Multi-Scale Memory-Based Video Deblurring

Bo Ji Angela Yao
National University of Singapore
{jibo,ayao}@comp.nus.edu.sg

Abstract

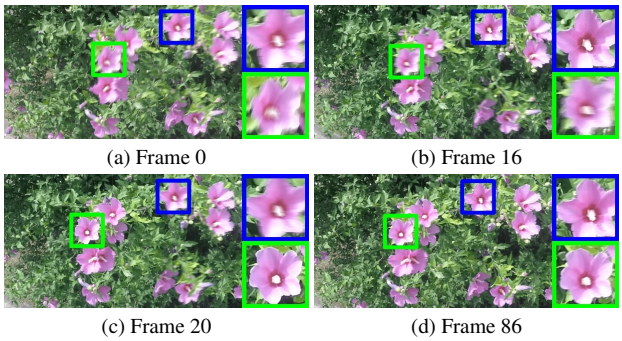
Video deblurring has achieved remarkable progress thanks to the success of deep neural networks. Most methods solve for the deblurring end-to-end with limited information propagation from the video sequence. However, different frame regions exhibit different characteristics and should be provided with corresponding relevant information. To achieve fine-grained deblurring, we designed a memory branch to memorize the blurry-sharp feature pairs in the memory bank, thus providing useful information for the blurry query input. To enrich the memory of our memory bank, we further designed a bidirectional recurrency and multi-scale strategy based on the memory bank. Experimental results demonstrate that our model outperforms other state-of-the-art methods while keeping the model complexity and inference time low. The code is available at https://github.com/jibo27/MemDeblur.

1. Introduction

Video deblurring is a core restoration and enhancement task aiming to recover a sharp video from a blurry input video. Blur in videos arises from various sources, *e.g.* object motion, camera shake, and depth of field. Video deblurring is a highly ill-posed problem as multiple sharp sources may correspond to a single blurred result. The problem becomes especially pertinent as more and more videos are captured by smartphones.

Video deblurring distinguishes itself from the image deblurring task in that it is critical to use the information from the entire sequence of frames effectively. To do so, window-based models [17,22,23,27] feed consecutive blurry frames to an encoder-decoder to directly restore a sharp frame. Recurrent models [6,15,30,35,36] on the other hand maintain a hidden state across the frames to sequentially propagate features from the first frame to the last. However, these methods cannot utilize the information from *all* the frames in the video sequence. Moreover, neighbouring frames are often used in simplistic manners, *e.g.* via an alignment.

An underexplored aspect in video-deblurring is that blurring occurs from non-uniform blur kernels. The blur artifacts differ not only for the same object across different frames but also for different objects or regions in the same frame. As shown in Fig. 1, the same flower in different frames of a sequence exhibits different extents of blur (compare the blue sample in frame 16 and 86 vs. 0 and 20). Different flowers in the same frame (see frames 16, 20) also have various blur distortions due to object motion. This makes spatio-temporal information aggregation difficult as it is necessary to provide the appropriate information to different regions of a frame.



(a) Frame 0 (b) Frame 16 (c) Frame 20 (d) Frame 86

Figure 1. Different image regions have different blurry artifacts.

To remedy this problem, we adopt the principle of memory networks [2, 12, 16, 24]. Memory networks were originally developed for language modeling [24] and are currently popular in vision for video object segmentation [2, 16]. In segmentation, the memory encodes high-level semantics like objects, but they have not been explored in the context of low-level enhancement tasks like deblurring. The memory networks used in our model record blurry-sharp feature pairs. We compute a spatio-temporal attention between each location of the memory features and that of a query frame region to find helpful sharp information. The implicit advantage is that even if the blur kernel is unknown, we can still extract the corresponding sharp information simply by matching the queried region with memorized features.

Different from previous works using memory networks, we use memory to supplement information to the deblurring backbone. The core deblurring branch is still responsible for recovering low-level details. To enrich the features

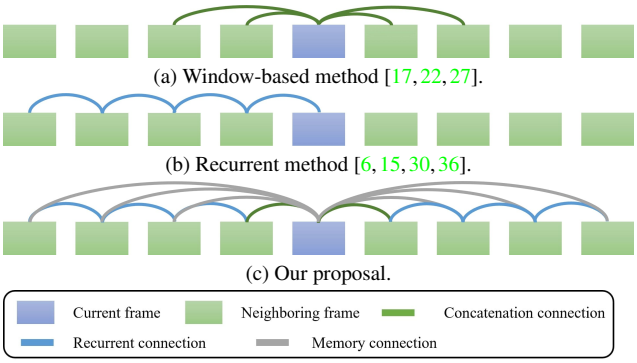


Figure 2. Temporal receptive field comparison between different methods for video deblurring. Window-based approach reconstructs sharp images referencing neighbouring frames within a small fixed-size window. Recurrent methods collectively aggregate frame features till up-to-date to provide restoration cues. In contrast, our method allows the feature propagation from the entire sequence when restoring the given frame.

in the memory bank, we propose a bidirectional and multi-scale structure that better captures information from the entire video sequence at different scales. The multi-scale design also allows our model to handle large displacements more effectively. Our contributions can be listed as follows:

- We present a novel memory-based architecture that stores blurry and sharp spatio-temporal patterns to achieve fine-grained video deblurring. To the best of our knowledge, we are the first to adopt a memory network for a video enhancement task.
- To increase the diversity of memories in the memory bank, we developed a bidirectional and multi-scale structure based on the memory bank. The multiple scales share the same memory bank, which allows cross-scale matching and effective handling of large motions.
- The experimental results demonstrate that our model achieves superior results than state-of-the-art methods under comparable computational budgets.

2. Related work

Video deblurring methods. Image or video deblurring methods are widely used in computer vision tasks, such as SLAM [9], 3D reconstruction [19] and visual tracking [31]. In recent years, deep learning methods have achieved remarkable success in low-level enhancement tasks such as super-resolution [10, 18, 33], denoising [4, 7] and deblurring [14, 22, 25]. Contemporary deep learning video deblurring methods can be roughly divided into window-based and recurrent methods. Window-based methods [17, 22, 27] (see Fig. 2a) take 3-7 consecutive frames and try to reconstruct the middle frame. One derivative [23] uses reinforcement learning to select the most effective frames based on the current input and concatenate them with the input of the model.

Recurrent models [6, 15, 30, 35, 36] deblur frames in sequence, maintaining a recurrent hidden state to propagate information from frame to frame (see Fig. 2b). ESTRNN [35] proposes a spatio-temporal attention module to fuse the neighboring features further. Our model is also recurrent, though we supplement the hidden vector with information from the memory network to extend the range of contextual information (see Fig. 2c)

Existing methods, be it window-based or recurrent, do not consider the characteristics of a localized region within a frame. Furthermore, they utilize only a portion of the frames in the sequence. Our method uses memory banks to provide appropriate spatio-temporal information for each region and exploits the features of all frames.

Memory networks. Memory networks are a class of learning models [29] that construct an external *memory bank* module to store potentially useful features for future use. Memory networks were originally developed for NLP applications [12, 24, 29], but the property of storing features lends itself naturally to being used in video where there is large redundancy across frames. As such, memory networks have been extended to several other video tasks such as movie understanding [13], object tracking [32] and, more recently, video object segmentation [2, 11, 16, 20].

To the best of our knowledge, memory networks have not been used for low-level enhancement tasks like video deblurring. As memory networks traditionally encode semantic information, it is not known if they can effectively handle the underlying feature reconstruction of low-level tasks. Different from existing works, we designed a new memory network customized for the video deblurring task. Especially noteworthy is our novel multi-scale memories which can be shared across different scales.

3. Approach

We adopt a multi-scale memory-based structure to solve the video deblurring task. There are two branches: the deblurring branch and the memory branch (see Fig. 3). The deblurring branch features a bidirectional recurrent structure, while the memory branch stores blurry-sharp feature pairs.

3.1. Preliminaries

Given a blurry video sequence $\mathbf{I} = \{I_1, \ldots, I_i, \ldots, I_N\}$ as input, our objective is to recover a deblurred output sequence $\mathbf{R} = \{R_1, \ldots, R_i, \ldots, R_N\}$. Considering a blurry frame i, the recovered frame R_i can be given by a learned

decoding or "upsampling"[1] $\mathcal{U}(\cdot)$ of a hidden vector h_i, *i.e.*

$$\begin{aligned} R_i &= \mathcal{U}(h_i), \\ \text{where } \ h_i &= \mathcal{F}([x_i, x_{i-1}, h_{i-1}, m_i]), \end{aligned} \tag{1}$$

where $\mathcal{F}$ represents a recurrent feature extraction module, $\mathcal{D}$ is a learned encoding or "downsampling", $x_i = \mathcal{D}(I_i)$, $x_{i-1} = \mathcal{D}(I_{i-1})$ and m_i is a retrieved deblurred memory feature. From Eq. (1), one can see that the feature extraction module $\mathcal{F}$ has a recurrent structure, in that h_i relies on the current downsampled input feature x_i, retrieved m_i and also the previous h_{i-1}. We also feed the previous downsampled feature x_{i-1} into the feature extraction module as this has been shown to be effective [3].

3.2. Memory-enhanced feature aggregation

The memory bank stores the blurry-sharp feature pairs in the latent space so that the relevant sharp features are retrieved with the blurry query input. We consider past restored frames as memory frames saved in the memory bank and the current blurry frame as the query frame.

Saved memory frames follow a key-value format. The key is encoded from the blurry frame I_i, while the values are features extracted from the corresponding hidden features h_i and deblurred result R_i. Given a query frame, we compare it with memory frames in the key space and retrieve the associated values. Then, we decode the values back as relevant effective information for the query input. For clarity, we ignore the bidirectional design and scale level s when describing the details of the memory alignment.

Key encoding. We reuse the downsampling module $\mathcal{D}$ to reduce input frame I_i into x_i, though the spatial size is still too large for the matching between the query and memory keys. Inspired by [16], we use a key encoder $\mathcal{K}$ to further reduce the computational overhead to a key k_i:

$$k_i = \mathcal{K}(x_i). \tag{2}$$

Value encoding. Regarding the value, we apply the encoder-decoder architecture and save the values in the latent space to reduce the memory cost and computational cost. The value encoding module $\mathcal{V}$ is responsible for encoding the corresponding pair into their latent values:

$$\begin{aligned} v_i^r &= \mathcal{V}_r([x_i, r_i]), \\ v_i^h &= \mathcal{V}_h([x_i, h_i]), \end{aligned} \tag{3}$$

where $r_i = \mathcal{D}(R_i)$ and $[\cdot]$ represents the concatenation operation. While it is feasible to directly store a value triplet, *i.e.* $v_i = \mathcal{V}([x_i, r_i, h_i])$, we find through preliminary studies that such a design is too limiting and degrades the performance (see Section 5), likely because it does not distinguish the different roles of r_i and h_i.

For more flexibility, we store v_i^r and v_i^h in two different memories. For storage, we concatenate them with the current memory $\mathbf{v}^{r,M}$, $\mathbf{v}^{h,M}$ and update:

$$\begin{aligned} \mathbf{k}^M &= [\mathbf{k}^M, k_i] \\ \mathbf{v}_i^{r,M} &= [\mathbf{v}_{i-1}^{r,M}, v_i] \\ \mathbf{v}_i^{h,M} &= [\mathbf{v}_{i-1}^{h,M}, v_i]. \end{aligned} \tag{4}$$

Note that the two values share the same key. While it is possible to update v^M with values at every temporal index i, increasing the frequency of memorization increases the memory cost without necessarily improving the performance. As such, we update the memory with every T frames.

Memory readout. Here, we ignore the frame index i for a given query key k_i and use q to denote the q-th location of k_i as k_q^Q. Similarly, we denote k_p^M as the p-th location of the memory key $\mathbf{k}^M$. Then, we compute the affinity matrix $\mathbf{S}$ between k_q^Q and k_p^M:

$$\mathbf{S}_{q,p} = d(k_q^Q, k_p^M), \tag{5}$$

where d is any similarity measure. The affinity matrix $\mathbf{S}$ is then normalized by a softmax to matrix $\mathbf{W}$:

$$\mathbf{W}_{q,p} = \frac{\exp(\mathbf{S}_{q,p})}{\sqrt{C^k} \cdot \sum_z \exp(\mathbf{S}_{q,z})}, \tag{6}$$

where C^k is the key dimension and $\sqrt{C^k}$ serves as a normalization term [26]. This operation is a form of spatio-temporal attention, where $\mathbf{W}$ can be viewed as an attention map, as we are looking for locations on the memory frames that are most relevant for restoring the current query location.

The matrix $\mathbf{W}$ is shared across the readout operation for two memory values, v_i^r and v_i^h, where i is the frame index. The readout memory for the query key k_i is given as

$$\begin{aligned} v_i^{r,Q} &= \mathbf{v}_i^{r,M}\mathbf{W} \\ v_i^{h,Q} &= \mathbf{v}_i^{h,M}\mathbf{W}, \end{aligned} \tag{7}$$

where $\mathbf{v}_i^{r,M}$ and $\mathbf{v}_i^{h,M}$ represent the current memory for the two pairs when we try to restore i-th frame I_i, $v_i^{r,Q}$ and $v_i^{h,Q}$ are the corresponding aggregated memory outputs.

Memory decoding. The readout memory $v_i^{r,Q}$ and $v_i^{h,Q}$ are further decoded by the corresponding decoders $\mathcal{G}_r$ and

[1] We abuse the terms "upsampling" and "downsampling" to emphasize a transformation in both features space and spatial dimensionality.

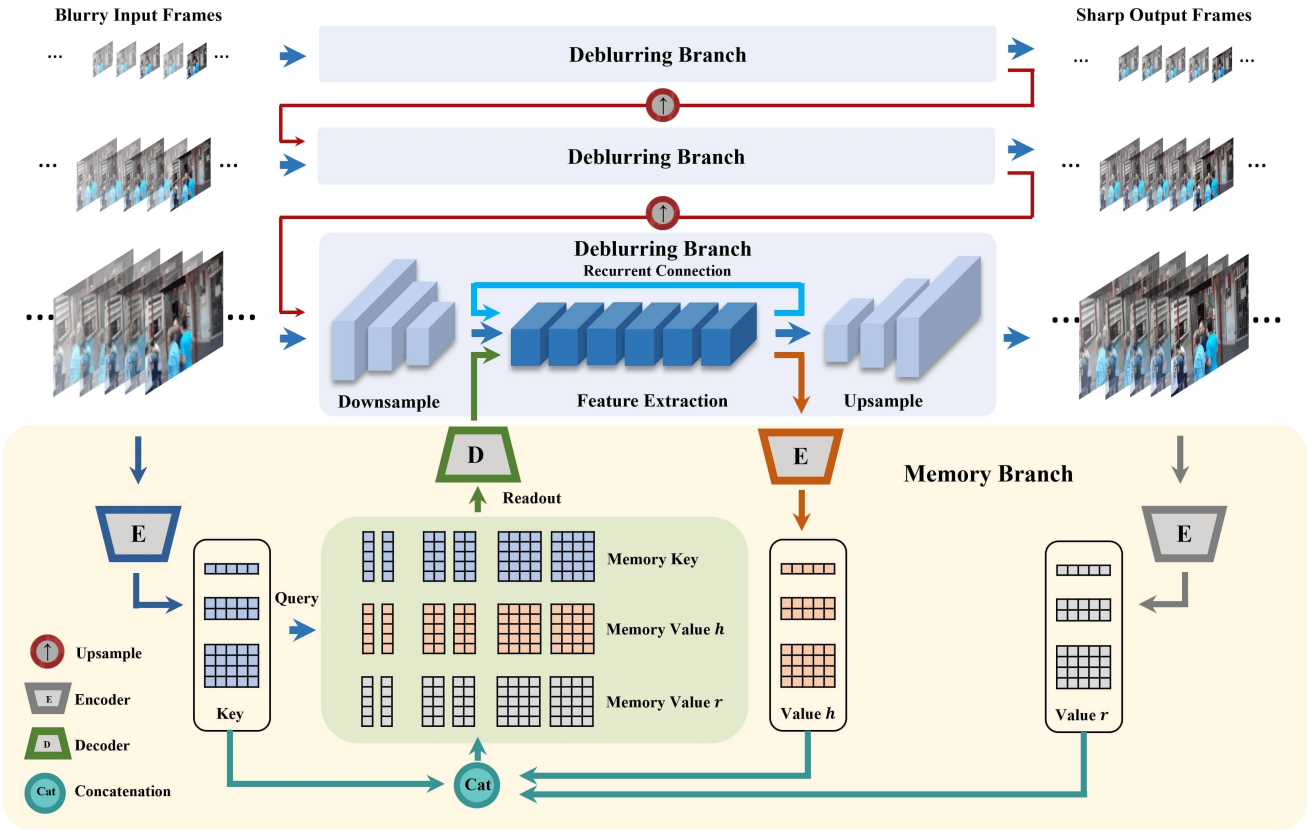


Figure 3. **Overview.** The input is represented at three different scales. We recover the entire video sequence at the corresponding scale, starting from the lowest scale, until the sequence of the original input size is restored. For each scale, we sequentially restore from the first to the last frame. The same memory bank is shared between scales, which ensures information propagation across multiple scales.

$\mathcal{G}_h$:

$$m_i^{r,Q} = \mathcal{G}_r(v_i^{r,Q}) \\ m_i^{h,Q} = \mathcal{G}_h(v_i^{h,Q}). \tag{8}$$

We then concatenate the two decoded memories to get m_i:

$$m_i = [m_i^{r,Q}, m_i^{h,Q}], \tag{9}$$

where m_i is the final memory output for the input frame I_i. As the encoders $\mathcal{K}$ and $\mathcal{V}$ downsample the input feature x_i, the decoder $\mathcal{G}$ upscales the memory back to the original spatial size.

3.3. Bidirectional recurrency

If a video sequence is deblurred in a uni-directional manner, *e.g.* forwards, then frame I_i receives context and memory only from frames up to $i-1$. Yet, there likely exist helpful details in frames after $i+1$. Therefore, we make the recurrent framework bidirectional, incorporating a forward and a backward pass of the sequence to further enrich the memory bank. Just as we establish a memory bank in the forward pass of the sequence, we can also do the same for the backward pass. Note that the sharp frame is restored with information from both passes. As we perform the backward pass first, there are no sharp frames available during the backward procedure, so we encode only the hidden features h_i when establishing the backward memory bank.

For the backward pass, the input sequence is fed in as $\{I_N, I_{N-1}, \ldots, I_1\}$, where each recurrent unit operates as

$$h_i^b = \mathcal{F}_b([x_i, x_{i+1}, h_{i+1}^b, m_i^b]), \tag{10}$$

and h_i^b is the hidden state of the backward module $\mathcal{F}_b$ for i-th downsampled feature x_i, m_i^b is the aggregated memory with the query key k_i. We store the forward and backward features separately in two memory banks. This allows the two memories to be extracted separately during the forward pass, which we find to be more beneficial for the model to distinguish and learn. Then, in the forward pass, the feature extraction module can utilize the backward memory to refine the current features:

$$h_i^f = \mathcal{F}_f([x_i, x_{i-1}, h_{i-1}^f, m_i^f, m_i^b]), \tag{11}$$

where h_i^f is the hidden state of the forward module $\mathcal{F}_f$, and m_i^f is the memory aggregated from the previous frames. We

fuse h_i^f and h_i^b using one convolutional layer to get h_i:

$$h_i = \text{conv}([h_i^f, h_i^b]) \tag{12}$$

The final reconstruction operation is given by Eq. 1.

The bidirectional design is important. The backward module not only provides the visibility of the future frames, but also gives more sufficient information for the restoration of the first frame x_1 in the forward pass. In Eq. 11, as x_1 is the first frame in the video sequence, x_{i-1}, h_{i-1}^f and m_i^f are all initialized as zeros. If we do not retrieve memory from the backward pass, i.e., m_i^b, the model only uses the information of the current frame to perform the deblurring. In that case, it downgrades into a single-image deblurring task and would produce an inferior result. Moreover, in Eq. 12, the backward pass provides an additional helpful feature h_i^b.

3.4. Multi-scale design

Memorizing the relationship among the downsampled feature x_i, hidden state h_i, and sharp frame R_i occupies memory and computational cost. To balance efficiency and performance, we only memorize every T frame, as mentioned in Section 3.2, and perform the temporally intensive memorization for downsample sequences. This motivates us to adopt a multi-scale strategy. Multi-scaling has proven to be effective in image deblurring [14, 25] but is not yet explored in video tasks.

Adopting a multi-scale strategy causes limited memory and computational overhead, but comes with the key advantage of allowing the matching between patterns at different scales. It also handles large displacement more effectively, as downscaling reduces the size of the original (large) displacements.

To consider multiple scales, we downsample the input frame $I_i \in \mathbb{R}^{H\times W\times C}$ into $I_i^2 \in \mathbb{R}^{H/2\times W/2\times C}$ and $I_i^3 \in \mathbb{R}^{H/4\times W/4\times C}$, where the original input I_i can be considered as I_i^1. Note that the downsampling and upsampling in the multi-scale strategy are different from downsampling module $\mathcal{D}$ and upsample module $\mathcal{U}$ in the deblurring branch. The former uses a deterministic method, *e.g.* bilinear interpolation, as the objective is purely scaling and not feature extraction. At scale level s for the forward pass, we obtain the hidden state by

$$h_i^{f,s} = \mathcal{F}_f([x_i^s, x_{i-1}^s, x_i^{s+1\uparrow}, m_i^{f,s}, m_i^{b,s}]), \tag{13}$$

where $h_i^{f,s}$ is the hidden state at scale level s, $x_i^{s+1\uparrow}$ is the upscaled version of the downsampled feature from the deblurred frames at scale s+1, *i.e.* $x_i^{s+1\uparrow} = (\mathcal{D}(R_i^{s+1}))\uparrow$, $m_i^{f,s}$ and $m_i^{b,s}$ are aggregated memories from forward and backward memory banks for the given input x_i^s. The $\uparrow$ denotes bilinear upsampling.

The forward module of different scales share the same memory bank. This makes it possible for the query frame to match features to multiple scales and enhances the memory utilization. The shared-scale memory is a key distinction of our work from previous multi-scale methods.

3.5. Architecture

We use residual dense blocks [34], residual blocks [10] and transposed convolution as the basic building blocks for $\mathcal{D}$, $\mathcal{F}$ and $\mathcal{U}$ in the deblurring branch, respectively. For our memory branch, the 3 encoders $\mathcal{K}$, $\mathcal{V}_r$ and $\mathcal{V}_h$ are of the same architecture, which is residual blocks followed the first stage of pre-trained ResNet50 [5]. The decoder $\mathcal{G}$ combines the output of 2 decoders $\mathcal{G}_r$ and $\mathcal{G}_h$, which contains residual blocks and a single $\times 4$ pixel shuffle layer [21]. More details are in the Supplementary.

4. Experiments

4.1. Setting

Dataset. We experimented on the GOPRO dataset [14], which features 22 training sequences with 2103 frames and 11 validation sequences with 1111 frames. We experimented with two variants: the original version [17] and the downsampled version with gamma correction [15, 35], which downsamples the original videos from 1280×720 to 960×540 to reduce noise and video compression artifacts.

Evaluation metrics. We use the peak signal-to-noise ratio (PSNR) and SSIM [28] to evaluate the deblurred results. For complexity, we compare the runtime and the multiply-accumulate operation (MAC). The runtime is calculated per frame on the input video containing 100 frames using a single NVIDIA RTX A5000 GPU. Both runtime and MAC assume the input frame is of shape $720 \times 1280 \times 3$.

Implementation details. For training, we used the ADAM optimizer [8] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The initial learning rate was set as 0.0005 and was decayed by half in $[200, 350, 450, 500]$ epochs. We trained the model for 600 epochs in total. For augmentation, we applied random rotations and flipping. To manage the memory for scale s, we set $T_1 = 5$, $T_2 = 2$ and $T_3 = 1$. The training patch size was 256×256, and the batch size was 8. The training subsequence contained 8 frames. To manage the memory cost and run-time, we empirically maintained a maximum of five recent frames in the memory. This technique reduced the runtime to one-fifth of the original , while the PSNR dropped by only 0.02dB. We borrowed the multi-scale content loss [14] to train our model, but we replaced the mean-square error with Charbonnier loss [1].

4.2. Comparisons with state of the art

A quantitative comparison on the Downsampled GoPro dataset shown in Table 1 and on the Original GoPro dataset shown in Table 2 indicates that our model achieves the best performance compared with other state-of-the-art models.

Table 1. Quantitative comparison on the downsampled GOPRO dataset [14].

Model	STRCNN [6]	DBN [22]	IFIRNN($c2h2$) [15]	IFIRNN($c2h3$)	ESTRNN(B_9C_{80}) [35]	ESTRNN(B_9C_{90})	Ours (Slim)	Ours
PSNR	28.74	29.91	29.92	29.97	30.79	31.07	31.21	31.77
SSIM	0.8465	0.8823	0.8838	0.8859	0.9016	0.9023	0.9203	0.9275
GMACs	276.2	784.75	167.09	217.89	163.61	206.70	197.34	344.49

Table 2. Quantitative comparison on the original GOPRO dataset [14]. We do not fill in the GMACs of STFAN [36] as the GMACs calculated for filter adaptive layer may be not accurate.

Model	SRN [25]	DBN [22]	Kim et al. [6]	EDVR [27]	STFAN [36]	ESTRNN [35]	CDVD-TSP [17]	Ours
PSNR	30.29	27.31	26.82	26.83	28.59	30.91	31.67	31.76
SSIM	0.9014	0.8255	0.8245	0.8426	0.8426	0.9091	0.9275	0.9230
GMACs	1527.01	784.75	276.2	468.25	-	204.19	5122.29	344.49

Table 3. Runtime comparison with state-of-the-art models.

Model	SRN	STRCNN	DBN	IFI-RNN
Runtime (s)	0.222	0.0448	0.085	0.054
Model	ESTRNN	CDVD-TSP	Ours(Slim)	Ours
Runtime (s)	0.083	1.015	0.079	0.191

For Downsampled GoPro, we designed a slim version of our model to match our GMACs with ESTRNN [35]. The slim version does not use the bidirectional and multi-scale design, yet it outperforms other state-of-the-art methods. Table 3 also shows that our runtime is less than ESTRNN. Our final model, which includes the bidirectional and multi-scale structures, uses 147.15 GMACs more than the slim version and improves performance by 0.57dB.

On the Original GoPro dataset in Table 2, the closest competing state-of-the-art model is CDVD-TSP [17]. Our model has 0.09dB higher PSNR than CDVD-TSP even though the GMACs is an order of magnitude less (7% specifically) and the runtime is one-fifth that of CDVD-TSP (see Table 3). CDVD-TSP has significantly higher computational complexity, because it relies on optical flow and uses a cascaded training strategy.

We present the qualitative comparisons in Fig. 4 and 5. Fig. 4 shows the performance on a frame with mild blur, while Fig. 5 presents the performance on consecutive frames with severe blur due to object motion. Our model recovers clearer and sharper images than other state-of-the-art models. Even on video sequences with particularly large motion and severe blur, our method can still recover some image structures. For example, in Fig. 5, only our deblurred images can be seen to have roughly 5 characters. This is because our memory bank aggregates information of the whole video sequence and stores diverse features, so that when processing extremely blurred images, we can find relevant memory features from the memory bank to help restore them. More results are provided in the Supplementary.

5. Network analysis

We conducted extensive ablation studies to verify our approach. All ablations in this section were performed on Downsampled GOPRO [14].

Table 4. Ablation of model components.

Memory	Bidirection	Multi-scale	PSNR	SSIM
			30.88	0.9145
✓			31.22	0.9203
✓	✓		31.63	0.9256
✓		✓	31.44	0.9237
✓	✓	✓	**31.79**	**0.9278**

Effects of proposed components. We evaluate the memory branch, bidirectional recurrency and our multi-scale strategy and show results in Table 4. The vanilla version with deblurring branch only achieves the worst performance. Adding the proposed components increases both PSNR and SSIM. The bidirectionality has slightly more impact than the multi-scale architecture, though the two components complement each other.

Memory branch design. We first followed a naive approach from the original memory network [16, 24, 29], where we only had memory branches and decoded the memory directly into a sharp frame with an upsample module (*'w/o Deblur.'* in Table 5), to establish our baseline. Fig. 8 shows that this naive form of stand-alone memory is insufficient to solve the deblurring task. The quantitative measures in Table 5 support this finding. The results illustrate that for video deblurring, the deep network used to extract deep low-level features is essential.

We also experimented with a memory variant, which always stores the previous adjacent blurry-sharp feature pairs as a temporary item in the memory bank [16] (*'Temp'* in Table 5). This approach increases the complexity but hurts the performance. This is because the adjacent frames are similar and contain fine details for sharpening the current frame. Saving them in the memory space is likely to compress out these details and the corresponding attention weights will be distracted by other memories, as the attention is calculated on the entire memory bank. This motivated us to design a more straightforward approach to directly concatenate the previously restored sharp results with the current frame and

(a) Input (b) DBN [22] (c) ESTRNN [35] (d) CDVD-TSP [17] (e) Ours (f) Ground-truth

Figure 4. Qualitative comparisons on the original GOPRO dataset [14].

(a) Input (b) DBN [22] (c) CDVD-TSP [17] (d) ESTRNN [35] (e) Ours (f) Ground-truth

Figure 5. Qualitative comparisons of frames with severe blur on the original GOPRO dataset [14].

Table 5. Experimental results on high-level designs of our models. The 'w/o Mem.' and 'w/o Deblur.' columns show the results without memory branch and deblurring branch, respectively; the 'Temp.' column shows the results using the temporary memory for the previous adjacent frame.

Design	w/o Mem.	w/o Deblur.	Temp.	Ours
PSNR	30.88	30.58	31.03	**31.22**
SSIM	0.9145	0.9105	0.9169	**0.9203**

feed them into the network. Since the motion between adjacent frames is generally small, our model can capture the correlation well. This approach achieves the best result as shown in Table 5.

Separate h and r memories. Previous methods using memory networks work well when encoding only the target output, *e.g.* only the segmentation map in object segmentation [2, 16]. In our case, this would correspond to memorizing only the deblurred results r. However, given the importance of the hidden feature h in recurrent methods [15, 35], it is also logical to memorize h. Table 6 shows that of these

Table 6. Comparison of different variants of utilizing features h and r. $[h, r]$ means we concatenate two features and save them as a single memory value.

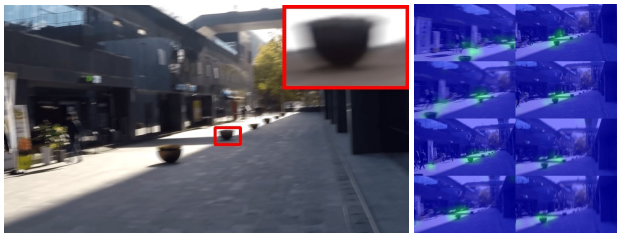
Variant	h only	r only	$[h, r]$	Ours
PSNR	31.03	30.94	31.00	**31.22**
SSIM	0.9178	0.9152	0.9174	**0.9203**

two, memorizing h is more beneficial than r. It is also possible to concatenate the two ('$[h, r]$') and save them as a single memory value, but this gives only a marginal bump in performance compared to only memorizing r. Storing the two in separate memories under the same key gives a much bigger boost.

Memorization period. As mentioned in Section 3.2, we memorize every T frames. As we adopt the multi-scale strategy, for scale levels larger than 1, the input resolution is small, so decreasing T does not result in a large amount of computation. However, for the scale level $s=1$, decreasing the memory period T_1 results in a significant increase in the computational and memory cost. We performed the

(a) Query frame and pattern (b) Matched frame (c) Attention map (d) Matched pattern (e) Matched deblurred result

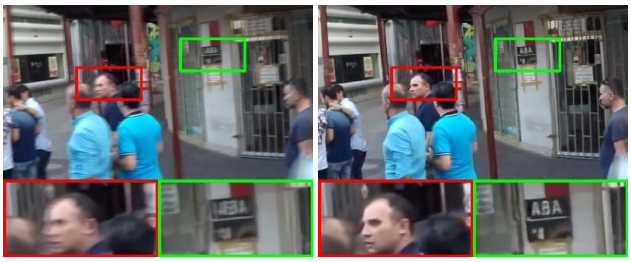
Figure 6. Visualization of spatio-temporal attention map obtained by our memory branch.



(a) Query frame and pattern (b) $s = 3$

(c) $s = 1$ (d) $s = 2$

Figure 7. Visualization of attention map with multi-scale architecture. The same patterns that reoccur but are not of the same scale are matched. Note that all these matched memories contribute to the deblurring task for the input. Zoom in for a better view.



(a) Naive memory network (b) Our memory solution

Figure 8. Qualitative comparison between naive memory approach and our memory network.

experiment on scale level of $s = 1$ to evaluate the effect of memorization period. Table 7 shows that even though memorizing every frame performs the best, its runtime is expensive. Therefore, to balance the trade-off between efficiency and performance, we memorize every 5 frames at scale level $s = 1$. Similarly, for $s = 2$ and $s = 3$, we empirically memorize every 2 and 1 frame, respectively.

Visualizations. To get a better sense of what the memory retrieves for a query pattern, we calculated the attention map encoded in $\mathbf{W}$ in Eq. 6 on the memory bank. The visualized result is shown in Fig. 6, where we select one of the matched frames for reference. It can be observed that since we need to deblur the license plate number, most of the attention is given to the license plate number in the matched frame. The attention maps for the multi-scale memory are shown in Fig. 7. We observe that the memory bank provides different scales of matching. This further enhances the query and utilization of features.

Table 7. Memorization period comparison. The PSNR is calculated on the downsampled GOPRO dataset.

T	PSNR	SSIM	GMACs	Runtime (s)
1	**31.42**	**0.9231**	218.79	0.582
3	31.21	0.9202	201.10	0.154
5	31.22	0.9203	**197.34**	**0.095**

Limitations. The size of the memory in the memory bank affects the GPU memory overhead and computational cost. In Section 4.1, we consider a simple way of reducing the memory size by discarding old memories under the assumption that the most recent memories are more important. We find this approach effective, but we believe that more principled memory management strategies can be considered for future work.

6. Conclusion

We proposed a multi-scale memory-based network for deep video deblurring, which saves blurry-sharp feature pairs in the memory bank. To restore a blurry input frame, we retrieve relevant information for each image region by performing the spatio-temporal attention in the memory bank. The memory bank allows region-aware information retrieval to achieve fine-grained deblurring. To enrich the diversity and utility of the memory bank, we developed a bidirectional and multi-scale strategy. Both quantitative and qualitative experimental results show that our proposal outperforms the state-of-the-art models while maintaining a minimal cost in computational complexity and runtime.

7. Acknowledgement

This research is supported by the National Research Foundation, Singapore under its NRF Fellowship for AI (NRF-NRFFAI1-2019-0001). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 168–172. IEEE, 1994. 5

[2] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. *arXiv preprint arXiv:2106.05210*, 2021. 1, 2, 7

[3] Jochen Gast and Stefan Roth. Deep video deblurring: The devil is in the details. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[4] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1712–1722, 2019. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[6] Tae Hyun Kim, Kyoung Mu Lee, Bernhard Scholkopf, and Michael Hirsch. Online video deblurring via dynamic temporal blending network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4038–4047, 2017. 1, 2, 6

[7] Yoonsik Kim, Jae Woong Soh, Gu Yong Park, and Nam Ik Cho. Transfer learning from synthetic to real-noise denoising with adaptive instance normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3482–3492, 2020. 2

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[9] Hee Seok Lee, Junghyun Kwon, and Kyoung Mu Lee. Simultaneous localization, mapping and deblurring. In *2011 International Conference on Computer Vision*, pages 1203–1210. IEEE, 2011. 2

[10] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 2, 5

[11] Xiankai Lu, Wenguan Wang, Martin Danelljan, Tianfei Zhou, Jianbing Shen, and Luc Van Gool. Video object segmentation with episodic graph memory networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 661–679. Springer, 2020. 2

[12] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016. 1, 2

[13] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. A read-write memory network for movie story understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 677–685, 2017. 2

[14] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3883–3891, 2017. 2, 5, 6, 7

[15] Seungjun Nah, Sanghyun Son, and Kyoung Mu Lee. Recurrent neural networks with intra-frame iterations for video deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8102–8111, 2019. 1, 2, 5, 6, 7

[16] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9226–9235, 2019. 1, 2, 3, 6, 7

[17] Jinshan Pan, Haoran Bai, and Jinhui Tang. Cascaded deep video deblurring using temporal sharpness prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3043–3051, 2020. 1, 2, 5, 6, 7

[18] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6626–6634, 2018. 2

[19] Hee Seok Lee and Kuoung Mu Lee. Dense 3d reconstruction from severely blurred images using a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 273–280, 2013. 2

[20] Hongje Seong, Junhyuk Hyun, and Euntai Kim. Kernelized memory network for video object segmentation. In *European Conference on Computer Vision*, pages 629–645. Springer, 2020. 2

[21] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 5

[22] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1279–1288, 2017. 1, 2, 6, 7

[23] Maitreya Suin and AN Rajagopalan. Gated spatio-temporal attention-guided video deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7802–7811, 2021. 1, 2

[24] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015. 1, 2, 6

[25] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8174–8182, 2018. 2, 5, 6

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia

Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3

[27] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1, 2, 6

[28] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[29] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014. 2, 6

[30] Patrick Wieschollek, Michael Hirsch, Bernhard Scholkopf, and Hendrik Lensch. Learning blind motion deblurring. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 231–240, 2017. 1, 2

[31] Yi Wu, Haibin Ling, Jingyi Yu, Feng Li, Xue Mei, and Erkang Cheng. Blurred target tracking by blur-driven tracker. In *2011 International Conference on Computer Vision*, pages 1100–1107. IEEE, 2011. 2

[32] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 152–167, 2018. 2

[33] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018. 2

[34] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018. 5

[35] Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020. 1, 2, 5, 6, 7

[36] Shangchen Zhou, Jiawei Zhang, Jinshan Pan, Haozhe Xie, Wangmeng Zuo, and Jimmy Ren. Spatio-temporal filter adaptive network for video deblurring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2482–2491, 2019. 1, 2, 6