Learning Fine-Grained Features for Pixel-wise Video Correspondences
Supplementary Material

Rui Li  Shenglong Zhou  Dong Liu
University of Science and Technology of China, Hefei, China
{liruid,slzhou96}@mail.ustc.edu.cn, dongeliu@ustc.edu.cn
https://github.com/qianduoduolr/FGVC

The supplementary material contains: 1) the detailed network architecture used in our method; 2) more details about the evaluation; 3) more training details; 4) the demo video consisting of several qualitative examples for point tracking.

1. Network Architecture

**Backbone.** Following the prior studies [4, 9, 11], we use the modified ResNet [3] architecture as our backbone, which is illustrated in Table 1. The res5 layer along with the stride in res4, is removed to get the feature maps with a resolution of 1/8 of the original video resolution. We set the stride of the encoder $\theta$ to 2 by further removing the pool1 and the stride in res3 to get more fine-grained features.

**Self-attention and Cross-attention Layers.** In our coarse-to-fine framework, after the coarse-grained features extraction, we send the coarse features $F_{\downarrow 1}, F_{\downarrow 2}$ to the self-attention and cross-attention layers, inspired by the recent studies [7,10] introducing the transformer modules to transform the features into representations that are easy to match. Formally, the attention layer is denoted as:

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T)V.$$  

(1)

For self-attention layer, the input features are the same for $Q/K/V$, while in the cross attention layer, the input features for $Q$ can be either $F_{\downarrow 1}$ or $F_{\downarrow 2}$, and the input features for $K/V$ are $F_{\downarrow 1}$ (when $Q$ is $F_{\downarrow 2}$) or $F_{\downarrow 2}$ (when $Q$ is $F_{\downarrow 1}$). We follow the Linear Transformer [5] proposed to reduce the computational cost of transformer by substituting the exponential kernel, and we set the number of the head to 8 for the feature dimension of 256. Besides, we sequentially interleave the self and cross attention layers in our coarse-to-fine framework by 2 times.

**Up-sampling.** In Figure 1, we illustrate the detailed architecture of the up-sampling layer which consists of several residual blocks [3] and interpolation modules. Note the first interpolation module is performed in the channel dimension, and the latter one is performed over space.

Table 1: Network Architecture of the modified ResNet-18 [3]. The residual blocks are shown in brackets and the kernel size of each convolution is presented followed by the output channels.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Network</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>-</td>
<td>$3 \times 256^2$</td>
</tr>
<tr>
<td>conv1</td>
<td>$7 \times 7, 64$, stride 2</td>
<td>$64 \times 128^2$</td>
</tr>
<tr>
<td>pool1</td>
<td>$3 \times 3, 64$, stride 2</td>
<td>$64 \times 64^2$</td>
</tr>
<tr>
<td>res2</td>
<td>$[3 \times 3, 64] \times 2$, stride 1</td>
<td>$64 \times 64^2$</td>
</tr>
<tr>
<td>res3</td>
<td>$[3 \times 3, 128] \times 2$, stride 2</td>
<td>$128 \times 32^2$</td>
</tr>
<tr>
<td>res4</td>
<td>$[3 \times 3, 256] \times 2$, stride 1</td>
<td>$256 \times 32^2$</td>
</tr>
</tbody>
</table>

2. More details about evaluation

Following previous works [4, 9, 11], all evaluation tasks are cast as video label propagation. This involves making predictions for target pixels in current frames, based solely on the ground-truth annotation for the first frame. We leverage the model as a similarity function to make predictions using k-nearest neighbors. This is a natural choice given the learned representations to align with prior research for ensuring fair comparisons. Specifically, given the labels $L_i \in \mathbb{R}^{N \times D}$ at the frame $i$ ($D$ represents the
number of target pixels or semantic masks), and encoded features $F_i, F_j \in \mathbb{R}^{N \times C}$. We obtain the probabilistic maps $P^j_{i, j} \in \mathbb{R}^{N \times (2r+1)^2}$ within a local range $r$. We take the common practice in [4, 9, 11] to utilize the $k$-nearest neighbor for selecting top-$k$ indexes of values in $P^j_{i, j}(\cdot|q) \in \mathbb{R}^{(2r+1)^2}$ for each query $q$. Then we obtain the selected $\hat{P}_{j, i} \in \mathbb{R}^{N \times k}$ and labels $\hat{L}_i \in \mathbb{R}^{k \times D}$. The predictions $L_j \in \mathbb{R}^{N \times D}$ on the frame $j$ can be computed as $L_j = \hat{P}_{j, i} \hat{L}_i$. Moreover, we adopt a queue of context frames (memory bank) for propagation as is commonly done in previous studies, where the set of pixels of the previous $m$ frames, are utilized for auto-regressive propagation. In our experiments of point tracking on three newly included datasets [1, 2], we set the $r, k$, and $m$ to 24, 10, and 5 for all feature-matching-based methods with a stride of 2, and reduce the $r$ to 18/12 for stride of 4/8.

3. More training details

The training of the coarse-to-fine framework is only performed on the real-world dataset YouTube-VOS [12] with a batchsize of 32, which takes 90k iterations to converge. The initial learning rate is set to 1e-3 with a cosine (half-period) learning rate schedule. In the training process, we regard the initial learning rate $L_{\text{KL}}$ as pseudo labels for the objective function $L_{\text{KL}}$. We perform all experiments on 4 GTX-3090 GPUs.

4. Qualitative examples for point tracking

We select several representative video clips on TAP-VID-DAVIS [2] to verify the effectiveness of our method compared with state-of-the-art methods which provide the code and pre-trained models, e.g., VFS [11], MAST [6] and RAFT [8]. Please refer to the video demo in https://www.youtube.com/watch?v=2ZCVUOiym0U for video-wise qualitative comparisons. Without fine-tuning our pre-trained model on any additional dataset, we propagate the points of the first frame to the current frame. As observed in the enclosed video, the results produced by our method show clear improvements over state-of-the-art methods. The predictions of our method tend to have more accurate and smooth trajectories even facing severe temporal discontinuity, e.g., appearance changes, large motion, and deformations. These examples again verify the effectiveness of our method.

References


