Real-time Streaming Video Denoising with Bidirectional Buffers

Chenyang Qi*
HKUST

Junming Chen*
HKUST

Xin Yang
HKUST

Qifeng Chen
HKUST

ABSTRACT

Video streams are delivered continuously to save the cost of storage and device memory. Real-time denoising algorithms are typically adopted on the user device to remove the noise involved during the shooting and transmission of video streams. However, sliding-window-based methods feed multiple input frames for a single output and lack computation efficiency. Recent multi-output inference works propagate the bidirectional temporal feature with a parallel or recurrent framework, which either suffers from performance drops on the temporal edges of clips or can not achieve online inference. In this paper, we propose a Bidirectional Streaming Video Denoising (BSVD) framework, to achieve high-fidelity real-time denoising for streaming videos with both past and future temporal receptive fields. The bidirectional temporal fusion for online inference is considered not applicable in the MoViNet. However, we introduce a novel Bidirectional Buffer Block as the core module of our BSVD, which makes it possible during our pipeline-style inference. In addition, our method is concise and flexible to be utilized in both non-blind and blind video denoising. We compare our model with various state-of-the-art video denoising models qualitatively and quantitatively on synthetic and real noise. Our method outperforms previous methods in terms of restoration fidelity and runtime. Our source code is publicly available at https://github.com/ChenyangQiQi/BSVD

CCS CONCEPTS

• **Computing methodologies** → **Computational photography**.

KEYWORDS

Video Denoising, Efficient Inference, Online Inference.

ACM Reference Format:
Chenyang Qi, Junming Chen, Xin Yang, and Qifeng Chen. 2022. Real-time Streaming Video Denoising with Bidirectional Buffers. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3547934

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '22, October 10–14, 2022, Lisboa, Portugal
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3547934

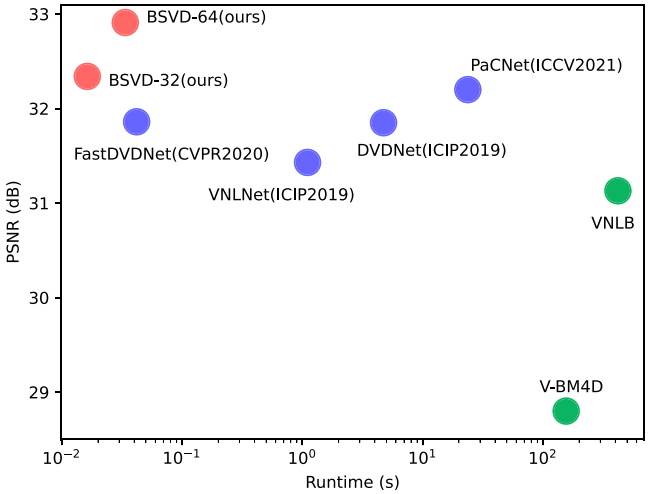


Figure 1: Comparison of PSNR and runtime on DAVIS test set with noise level $\sigma = 50$. Our BSVD-64 outperforms state-of-the-art PaCNet [23] with 700× speedup. It takes BSVD-64 33.7ms per frame to process a video with resolution of 960×540. Green: CPU method. Blue: Learning-based GPU method. Red: Ours.

1 INTRODUCTION

With the explosive growth of social media, there is an increasing need for video streaming applications on the user's device, such as live streaming on YouTube, TikTok, and video meetings on Zoom. Since the noise from video capturing and compression degrades the quality of videos, noise reduction plays a vital role in improving video fidelity. Although offline video processing [3, 9, 21, 23] has been an active research topic in recent years, few works [14, 22, 26, 27] are feasible for online streaming video denoising, which is more challenging. Streaming video denoising is featured in the following two points compared with offline video denoising. First, since streaming video denoising is primarily applied to sports games and other live streaming media, a real-time inference speed should be satisfied. Second, the frames should be processed online in a continuous frame-by-frame way because there is no explicit end for streaming video. To fulfill the above requirements, we propose a real-time high-quality streaming denoising framework.

To the best of our knowledge, the existing works can be categorized into three classes: sliding-window-based methods [22, 23] (Fig. 2(a)), recurrent methods [14, 30] (Fig. 2(b,c)) and multi-input multi-output (MIMO) methods [9, 30] (Fig. 2(d)). The sliding-window-based methods [22, 23] (Figure 2(a)) restore each frame by feeding the degraded one with its neighbours. However, such sliding-window strategy has unnecessary, redundant computation:

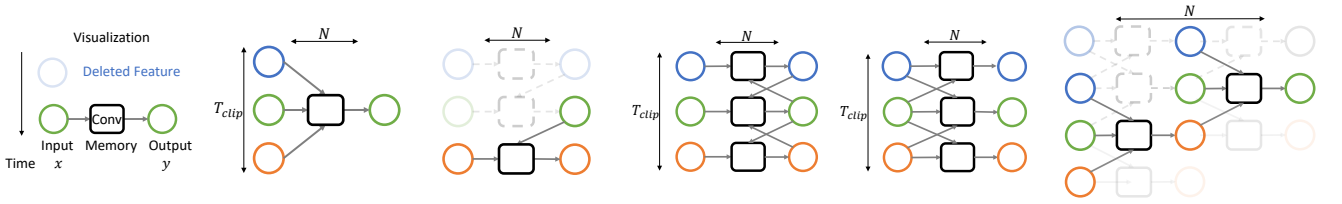
Method	(a) Sliding-window	(b) Uni-RNN	(c) Bi-RNN	(d) MIMO	(e) Ours
					
Input	$x_{i-T_{clip}/2}, \ldots, x_{i+T_{clip}/2}$	x_i	$x_1, x_2, \ldots, x_{T_{clip}}$	$x_1, x_2, \ldots, x_{T_{clip}}$	x_i
Buffered Feature Index	-	$i-1$	$1, 2, \ldots, T_{clip}$	-	$i-N, \ldots, i-1$
Output	y_i	y_i	$y_1, y_2, \ldots y_{T_{clip}}$	$y_1, y_2, \ldots y_{T_{clip}}$	y_{i-N}
Memory Complexity	$O(T_{clip})$	$O(N)$	$O(T_{clip})$	$O(T_{clip})$	$O(N)$
Time Per Frame	$O(T_{clip}N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
Bidirectional Fusion	✓	✗	✓	✓	✓
Clip-edge Fidelity	✓	✓	✗	✗	✓

Figure 2: The comparison of computation graph and complexity for different methods. For an input clip with length T_{clip}, we assume all methods use same N convolution blocks for temporal fusion. Blue, green, and red features represent the past, present, and future features from three adjacent frames. Solid features are cached in the GPU memory, while dotted features have been deleted. Compared with sliding-window methods (a), our inference time is shorter. For unidirectional-RNN (b), our framework utilize bidirectional temporal fusion, and achieve better fidelity. In addition, bidirectional-RNN (c) and MIMO (d) framework suffer from $O(T_{clip})$ memory and fidelity degradation on the clip edges, which is solved in our inference framework.

each frame is fed into the network multiple times. For instance, to restore a video with resolution of 960×540, it takes PaCNet [23] about 24 seconds per frame, which impedes this method to real-time application. Meanwhile, such methods only fuse the temporal information in pixel-level [22] or neighbor patch-level [23], while the intermediate multi-scale features are also important for the denoising performance, as shown in (Sec. 5.2).

Different from sliding-window-based methods, recurrent works [3, 14] use previous reconstructed results as a reference for next restoration. Such recurrent methods can be further divided into unidirectional [14, 30] (Fig. 2(b)) and bidirectional [3] (Fig. 2(c)) propagation. Although unidirectional recurrent methods can exploit past information for streaming processing, their denoising performance degrades by ignoring future information. To utilize both past and future information, bidirectional recurrent methods are proposed to propagate the temporal information in both forward and backward directions. For best results from a bidirectional recurrent network, users must prepare the entire video in advance, which means they cannot realize streaming video denoising [3].

Most recently, MIMO frameworks [9, 30] have been introduced for efficient processing a clip of video frames in one-forward-pass(Fig. 2(d)). The inference schema can be implemented as a channel shift [12] or a local window [9] in temporal dimension. However, with the growth of clip size T_{clip}, the inference-time memory consumption also increases linearly in $O\left(T_{clip}\right)$. Thus, such methods suffer from heavy memory consumption [9], and the long input video must be segmented into short clips. Moreover, similar to bidirectional recurrent methods [3], MIMO also has performance degradation on the edges of clips (Sec. 5.1).

In this work, we propose the **B**idirectional **S**treaming **V**ideo **D**enoising framework (BSVD). Bidirectional temporal fusion is critical for low-level video processing since it fully utilizes the information of both past and future frames, which is demonstrated in our experiment (Sec. 5.2). Thus, we train our model as a MIMO framework using bidirectional Temporal Shift Modules (TSM) [12], which is also more efficient than sliding-window. To address the clip-edges-drop problem [3, 9] and enable the bidirectional fusion for streaming videos, which is regarded as not applicable in previous works [7, 12], we propose a Bidirectional Buffer Block that can cache and reuse the features for both past and future propagation during pipeline-style inference. Thus, our BSVD featured with the pipeline-style buffer block can achieve a constant memory complexity of $O\left(1\right)$, in contrast with $O\left(T_{clip}\right)$ in other MIMO methods [3, 9]. Extensive evaluations on datasets with synthetic and real noise show that the proposed method outperforms previous methods in denoising performance and runtime.

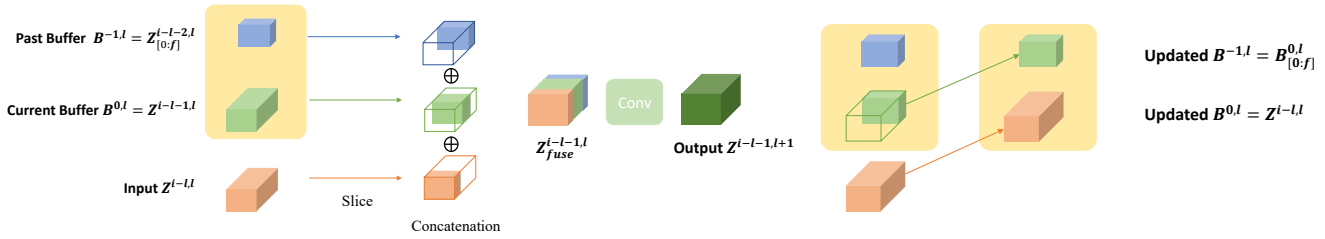
Our contributions can be summarized as follows:

(a) We propose a pipeline-style buffer-based video denoising framework, named BSVD, which processes the video streams of resolution 960×540 in real-time.

(b) The proposed novel Bidirectional Buffer Block and pipeline-style inference framework enable bidirectional temporal fusion for online streaming video processing.

(c) In addition, we address the fidelity degradation on the clip edges in the MIMO framework [3] by utilizing buffered features. Our proposed computation graph is general and applicable for pre-trained checkpoints of the existing method FastDVDNet [22].

(d) Our succinct BSVD framework achieves SOTA performance on Gaussian noise [22, 23], real noise [14], and blind denoising [20, 30]. Figure 1 provides a comparison of PSNR and runtime on DAVIS test set with noise level $\sigma = 50$. BSVD outperforms the previous most efficient baseline FastDVDnet [22] with more than 2× speedup and improves average PSNR by 0.57dB on DAVIS dataset. Moreover, it surpasses the best fidelity work PaCNet [23] while being 700× faster.



(a) Temporal Fusion (b) 2D Convolution (c) Update Buffer

Figure 3: The forward operation of l^{th} Bidirectional Buffer Block at time step i. This block aggregates input feature $Z^{i-l,l}$ with buffered features $B^{-1,l}, B^{0,l}$, to output $Z^{i-l-1,l+1}$, which is the input for the $(l+1)^{\text{th}}$ temporal buffer block. After convolution operation, the buffered feature are updated using input $Z^{i-l,l}$.

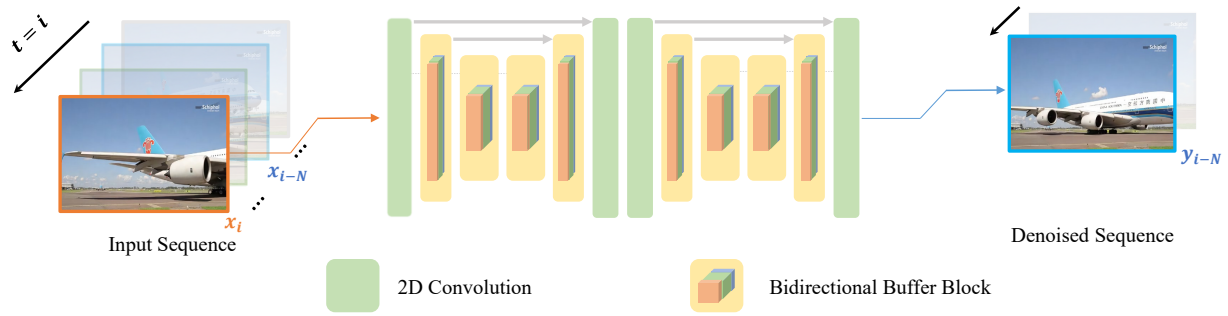


Figure 4: An overview of our framework. The backbone of our network is two light-weight U-Nets [19] with temporal fusion operation inserted between convolution layers. At time step i during inference, one noisy frame x_i and its noise map are fed into the neural network. Then, our network outputs another clean frame y_{i-N}.

2 RELATED WORK

Image denoising. Single image denoising has been a long-standing problem as one of the most fundamental image restoration tasks. Some traditional and representative methods [2, 4, 8, 15] typically exploit the texture similarity in non-local image patches to achieve satisfying denoising performance. However, such methods suffer from time-consuming patch searching. Recently, deep learning has made great progress for single image denoising tasks. Zhang et al. [34] propose a deep CNN using residual learning, which outperforms the traditional methods by a large margin. They present an efficient model using downsampled sub-images and a non-uniform noise level map. Meanwhile, some other works attempt to predict pixel-wise kernels [17, 29, 31] firstly and then apply these spatial-variant kernels as convolutional weights in the main denoising network branch. Recently, Transformer-based architectures [10, 33] achieve state-of-the-art performance on single image denoising tasks. While single-image methods can restore each frame online, they also discard the information in adjacent frames, which should be considered in the streaming denoising task.

Video denoising. In the video, pixels in adjacent frames can be very similar. The temporal correlation can be utilized by deformable convolution [24], kernel prediction [17, 29], block matching [13, 23], optical flow [21] and attention [9] which typically improve the image fidelity at the cost of expensive computation. Meanwhile, various works focus on the reduction of computation in video denoising [14, 22]. FastDVDnet [22] is composed of two lightweight U-Nets [19] without explicit motion estimation. It works in a multiple-to-single manner, while ours adopt an efficient buffer-based inference pipeline without redundancy. More recently, EMVD [14] presents a unidirectional recurrent solution, which reuses the previous predicted clean frame. Compared with EMVD [14], our buffer-based bidirectional fusion fully utilized the feature from neighboring frames. Meanwhile, we solve the problem of high memory consumption and performance drop on the temporal edges of clips, which widely exist in MIMO frameworks [3, 9, 30].

3 METHOD

Given a stream of noisy frames without an explicit ending frame $\{x_0, x_1, \ldots, x_i, \ldots\}$, where $x_i \in \mathbb{R}^{C\times H\times W}$ with input channel C, height H and width W, we restore the clean frames $\{y_0, y_1, \ldots, y_i, \ldots\}$ in a continuous pipeline. Our method is suitable for both non-blind and blind denoising. For non-blind denoising, we set $C = 4$ [22, 23] or $C = 5$ [14] in input frames following previous works. For blind denoising, we only input the RGB channels and thus $C = 3$.

We train our model in MIMO framework using Temporal Shift Module(TSM) [12] for distributed parallelism (Sec. 3.1). During inference, we replace TSM with Bidirectional Buffer Block (Sec. 3.2), and our framework processes the video in a pipeline (Sec. 3.3). Figure 2 demonstrates the difference between our method and previous inference frameworks. Figure 4 gives an overview of our inference algorithm with Bidirectional Buffer Block (Figure 3). Specifically, at

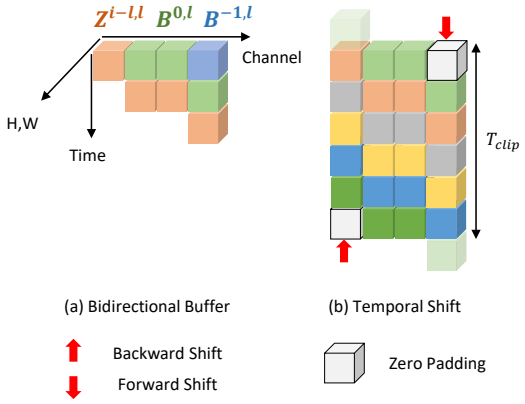


Figure 5: Illustration of the difference between a single block of our pipeline-style inference method and TSM. We set the ratio $r = 4$ in visualization. TSM uses zero paddings to fill the slots produced by shift operation, while we use buffered feature without zero padding at the temporal boundaries.

time step $i \in \{0, 1, 2, \ldots\}$, we feed a noisy frame x_i. Then, the input is fused with previous features buffered in N Bidirectional Buffer Blocks. Finally, the model produces a single clean frame y_{i-N} for the noisy input at time step $i - N$ at the end of the time step i.

3.1 Training with Temporal Shift Module

During the training stage, we utilize TSM [12] as the temporal fusion operation in our convolution backbone. We randomly crop T_{clip}-frame clips from videos as input and use the pixel loss (e.g., L1 or L2) between $\{x_t\}_{t=1}^{T_{clip}}$ and $\{y_t\}_{t=1}^{T_{clip}}$ as the training objective. Assume a feature clip $Z \in \mathbb{R}^{T_{clip} \times C_f \times H_f \times W_f}$ is the stack of T_{clip} continuous frame features, with feature channel C_f, height H_f and width W_f. Then, the temporal shift can be represented as the concatenation ($\oplus$) of features channels:

$$Z^0_{fused} = \quad Z^{-1}_{[0:f]} \oplus Z^0_{[f:-f]} \oplus Z^{+1}_{[-f:]}, \tag{1}$$

where superscript -1 and $+1$ represent one-frame forward and backward shift, respectively. $f = \lfloor C_f / r \rfloor$ is the number of shifted channels, and the ratio of channels shifted in each direction is empirically set as $r = 8$ following TSM [12]. As shown in Figure 5(b), TSM is a MIMO framework, which generates blank slots and humps on the boundaries of the clip. The original TSM [12] directly truncates the humps and fills the blank slots with zero paddings, which degrades the image fidelity. Thus, it is not an optimal inference framework for processing video streams.

Backbone network. We utilize two enhanced lightweight U-Nets [19, 28] as our base model W-Net [28]. Such a two-step denoising architecture has proven to be effective in previous works [20, 22, 28]. Batch normalization (BN) layer is observed to decrease image fidelity in super-resolution [11, 25] and deblurring [18], but it still exists in current SOTA video denoising methods [22, 23]. Thus, we remove BN layers in the W-Net. Besides, we replace ReLU activation with ReLU6 to alleviate artifacts during FP16 inference. In the following, "BSVD-32" denotes our model whose smallest feature channel is "32". We adjust our backbone by multiplying or dividing all channel numbers by a scale. Please refer to the supplement for more details about our implementation.

Algorithm 1 Pseudo code of a single Bidirectional Buffer Block.

```
class BidirectionBufferedBlock(nn.Module):
    def __init__(self, in_channels, out_channels, r=8):
        self.conv = nn.Conv2d(in_channels, out_channels)
        self.f = in_channels//r
        self.B^{-1,l} = torch.zeros(n, self.f, h, w)
        self.B^{0,l} = torch.zeros(n, in_channels, h, w)
    def forward(self, Z^{i-l,l}):
        # Activate the block
        if self.B^{0,l} is None:
            self.B^{0,l} = Z^{i-l,l}
            return None
        # Fuse the temporal adjacent features
        fusion = torch.cat([
            self.B^{-1,l},
            self.B^{0,l}[:,self.f:-self.f, :, :]
            Z^{i-l,l}[:, -self.f:, :, :], dim=1)
        # 2D Convolution for output
        Z^{i-l-1,l+1} = self.conv(fusion)
        # Update the buffer
        self.B^{-1,l} = self.B^{0,l}[:, :self.f, :, :]
        self.B^{0,l} = Z^{i-l,l}
        return Z^{i-l-1,l+1}
```

3.2 Inference with Bidirectional Buffer Block

During the inference stage, we replace the temporal shift operation with the designed Bidirectional Buffer Block, while keeping the weights of the convolution backbone unchanged. As shown in Figure 5(a), our buffer blocks serve as a bridge to transfer the feature of neighboring frames, which can enlarge the temporal receptive field as well as make bidirectional temporal fusion possible for streaming video, which is regarded not applicable in previous works [7, 12]. Therefore, we solve the fidelity degradation at two clip boundaries in such a MIMO system.

The pseudo code shown in Alg. 1 and Figure 3 delineate the computation steps of a single Bidirectional Propagation Block. Each block is initialized with two buffers, which will be filled with intermediate features. For l^{th} block, we denote them as current buffer $B^{0,l} \in \mathbb{R}^{C_f \times H_f \times W_f}$ and past buffer $B^{-1,l} \in \mathbb{R}^{f \times H_f \times W_f}$. We denote the relative temporal index as the superscript $0, -1$.

During the forward inference at time step $i \geq N$, a future feature $Z^{i-l,l} \in \mathbb{R}^{C_f \times H_f \times W_f}$ is fed into l^{th} buffered block. The superscript index $i - l$ denotes latency of l frames caused by l blocks. More details about latency and the operation for $i < N$ at will be discussed later in Sec.3.3. As shown in Figure 3, The forward operation is composed of three steps:

(a) Fusion of new input and buffered features. The temporal fusion can be represented as

$$Z^{i-l-1,l}_{fused} = \quad B^{-1,l} \oplus B^{0,l}_{[f:-f]} \oplus Z^{i-l,l}_{[-f:]}, \tag{2}$$

where two buffers are filled with features $B^{-1,l} = Z^{i-l-2,l}_{[0:f]}$, $B^{0,l} = Z^{i-l-1,l}$, which are cached in the last time step $i-1$. We will discuss the features in buffers in (c).

(b) Feature extraction. The feature extraction operation can be any general 2d operator, which is implemented as 2d convolution:

$$Z^{i-l-1,l+1} = \quad Conv(Z^{i-l-1,l}_{fused}), \tag{3}$$

Table 1: Pipeline-Style inference at different time step.

Time step	Input	Buffered features	Output
0	x_0		None
1	x_1	$Z^{0,1}$	None
2	x_2	$Z^{1,1}, Z^{0,2}$	None
...			
N	x_N	$Z^{N-1,1}, Z^{N-2,2} \ldots, Z^{1,N-1}$	$Z^{0,N} = y_0$
$i \geq N$	x_i	$Z^{i-1,1}, Z^{i-2,2}, \ldots, Z^{i-N+1,N-1}$	$Z^{i-N,N} = y_{i-N}$

where $Z^{i-l-1,l+1}$ is the input for the next $(l+1)^{\text{th}}$ Bidirectional Buffer Block.

(c) Update the buffered feature in the memory. At the end of timestep i for l^{th} block, we update the intermediate feature in the fixed buffer to reuse the previous computation.

$$B^{-1,l} = B^{0,l}_{[0:f]} \tag{4}$$

$$B^{0,l} = Z^{i-l,l} \tag{5}$$

After the above three steps, the temporal index of output ($i-l-1$ for $Z^{i-l-1,l+1}$) has moved backward by 1 frame compared with the original input ($i-l$ for $Z^{i-l,l}$), since most of $Z^{i-l-1,l}_{fused}$'s channels are from $B^{0,l} = Z^{i-l-1,l}$.

3.3 Pipeline-style Inference

A single Bidirectional Buffer Block can be seen as a sliding window with size 3, which has 1 frame temporal latency. We consider the first Bidirectional Buffer Block for ease of description. At time step $i = 0$, both buffers ($B^{-1,1}$ and $B^{0,1}$) in 1^{st} block are empty, which means this block does not have enough information for temporal fusion. Thus, it caches the input $Z^{0,1}$ to $B^{0,1}$, and the whole pipeline exits. At time step $i = 1$, together with the new input, the block is activated by information from two time steps. It fills $B^{-1,l}$ with zeros and conduct forward temporal fusion to extract feature $Z^{0,1}$ for x_0. Table 1 shows the operation and intermediate feature at each time step. For time step $i < N$, each input frame activates one deeper Bidirectional Buffer Block $B^{0,i}$ in our network. Therefore, when the first denoised frame y_0 comes out, there are already $N+1$ frames fed into our inference block. At a general time step $i \geq N$, the network takes x_i as input. l^{th} block takes $Z^{i-l,l}$ as input and produce $Z^{i-l-1,l+1}$. The whole pipeline generates and caches N features. Finally, the network produces a clean frame output y_{i-N} at the end of the time step i. For the last N frames in the entire video sequence, we feed dummy zero tensors into the pipeline to get the clean output of the last few frames. Our supplement provides more details about the inference pipeline.

3.4 Analysis

Running time. FastDVDnet [22] is a sliding-window-based framework with a window size of 5, which is composed of 2 U-Nets with the smallest channel 32. FastDVDnet fuses the three adjacent noisy frames at each U-Net's input layer. Thus, the computation cost for each output is 3 + 1 = 4 times that of a single U-Net. In our work, BSVD-32 uses two similar U-Nets with the same channel setting as the convolution backbone. Instead of feeding a clip of frames, we feed in a single frame at any time step. Therefore, the computation cost of our method for each frame is 1 + 1 = 2 times of U-Net. As shown in Table 2, our BSVD-32 reduces more than 50% of runtime compared with FastDVDnet.

Memory. In bidirectional-RNN and MIMO frameworks [3, 9, 30], memory consumption is typically proportional to the clip length T_{clip} at inference time, which impedes the processing of the long video on devices with limited resources. Thus, the whole video is divided into multiple short sequences to fit the memory size on the user device. However, for low-level denoising tasks, this will lead to a quality drop on the boundary frames in a video clip due to the loss of the marginal feature (Sec. 5.1). Unlike MIMO frameworks, our buffers work in a pipeline that consumes frames one by one. Therefore, our method can keep a constant run-time memory, which depends only on the number of Bidirectional Buffer Blocks N.

Temporal receptive field. The temporal receptive field of a single Bidirectional Propagation Block is 3. During the inference of the neural network, the receptive field of temporal shift will be accumulated like 1-D convolution. In our framework, we apply $N = 16$ layers of Bidirectional Propagation Block in the network. Therefore, the accumulated temporal receptive field of our BSVD is 33, which is larger than previous window-based methods (e.g., 7 for PaCNet [23] and 5 for FastDVDnet [22]) by a large margin.

4 EXPERIMENTS

4.1 Datasets and Settings

To evaluate our method, we use both RGB images with synthetic noise [6] and raw images with real-world noise [32].

DAVIS and Set8 datasets. For RGB images with synthetic noise, we follow the data preparation of FastDVDnet [22]. The clean patches are randomly sampled from the training set of the DAVIS [6]. The noisy patches are generated by adding additional white Gaussian noise (AWGN) of $\sigma \in [5, 50]$ to clean patches. For models trained on the DAVIS train set, we evaluate them on both Set8 and DAVIS test sets [22]. We follow the FastDVDnet to limit all test video sequences to the first 85 frames.

CRVD dataset. For real-world raw images, we use the dataset collected by Yue et al. [32]. It is composed of one synthetic dataset (SRVD) generated from MOT [16] and one captured dataset (CRVD). The real raw noise is often assumed to conform to a heteroskedastic Gaussian distribution with signal-dependent variance:

$$\sigma^2(\mathbf{x}) = a\mathbf{x} + b, \tag{6}$$

where $a, b \in \mathbb{R}$ are parameters for shot and read noise respectively [14]. 5 possible pairs of (a, b) map to 5 different ISO settings. Following previous works [14, 32], we train our model with SRVD and CRVD scenes 1 to 6, and test on CRVD scenes 7 to 11.

4.2 Baselines

We compare our method with state-of-the-art methods, including raw data baselines [14, 22, 32], and RGB image baselines [1, 13, 20–23, 30]. For evaluation on RGB images with Gaussian noise, we train our model with the same L2 loss as FastDVDnet [22], and compare it with the quantitative results of [1, 13, 21, 22] in the FastDVDnet paper. For experiments on raw images with real-world noise, we

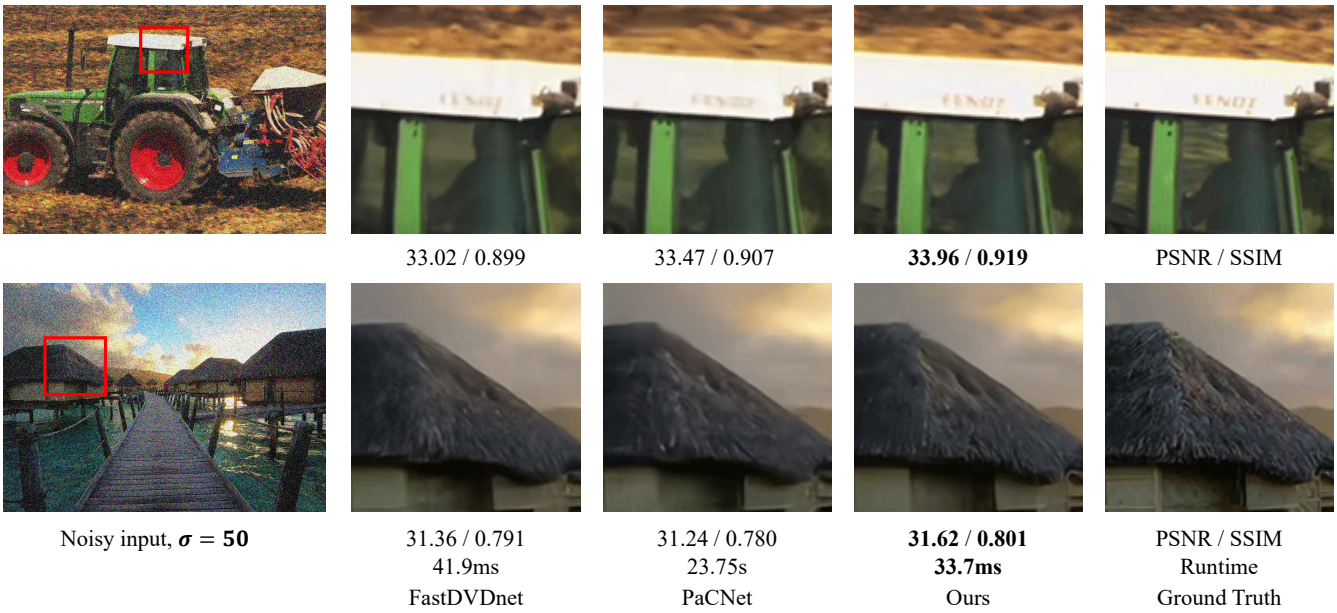


Figure 6: Qualitative comparison on Set8 dataset. Ours reconstructs more high-frequency details in shorter running time.

Table 2: Quantitative comparisons of PSNR (dB) and runtime on the test set of DAVIS and Set8. C and G represent CPU and GPU time cost, respectively. 10, 20, 30, 40, 50 **represents the σ of test data. We show the averaged inference time per frame with the resolution of 960 × 540 in PyTorch framework with FP16 precision. Our method outperforms previous methods on average PSNR and runtime.**

		DAVIS						Set8					
Model	Runtime(s)	10	20	30	40	50	Average	10	20	30	40	50	Average
VNLB [1]	420.0 (C)	38.85	35.68	33.73	32.32	31.13	34.34	**37.26**	33.72	31.74	30.39	29.24	32.47
V-BM4D [13]	156.0 (C)	37.58	33.88	31.65	30.05	28.80	32.39	36.05	32.19	30.00	28.48	27.33	30.81
VNLnet [5]	1.104 (G)	35.83	34.49	-	32.32	31.43	-	37.10	33.88	-	30.55	29.47	-
DVDnet [21]	4.731 (C+G)	38.13	35.70	34.08	32.86	31.85	34.52	36.08	33.49	31.79	30.55	29.56	32.29
FastDVDnet [22]	0.0419 (G)	38.71	35.77	34.04	32.82	31.86	34.64	36.44	33.43	31.68	30.46	29.53	32.31
PaCNet [23]	23.75 (G)	**39.97**	**36.82**	34.79	33.34	32.20	35.42	37.06	**33.94**	32.05	30.70	29.66	32.68
BSVD-32 (ours)	0.0163 (G)	39.47	36.36	34.57	33.31	32.34	35.21	36.49	33.56	31.85	30.65	29.74	32.46
BSVD-64 (ours)	**0.0337** (G)	39.81	**36.82**	**35.09**	**33.86**	**32.91**	**35.70**	36.74	33.83	**32.14**	**30.97**	**30.06**	**32.75**

use the evaluation metric from RViDeNet [32] and train with L1 loss, following EMVD [14]. In Table 4, we quantitatively compare with the results of all baselines from the EMVD [14] paper. Since the code of EMVD is not open-sourced, we test the efficiency of EMVD using an unofficial implementation[1].

4.3 Results

Non-blind denoising on RGB images. Table 2 shows the quantitative comparison of our method with baselines on the DAVIS and Set8 datasets. The inference time of VNLB, V-BM4D is obtained from the FastDVDnet paper. These CPU-based baselines are extremely slower than GPU-based methods. For fairness, we test the inference time of the most related baselines [5, 21–23] and our method in PyTorch framwork on the same NVIDIA RTX3090 GPU with FP16 precision.

The results in Table 2 indicates that our model surpasses all previous methods in terms of overall denoising performance with a much lower runtime. Compared with the best fidelity baseline PaCNet [23], our BSVD-64 achieve a 0.28dB improvement in average PSNR on the DAVIS dataset and 0.13dB on Set8 with 700× speedup during inference. In addition, our GPU memory cost (3.09GB) is much lower than the cost of PaCNet (10.58GB). Specifically, our method has significant advantages at $\sigma = 50$ (e.g., 0.71dB improvement on DAVIS, 0.4dB on Set8). PaCNet exploits explicit search and fusion of nearest neighbor patches in the RGB pixel domain, which consumes an extremely long runtime and lacks matching of deep features. In contrast, our method implicitly conducts the alignment and fusion through convolution on multi-scale features, which proves to be more efficient with better fidelity. Figure 6 also demonstrates that our method restores more high-frequency details on the characters and the roof. Our BSVD-64 processes the video

[1]https://github.com/Baymax-chen/EMVD

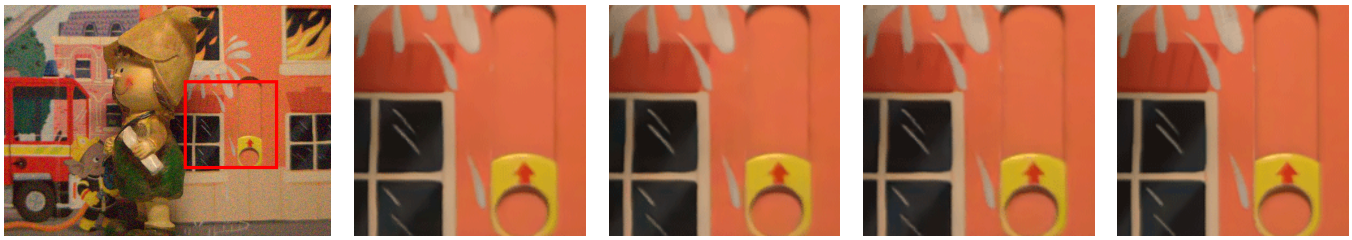


Figure 7: Qualitative Comparison of real noisy frames from CRVD dataset.

Table 3: Quantitative comparisons of PSNR for blind denoising.

Model	Runtime(s)	DAVIS 10	20	30	40	50	Average	Set8 10	20	30	40	50	Average
ReMoNet [30]	-	38.97	35.77	33.93	32.64	31.65	34.59	36.29	33.34	31.59	30.37	29.44	32.21
UDVD [20]	0.9561	-	34.99	33.86	32.61	31.63	-	-	33.25	31.86	30.62	29.63	-
BSVD-32-blind (ours)	**0.0162**	39.31	36.23	34.46	33.21	32.25	35.09	36.23	33.40	31.71	30.53	29.62	32.30
BSVD-64-blind (ours)	0.0335	**39.68**	**36.66**	**34.91**	**33.68**	**32.72**	**35.53**	**36.54**	**33.70**	**32.02**	**30.85**	**29.95**	**32.61**

Table 4: Quantitative comparisons of PSNR and SSIM on the CRVD test set. We test the averaged computation cost per frame on RGGB raw data with resolution 960×540**.**

Model	Time(ms)	GFLOPs	raw	sRGB
RViDeNet [32]	-	1965.0	44.08 / 0.9881	40.03 / 0.9802
FastDVDnet [22]	44.3	665.0	44.30 / 0.9891	39.91 / 0.9812
EMVD [14]	24.0	79.5	44.05 / 0.9890	39.53 / 0.9796
BSVD-16 (ours)	**9.9**	**78.76**	44.10 / 0.9884	40.17 / 0.9804
BSVD-24 (ours)	14.2	175.46	**44.39** / **0.9894**	**40.48** / **0.9820**

streams of resolution 960×540 at 30Hz framerate, which achieves real-time performance.

Furthermore, we half the channels in BSVD-64 to BSVD-32 as a lightweight version. Compared with real-time SOTA FastDVDnet [22], our BSVD-32 has more than 60% reduction in runtime, with a 0.57dB increase of average PSNR on the DAVIS test set and 0.15dB on the Set8.

Blind denoising on DAVIS and Set8 datasets. Our method is also applicable for blind video denoising without the noise map as input. We compare ours with previous blind denoising SOTA UDVD [20], and a concurrent work [30]. As shown in Table 3, our BSVD-64-blind shows better fidelity.

Non-blind denoising on CRVD dataset. In Table 4, we compare with baselines on real-world raw data. Since the real-time baseline EMVD [14] has a very low computation cost, we shrink the channel of our model and train BSVD-24 and BSVD-16. Compared with FastDvDnet, our BSVD-24 produces better images with 3× speedup. Compared with EMVD, our BSVD-16 has 58.3% shorter inference time, and achieves better image fidelity in both raw and sRGB domains. Our model consists of only 2d convolutions, without special designs for raw data prior, such as attention, depth-wise separable convolution in EMVD. Thus, BSVD-16 has shorter runtime with similar GFLOPs as EMVD. As shown in Figure 7, our results have sharper edges on CRVD dataset. More results are shown in the supplement.

Table 5: Quantitative ablation study based on our BSVD-64. The "Ave" denotes the average PSNR of $\sigma \in \{10, 20, 30, 40, 50\}$**. Due to the space limit, we omit the results for** $\sigma \in \{20, 40\}$**.**

Model	DAVIS 10	30	50	Ave	Set8 10	30	50	Ave
Ours-Uni [7]	39.40	34.40	32.14	35.06	36.51	31.70	29.54	32.34
Ours-BN	39.14	34.57	32.33	35.13	36.50	31.86	29.72	32.46
Ours-U-Net [19]	39.54	34.66	32.44	35.30	36.56	31.87	29.76	32.49
Ours-MIMO [12]	39.68	34.83	32.61	35.46	36.68	31.98	29.86	32.60
Ours	**39.81**	**35.09**	**32.91**	**35.70**	**36.74**	**32.14**	**30.06**	**32.75**

4.4 Application of Buffer in FastDVDnet.

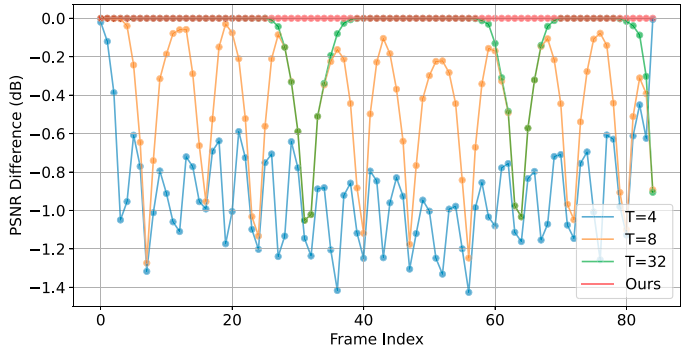
Our buffer-based pipeline-style inference can be applied to the existing method. FastDVDnet is a two-stage sliding-window-based method that conducts temporal fusion at the input layer of each U-Net. We utilize the pre-trained checkpoint and buffer the intermediate feature during each forward inference, which modifies the original computation graph into pipeline style. As a result, we half the runtime from 42ms to 23ms with the same image fidelity as the original implementation. More details are in the supplement.

5 ABLATION STUDIES

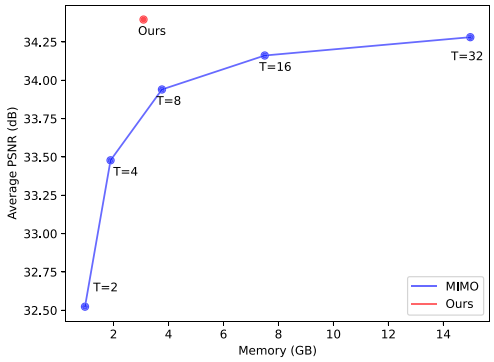
5.1 Pipeline Inference vs MIMO

We modify our BSVD-64 into a MIMO framework using temporal shift operation [12] with the same model parameters. We set the clip size $T_{clip} = 8$, which is close to previous works (e.g., 5 for FastDVDnet [22] and ReMoNet [30], 7 for PaCNet [23]).

As shown in table 5, our buffer-based pipeline inference framework brings 0.24dB improvement in average PSNR on DAVIS test



(a) PSNR difference at each temporal index



(b) Average PSNR and memory consumption

Figure 8: Fidelity and memory comparison between our pipeline framework and MIMO for a denoised video sequence *sunflower* from Set8 at $\sigma = 50$. (a) Our inference method solves the PSNR drop on the two edges of the clips with up to 1dB improvement. (b) Our framework consumes lower memory than $T_{clip} = 8$ with a 0.45dB improvement at average PSNR.

PSNR / SSIM
Ground Truth

29.94 / 0.879
Ours w/o buffer

31.76 / 0.924
Ours

Figure 9: Qualitative ablation study of inference method.

set. To further analyze our image fidelity, we demonstrate the PSNR difference at each frame index with clip size $T_{clip} = 4, 8, 32$ on the video *sunflower* from Set8 dataset in Figure 8(a). The MIMO framework suffers significant fidelity degradation (up to 1 dB in PSNR) at the boundary of clips (e.g., 32$^{\text{nd}}$ frame). Although increasing T_{clip} improves the fidelity of the middle frames in the clip, it can not achieve obvious improvement on the two temporal edges of the clip. As shown in Figure 8(b), increasing the T_{clip} of MIMO framework results in linear growth of memory consumption. In contrast, our framework processes the video continuously in the constant memory. Figure 9 provides another qualitative comparison with $T_{clip} = 8$ MIMO framework at 32th frame of the "skate-jump" sequence from DAVIS. Our bidirectional buffer improves the sharpness of the tree, especially in areas close to the moving player.

5.2 Bidirectional Feature Propagation

Bidirectional vs unidirectional temporal fusion. MoViNet [7] propose a buffer-based unidirectional temporal fusion for video steam recognition. In Table 5, we modify our bidirectional fusion to unidirectional temporal fusion with stream buffer [7]:

$$Z^{i,l+1}_{fused} = \quad Conv(B^{-1,l} \oplus Z^{i,l}_{[2f:]}). \tag{7}$$

Then, update the buffer $B^{-1,l}$ with $Z^{i,l}_{[0:2f]}$. The unidirectional version of our method "Ours-Uni" does not need future frames and thus does not have frame latency. However, the results in table 5 show a large gap of 0.64dB between bidirectional and unidirectional fusion. For low-level video tasks, the bidirectional fusion can be more advantageous since the future frames also have similar and helpful features to be utilized by each frame.

Table 6: Ablation study for the number of buffer blocks N.

Model	N	RF	Scale	DAVIS-Ave	Set8-Ave
Ours-w/o Fusion	0	1	-	34.01	31.63
Ours-only Pixel	2	5	Pixel	34.03	31.66
Ours-with Pixel	24	49	Down + Pixel	35.12	32.35
BSVD-64 (ours)	16	33	Down	**35.70**	**32.75**

Number of buffer blocks N. In Table 6, we add buffer blocks at pixel resolution or remove them at downsampled resolution, using the same backbone. Larger N brings a larger receptive field (RF) at the cost of longer latency during online inference. The result shows that pixel-level fusion is not necessary. Thus, we only fuse the downsampled features, which can be more robust against noise.

5.3 Backbone

Batch normalization. We train a W-Net with BN layers and tune other hyperparameters. BN layers decreases the average PSNR by 0.57dB on the DAVIS test set, as shown in Table 5.

W-Net vs U-Net. We replace our backbone with a U-Net with increased channels, such that the new model has similar FLOPs and runtime as W-Net. As shown in Table 5, W-Net outperforms U-Net by 0.40dB. This result indicates that a second U-Net for refinement may help to remove the remaining noise.

6 CONCLUSION

We propose a SOTA streaming video denoising method BSVD that outperforms existing methods on videos with synthetic and real noise in both inference speed and image fidelity. Our pipeline-style inference with Bidirectional Buffer Blocks allows bidirectional temporal fusion for online streaming video processing, which is proved to be more effective than unidirectional fusion. In addition, we solve the degradation of clip edges, which exists in MIMO frameworks. Our method is effective for both non-blind and blind denoising, and

is also general for similar architectures. Extensive experiments on public datasets have demonstrated the effectiveness of our method.

REFERENCES

[1] Pablo Arias and Jean-Michel Morel. 2018. Video Denoising via Empirical Bayesian Estimation of Space-Time Patches. *J. Math. Imaging Vis.* (2018).

[2] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. 2011. Non-Local Means Denoising. *Image Process. Line* 1 (2011).

[3] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. 2021. BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond. In *CVPR*.

[4] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. 2007. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing* (2007).

[5] Axel Davy, Thibaud Ehret, Jean Michel Morel, Pablo Arias, and Gabriele Facciolo. 2018. Non-Local Video Denoising by CNN. *arXiv* abs/1811.12758 (2018).

[6] Anna Khoreva, Anna Rohrbach, and Brent Schiele. 2018. Video Object Segmentation with Referring Expressions. In *ECCV*.

[7] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. 2021. MoViNets: Mobile Video Networks for Efficient Video Recognition. In *CVPR*.

[8] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. 2013. A Nonlocal Bayesian Image Denoising Algorithm. *SIAM J. Imaging Sci.* (2013).

[9] Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. 2022. VRT: A Video Restoration Transformer. *arXiv:2201.12288* (2022).

[10] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021. SwinIR: Image Restoration Using Swin Transformer. In *ICCV Workshops*.

[11] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 2017. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *CVPR Workshops*.

[12] Ji Lin, Chuang Gan, and Song Han. 2019. TSM: Temporal Shift Module for Efficient Video Understanding. In *ICCV*.

[13] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen O. Egiazarian. 2012. Video Denoising, Deblocking, and Enhancement Through Separable 4-D Nonlocal Spatiotemporal Transforms. *IEEE Transactions on Image Processing* (2012).

[14] Matteo Maggioni, Yibin Huang, Cheng Li, Shuai Xiao, Zhongqian Fu, and Fenglong Song. 2021. Efficient Multi-Stage Video Denoising With Recurrent Spatio-Temporal Fusion. In *CVPR*.

[15] Matteo Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. 2013. Nonlocal Transform-Domain Filter for Volumetric Data Denoising and Reconstruction. *IEEE Transactions on Image Processing* (2013).

[16] Anton Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. 2016. MOT16: A Benchmark for Multi-Object Tracking. *ArXiv* abs/1603.00831 (2016).

[17] Ben Mildenhall, Jonathan T. Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. 2018. Burst Denoising With Kernel Prediction Networks. In *CVPR*.

[18] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. 2017. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. In *CVPR*.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*.

[20] Dev Yashpal Sheth, Sreyas Mohan, Joshua L. Vincent, Ramon Manzorro, Peter A. Crozier, Mitesh M. Khapra, Eero P. Simoncelli, and Carlos Fernandez-Granda. 2021. Unsupervised Deep Video Denoising. In *ICCV*.

[21] Matias Tassano, Julie Delon, and Thomas Veit. 2019. Dvdnet: A fast network for deep video denoising. In *ICIP*.

[22] Matias Tassano, Julie Delon, and Thomas Veit. 2020. FastDVDNet: Towards real-time deep video denoising without flow estimation. In *CVPR*.

[23] Gregory Vaksman, Michael Elad, and Peyman Milanfar. 2021. Patch Craft: Video Denoising by Deep Modeling and Patch Matching. In *ICCV*.

[24] Xintao Wang, Kelvin C. K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. EDVR: Video Restoration With Enhanced Deformable Convolutional Networks. In *CVPR Workshops*.

[25] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. 2018. ESRGAN: Enhanced super-resolution generative adversarial networks. In *ECCV Workshops*.

[26] Bihan Wen, Yanjun Li, Luke Pfister, and Yoram Bresler. 2017. Joint Adaptive Sparsity and Low-Rankness on the Fly: An Online Tensor Reconstruction Scheme for Video Denoising. In *ICCV*.

[27] Bihan Wen, Saiprasad Ravishankar, and Yoram Bresler. 2019. VIDOSAT: High-Dimensional Sparsifying Transform Learning for Online Video Denoising. *IEEE Transactions on Image Processing* (2019).

[28] Xide Xia and B. Kulis. 2017. W-Net: A Deep Model for Fully Unsupervised Image Segmentation. *ArXiv* abs/1711.08506 (2017).

[29] Zhihao Xia, Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. 2020. Basis prediction networks for effective burst denoising with large kernels. In *CVPR*.

[30] Liuyu Xiang, Jundong Zhou, Jirui Liu, Zerun Wang, Haidong Huang, Jie Hu, Jungong Han, Yuchen Guo, and Guiguang Ding. 2022. ReMoNet: Recurrent Multi-output Network for Efficient Video Denoising. In *AAAI*.

[31] Lu Xu, Jiawei Zhang, Xuanye Cheng, Feng Zhang, Xing Wei, and Jimmy S. J. Ren. 2021. Efficient Deep Image Denoising via Class Specific Convolution. In *AAAI*.

[32] Huanjing Yue, Cong Cao, Lei Liao, Ronghe Chu, and Jingyu Yang. 2020. Supervised Raw Video Denoising with a Benchmark Dataset on Dynamic Scenes. In *CVPR*.

[33] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2022. Restormer: Efficient Transformer for High-Resolution Image Restoration. In *CVPR*.

[34] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing* (2017).