In this supplementary material, Sec. 1 illustrates the details of the algorithm. Sec. 2 and Sec. 3 describe the detailed architecture and experimental settings, respectively. Sec. 4 provides the complexity analysis of trajectory-aware attention in TTVSR. Sec. 5 analyzes the limitations of TTVSR. Finally, Sec. 6 shows more comparison results.

1. Algorithm Details

In this section, we illustrate the details of the algorithm. It includes the pseudocode of the entire algorithm, the detailed analyzes of the proposed cross-scale feature tokenization module, and location map updating mechanism in our TTVSR.

Algorithm pseudocode. As shown in Alg. 1, we describe our proposed TTVSR in the form of pseudocode. Besides, we follow previous works [1, 2] to adopt a bidirectional propagation scheme, where the features in different frames can be propagated backward and forward. For clarity, in this algorithm, we only show the process of forward propagation, and the process of backward propagation is similar to forward propagation.

Cross-scale feature tokenization. As discussed in the main paper, in addition to the more complex motion in the long-range sequence, at the same time, the contents in sequences have the distinct scale changing as it moves. The use of texture from a larger scale often helps to recover more detailed texture on a smaller scale. Therefore, to adapt to the changes of different scales, we propose the cross-scale feature tokenization before trajectory-aware attention to obtain the unified features from the multi-scale. As shown in Fig. 1, we first use the successive unfold and fold operations to expand the receptive field of features, which capture features with more textures on a larger scale. Then, features of different scales are sampled to the same one by a pooling operation, which is used to unify features from different scales. Finally, the features are split by unfolding operation to obtain the output tokens. It is noteworthy that this process can extract features of the whole temporal sequence in parallel, and obtain features of larger scales without adding additional parameters unlike other tasks [6, 15].

Location map updating. As discussed in the main paper,
the location maps will change over time. We denotes the updated location maps as $L^t$ and provides a more detailed and formulated description in this section.

When changing from time $T$ to time $T+1$, first, based on Equ. 7 in the main paper, $L_{m,n}^{T+1}$ represents the coordinate at time $T+1$ of a trajectory which is ended at $(m, n)$ at time $T+1$, which can be expressed as:

$$L_{m,n}^{T+1} = (m, n).$$

Then we update the existing location maps $\{L_{m,n}^0, \ldots, L_{m,n}^T\}$. To build the connection of trajectories between time $T$ and time $T+1$, we introduce a lightweight motion estimation network which computes a backward flow $O^{T+1}$ from $I_{LR}^{T+1}$ to $I_{LR}^T$. This process can be formulated as:

$$O^{T+1} = H(I_{LR}^{T+1}, I_{LR}^T; \theta),$$

where $H(\cdot)$ is composed of the motion estimation network with parameter $\theta$ and an average pooling operation. The average pooling is used to ensure that the output of the motion estimation is the same size as $L^t$.

Finally, we get the updated coordinates in location map $L^t$ by interpolating between its adjacent coordinates:

$$L^t = S(L^t, O^{T+1}),$$

where $S(\cdot)$ represents the spatial sampling operation on matrix $L^t$ by spatial correlation $O^{T+1}$ (i.e., grid_sample in PyTorch). Thus far, we have all the location maps for time $T+1$.

2. Details of Architecture and Runtimes

In this section, we will illustrate the detailed architecture and runtimes of TTVSR.

**Architecture.** We follow IconVSR [1], the feature extraction network uses five residual blocks, which are part of embedding operation $\phi(\cdot)$. The feature reconstruction network $R(\cdot)$ uses 60 residual blocks, which are part of embedding operation $\varphi(\cdot)$. The channel number of feature is set to 64.

**Runtimes.** As shown in Tab. 1, we analyze the parameter size and runtime of each component in TTVSR. The runtime is computed on one LR frame with the size of $180 \times 320$ and $\times 4$ upsampling, and all models are conducted on an NVIDIA Tesla V100 GPU.

As shown in Tab. 1, our method is much lighter and faster than other attention-based methods (e.g., MuCAN [5], which is the SOTA attention-based method), this is thanks to our efficient design of trajectory-aware attention. Compared with other methods without attention mechanisms, TTVSR achieves higher results with the smaller parameters size and comparable FLOPs (as shown in main paper Table 3). While it’s worth noting that TTVSR is slower than IconVSR [1]. This is because the runtime highly depends on the code implementation and hardware platforms. Specific to our case, the attention mechanism contains a lot of small matrix multiplications which limits our method to run at the high efficiency (as shown in the fourth row of Tab. 1, the time used to calculate attention is more than half of the total time). However, we believe that with the optimization of code and hardware, TTVSR can achieve the comparable runtime to IconVSR [1] as demonstrated by FLOPs which is independent of hardware.

3. More Experimental Settings

**Datasets.** We use the REDS [9] and Vimeo-90K [11] datasets to construct our training data. For REDS [9], we apply the MATLAB bicubic downsample (BI) degradation on REDS4 [9] to evaluate TTVSR. For Vimeo-90K [11], we use the Gaussian filter with a standard deviation of $\sigma = 1.6$ and downsampling (BD) degradation, and use Vid4 [7], UDM10 [12] and Vimeo-90K-T [11] as test sets along with it.

**Settings.** To leverage the information of the whole sequence, we follow previous works [1,2] to adopt a bidirectional propagation scheme. In the process of bidirectional propagation, the same parameters are shared, and the final feature at each frame is obtained by cascading bidirectional output features.

For REDS [9], we use sequences with a length of 50 as inputs, and loss is computed for the 50 output frames. For Vimeo-90K [11], we augment the sequence by flipping twice to extend the length of the sequence to 28 as input, and the inference results on Vimeo-90K-T [11] are the average of output frames derived from the same frame extension.

The Charbonnier penalty loss [4] is applied on whole sequence between the ground-truth and restored SR frame. It can be formulated as:

$$\ell = \frac{1}{T} \sum_{t=1}^{T} \sqrt{\|I_{HR}^t - I_{SR}^t\|^2 + \epsilon^2},$$

where $\epsilon = 1 \times 10^{-8}$. $T$ denotes the sequence length. All
Table 2. Quantitative comparison (PSNR↑, SSIM↑, and LPIPS↓) on the REDS4 [9] dataset for 4× video super-resolution. The results are tested on RGB channels. Red indicates the best and blue indicates the second best performance (best view in color).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIM</td>
<td>0.7292</td>
<td>0.8200</td>
<td>0.8196</td>
<td>0.7990</td>
<td>0.8251</td>
<td>0.8800</td>
<td>0.8750</td>
<td>0.8909</td>
<td>0.8948</td>
<td>0.9021</td>
</tr>
<tr>
<td>LPIPS</td>
<td>0.3395</td>
<td>0.2716</td>
<td>0.2668</td>
<td>0.2969</td>
<td>0.2911</td>
<td>0.2225</td>
<td>0.2112</td>
<td>0.1979</td>
<td>0.1890</td>
<td>0.1786</td>
</tr>
</tbody>
</table>

4. Complexity Analysis

By introducing trajectories into Transformer in our proposed TTVSR, the attention calculation on $K$ and $V$ can be significantly reduced because it can avoid the computation on spatial dimension compared with vanilla vision Transformers. In this section, we analyze the complexity of trajectory-aware attention calculation in detail.

Take the attention process of one token in $Q$ as an example, when a sequence of length $T$ and size $C \cdot H \cdot W$ is input as $K$, the size of tokens and the number of tokens can be expressed as $C \cdot D_h \cdot D_w$ and $T \cdot \frac{H}{D_h} \cdot \frac{W}{D_w}$, respectively. The similarity of attention mechanisms in vanilla vision Transformer has a computational cost of:

$$\left(T \cdot \frac{H}{D_h} \cdot \frac{W}{D_w}\right) \cdot (C \cdot D_h \cdot D_w). \tag{5}$$

The similarity of trajectory-aware attention mechanisms in TTVSR has a computational cost of:

$$\left(T \cdot 1 \cdot 1\right) \cdot (C \cdot D_h \cdot D_w), \tag{6}$$

here, the attention calculation on spatial dimension is avoided, so the number of tokens in the attention calculation is reduced from $(T \cdot \frac{H}{D_h} \cdot \frac{W}{D_w})$ to $(T \cdot 1 \cdot 1)$.

By expressing attention as the multiplication of tokens, we get a reduction in computation of:

$$\frac{T \cdot 1 \cdot 1 \cdot (C \cdot D_h \cdot D_w)}{(T \cdot \frac{H}{D_h} \cdot \frac{W}{D_w}) \cdot (C \cdot D_h \cdot D_w)} = \frac{1}{\left(\frac{H}{D_h} \cdot \frac{W}{D_w}\right).} \tag{7}$$

In general, by introducing trajectories into Transformer, the computational complexity of attention is reduced by $(\frac{H}{D_h} \cdot \frac{W}{D_w})$ times, and provides a more efficient way to enable our TTVSR can directly leverage the information from a distant video frame.

5. Limitation Analysis

In this section, we discuss the limitations of TTVSR.

Length of sequence. The design of TTVSR is to fully utilize the temporal information of more frames in the sequence, so the performance improvement is limited for shorter sequences. (e.g., each sequence contains seven frames on Vimeo-90K [11]). Therefore, to enable long-range sequence capability, for Vimeo-90K [11], we augment the sequence by flipping twice to extend the length of the sequence to 28 as input. However, due to the limitation of original sequences, the improvement is still limited. We will add relevant features on spatial dimensions to further enhance the utilization of spatial information in the model.

Training Time. To make the model have the ability to model long-range sequences, we input the sequences as long as possible during training. However, longer sequences often lead to longer training times. Next, we will optimize the training process so that TTVSR can speed up the training process with long sequences input.

GPU memory. According to the model design, TTVSR needs to store the features of each moment during inferring and training. It inevitably takes up GPU memory. Therefore, the next step is to design an adaptive feature storage mechanism dynamically and selectively retain the useful features for reconstruction.

6. More Results

In this section, to further verify the effectiveness of our method, we compare the perceptual results and show more comparison results among the proposed TTVSR and other advanced methods on four different benchmarks.

Perceptual results. We use LPIPS [13] as a widely used metric to evaluate perceptual quality. Results, shown in the following Additional Tab. 2, demonstrate that TTVSR is still highly superior in the perceptual metrics.

Visualization results. We show more comparison results among the proposed TTVSR and other advanced methods on four different benchmarks. The results on REDS4 [9], Vid4 [7], UDM10 [12] and Vimeo-90K-T [11] are shown in Fig. 2-3, Fig. 4, Fig. 5 and Fig. 6-7, respectively.
Figure 2. Visual results on REDS4 [9] for $4 \times$ scaling factor. The frame number is shown at the bottom of each case. Zoom in to see better visualization.
Figure 3. Visual results on REDS4 [9] for 4× scaling factor. The frame number is shown at the bottom of each case. Zoom in to see better visualization.
Figure 4. Visual results on Vid4 [7] for $4 \times$ scaling factor. The frame number is shown at the bottom of each case. Zoom in to see better visualization.
Figure 5. Visual results on UDM10 [12] for $4 \times$ scaling factor. The frame number is shown at the bottom of each case. Zoom in to see better visualization.
Figure 6. Visual results on Vimeo-90K-T [11] for $4 \times$ scaling factor. The frame number is shown at the bottom of each case. Zoom in to see better visualization.
Figure 7. Visual results on Vimeo-90K-T [11] for $4 \times$ scaling factor. The frame number is shown at the bottom of each case. Zoom in to see better visualization.
References


