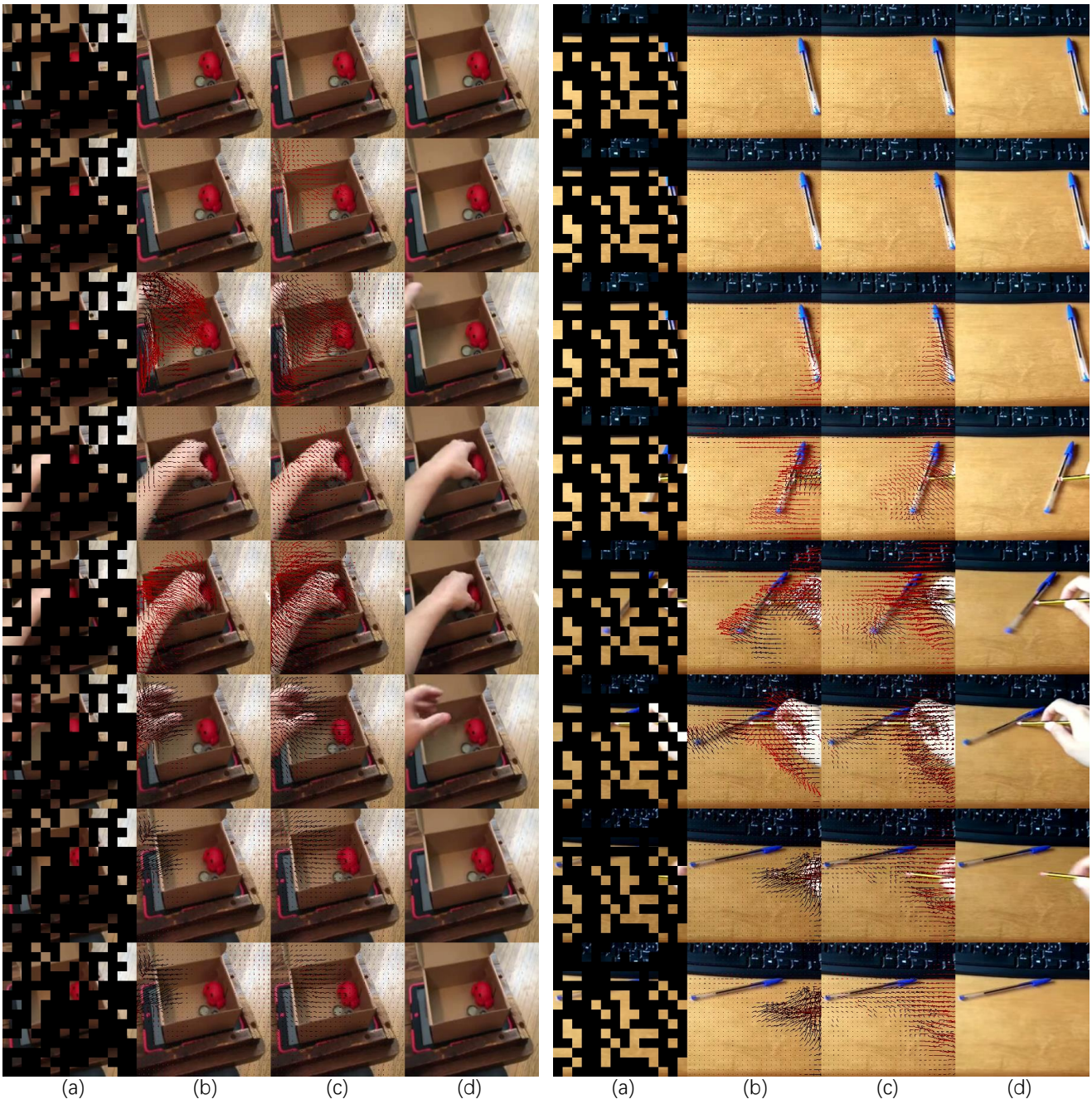

Figure 11. **Visualization of predicted motion trajectory.** Each column represents: (a) Masked frames; (b) ground truth; (c) prediction; (d) original frame. Trajectories are colored from dark to light in chronological order.