Research Article

ADVANCED IMAGING

Block-modulating video compression: an ultralow complexity image compression encoder for resource-limited platforms

Siming Zheng,[a,†] Yujia Xue,[b,†] Waleed Tahir,[b] Zhengjue Wang,[c] Hao Zhang,[d] Ziyi Meng,[e] Gang Qu,[a] Siwei Ma,[f] and Xin Yuan[a,*]

[a]Research Center for Industries of the Future (RCIF) and School of Engineering, Westlake University, Hangzhou, China
[b]Department of Electrical and Computer Engineering, Boston University, Boston, USA
[c]National Key Laboratory of Radar Signal Processing, Xidian University, Xi'an, China
[d]State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China
[e]Westlake Intelligent Vision Technology, Hangzhou, China
[f]School of Computer Science, Peking University, Beijing, China

Abstract. Considering the image (video) compression on resource-limited platforms, we propose an ultralow-cost image encoder, named block-modulating video compression (BMVC) with an extremely low-cost encoder to be implemented on mobile platforms with low consumption of power and computation resources. Accordingly, we also develop two types of BMVC decoders, implemented by deep neural networks. The first BMVC decoder is based on the plug-and-play algorithm, which is flexible with different compression ratios. The second decoder is a memory-efficient end-to-end convolutional neural network, which aims for real-time decoding. Extensive results on the high-definition images and videos demonstrate the superior performance of the proposed codec and the robustness against bit quantization.

Keywords: block modulation; image compression encoder; resource-limited platforms.

Received Jan. 22, 2024; revised manuscript received Jul. 3, 2024; accepted Jul. 9, 2024; published online Aug. 7, 2024.

[DOI: 10.3788/AI.2024.10006]

1. Introduction

Image and video compression algorithms have been developed for about 30 years, while state-of-the-art coders are still mainly based on the moving picture experts group MPEG) structure[1], which was originally developed for broadcasting. However, 30 years ago, videos were usually captured and produced in studios and thus the encoding time and cost can be very long and expensive, while the decoder at the customer's end needs to be of low complexity since it is in every family home (on television). Nowadays, image and video codecs are all over mobile platforms like cellphones, drones, etc., where images and videos can be captured anywhere, anytime, given enough power. Moreover, the devices (e.g., cell phones, laptops, and desktops) we now use to decode image and video streams have by magnitude higher computation power than it was decades ago. Therefore, now it is natural for the image and video compression to evolve towards the combination of a low-cost encoder and a (possibly) computationally heavy decoder. The low-complexity encoder has been studied in the literature of information and communication theory under the topic distributed source coding (DSC)[2]. A low-cost encoder is desired because we aim for applications on resource-limited robotic platforms such as drones.

In this paper, we consider the image and video compression codecs on those mobile platforms with sensitive constraints on battery, computation, and bandwidth. In particular, we highlight drones and robotics as representative applications. In these use cases, only the low-cost encoder needs to be implemented in real time on the mobile platform, but the decoding can happen after transmission on other tabletop platforms such as edges[3] or the cloud. Since most of these mobile platforms are running on

*Address all correspondence to Xin Yuan, xyuan@westlake.edu.cn
†These authors contributed equally to this work.

standalone batteries, a power saving on the encoder can extend the running time of other sensors and motion modules on drones or robotics, which is of significant interest in extreme cases such as moon rovers and other military applications.

Bearing this concern in mind, we propose an ultralow-complexity image and video codec using block modulation, dubbed the block-modulating video compression (BMVC) codec. The underlying principle of BMVC is to mask the high-resolution image (via predefined binary random coding patterns composed of {0,1}) and then decompose it into different small (modulated) blocks. Finally, these blocks are summed up to a single block and quantized as the compressed signal to be transmitted. Since no multiplication is involved during this encoding process, the complexity of the BMVC encoder is way lower than MPEG-based codecs. Moreover, the summation over modulated image blocks by binary masks can be essentially implemented as additions of pixel readouts according to a predefined look-up table during this encoding process.

Therefore, the computation cost of the BMVC encoder can be minimized, compared to transform-based (DCT, DWT) compression algorithms. Hereby, the mask pattern plays the role of basis or key during the compression, which is predeployed on the mobile platform. Without loss of generality, we use random binary patterns, with equal probability being 1 or 0, which is stored as a look-up table on the mobile platform. Note that only a single encoding mask is generated to encode all images and videos.

1.1. BMVC pipeline: key idea

Figure 1 shows the basic principle of the proposed BMVC encoder, where the input is a raw image captured by the sensor, e.g., a charge-coupled device (CCD) or complementary metal–oxide semiconductor (CMOS) camera, with a spatial resolution of $N_h \times N_w$ pixels, where N_h denotes the height and N_w denotes the width, respectively. Hereby, we consider the gray-scale images. BMVC can be extended to color images on the Bayer patterns or imposed on red, green, and blue (RGB) channels separately or on the Y, U, and V (YUV) channels, respectively. A binary mask of the same image size composed of {0,1} with each one of half-probability, is predefined and stored on the mobile platform and plays the role of a key (or basis) in BMVC.

Both the high definition (HD) image and the mask are divided into nonoverlapping blocks of size $B_h \times B_w$. Overlapping blocks can also be used in our BMVC framework with minimum changes in the decoder; the performance is similar to

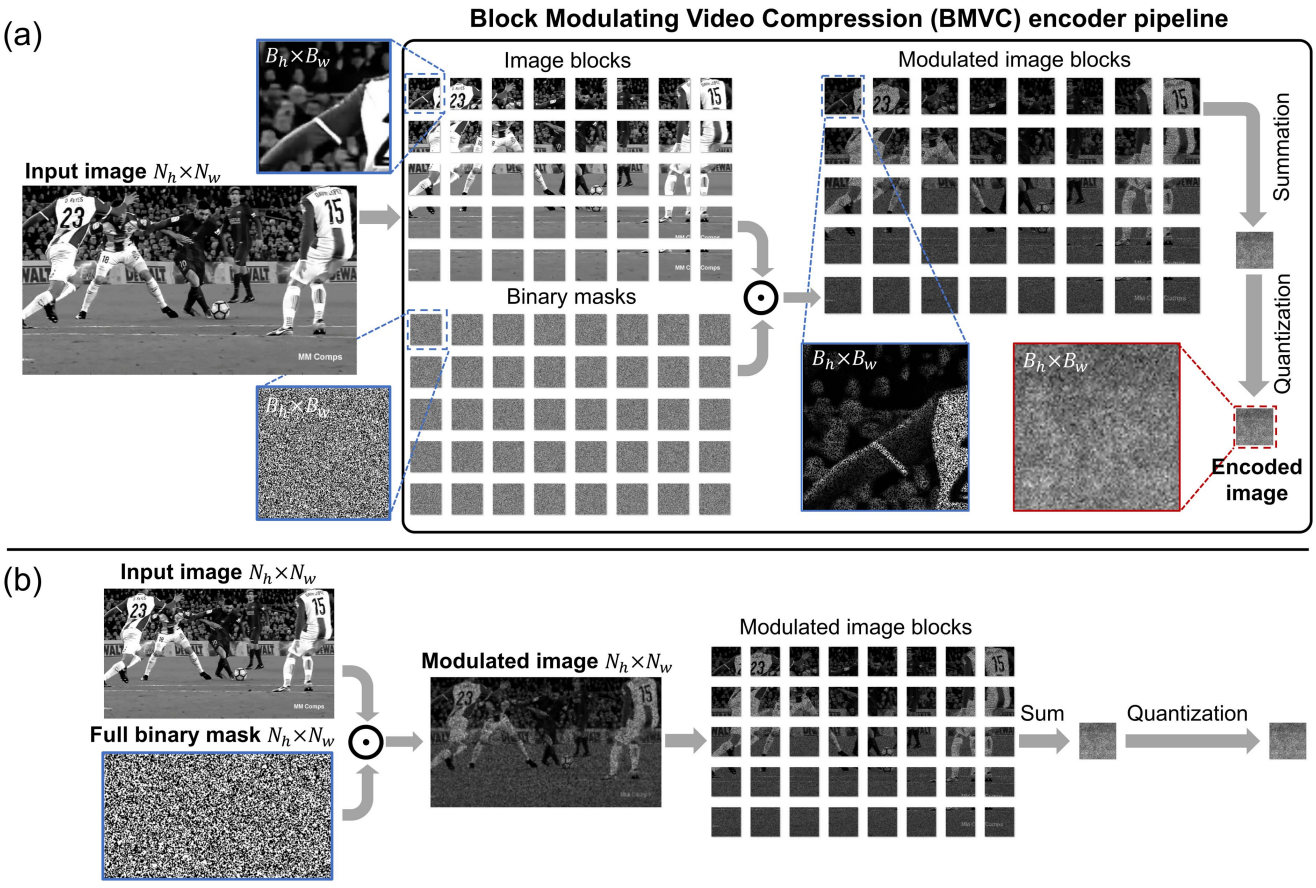

Fig. 1 Pipeline of the proposed BMVC encoder. (a) For each input image with a size of $N_h \times N_w$, the input frame is first divided into nonoverlapping blocks of size $B_h \times B_w$. Then the image blocks are modulated (element-wise multiplication) by binary masks of the same size. Next, these modulated image blocks are summed together to yield a single block of size $B_h \times B_w$. At last, the summed modulated image block is quantized to a user-defined bit depth (8–16 bit) and then transmitted to the receiver. An equivalent encoding pipeline is shown in (b), where the modulation happens before dividing into blocks.

the nonoverlapping case. Following this, the mask blocks are superimposed onto each image block to yield the modulated image blocks with the same block size, but half the number of pixels are missing. Next, these modulated blocks are summed into a single block, which is then quantized to the encoded image block. This quantized single block is the only information we need to transmit after encoding, i.e., the output of the encoder. Note that the masks are predefined and shared between the encoder and decoder, and thus do not need to occupy additional bandwidth. Hereby, the number of divided image blocks defines the compression ratio (Cr) of this encoding process $C_r = \frac{N_h N_w}{B_h B_w}$. Notably, the summation and quantization may hurt the dynamic range of the compressed data, but we later show the BMVC pipeline is robust against different quantization bits and able to provide consistent performance in a wide range of compression ratios.

On the decoder side, after receiving the encoded block, the BMVC decoding algorithms are employed to reconstruct the original HD image, provided the masks known a priori used in the encoder. In this paper, we consider two reconstruction algorithms, one being the plug-and-play (PnP)[4–7] optimization algorithm (BMVC-PnP) with a deep denoising neural network in each iteration[8] and the other being an end-to-end convolutional neural network (BMVC-E2E) for real-time video decoding.

The proposed BMVC pipeline is frame-independent, meaning that each image or frame (from a video sequence) can be independently decoded without knowing its previous image or frame.

While it is true that in video compression applications, further exploration of the temporal redundancy can significantly boost the compression ratio, the extra computing of temporal difference between frames inevitably increases the encoding cost. Our goal here is to minimize the computation cost on the encoder hardware so that the resources can be saved for other modules on the robotics.

Therefore, we do not consider temporal processing in our current BMVC pipeline. However, it is worth mentioning that any type of temporal processing is compatible and can be added to the existing BMVC pipeline for a higher compression ratio at the cost of increased encoder complexity.

1.2. Contributions

We want to emphasize that BMVC is not designed to replace the current codec standard but to provide an alternative option for platforms under extreme power/computation constraints.

Here we summarize the contributions of this work:

(1) A brand-new image and video compression codec is proposed for resource-limited platforms.

(2) The proposed BMVC encoder has extremely low computation cost.

(3) Two high-quality BMVC decoders are developed, with each enjoying different benefits, i.e., the BMVC-PnP features flexibility and BMVC-E2E for real-time applications.

(4) The proposed BMVC codec is fully evaluated by different compression ratios and different quantization bits, and benchmarked with other compression methods.

(5) Extensive results demonstrate the superior and consistent performance of BMVC on resource-limited platforms.

2. Background Knowledge

Similar to other codecs, the proposed BMVC pipeline is composed of an encoder and a decoder. As shown in Fig. 1(a), the encoder consists of blocking, masking, summation, and quantization.

It is worth noting that the blocking and masking steps can be switched; specifically, a mask of size $N_h \times N_w$ can be used to first mask the entire image (or an HD frame) and then the modulated image (of size $N_h \times N_w$) can be divided into blocks of size $B_h \times B_w$; following this, these modulated blocks are summed to a single block. This alternative interpretation of BMVC gives us the benefit that only one full-frame mask of size $N_h \times N_w$ needs to be predefined and stored before deployment, as shown in Fig. 1(b).

One underlying principle of the BMVC encoder is compressive sensing (CS)[9,10], where a small number of measurements can be used to reconstruct the full signal with a higher dimension under some mild conditions. Specifically, BMVC shares the same spirit with snapshot compressive imaging (SCI)[11]. In the following sections, we review the basic idea of SCI and deep-learning (DL) methods for CS inversion, i.e., the reconstruction process.

2.1. SCI

SCI utilizes a 2D detector to capture high-dimensional (≥3D) data.

It was first developed to capture high-speed videos (spatio-temporal data cubes) with a relatively low-speed camera[12–14] or to capture hyperspectral images (spatiospectral data cubes) in a single shot[15,16] among other applications[17–22]. Generally, the fundamental idea of SCI is to modulate each low-dimensional slice (frames or spectral channels) from an underlying high-dimensional data structure with a different mask and then sum these modulated slices into a single low-dimensional measurement.

In our proposed BMVC encoding process, as shown in Fig. 1, we can interpret its forward model in a slightly different way following the same spirit of SCI. Consider the image blocks in BMVC as low-dimensional slices in the SCI modality, with the modulation masks corresponding to the SCI masking scheme. Then the BMVC-modulated blocks are summed to a single block measurement, which corresponds to the compressed measurement in SCI. From this perspective, the encoding process of BMVC is essentially the same as that of SCI, whereas there is a key difference between BMVC and SCI. That is, in SCI, the frames in the video sequence or the spectral channels in the hyperspectral data cube are strongly correlated, as they share highly similar spatial features, while in BMVC, each block corresponds to a different portion of the image, and these blocks are not necessarily correlated, which makes the BMVC decoding more challenging than SCI modalities. Though the decoding task poses a big challenge, in this paper we show that using advanced DL-based CS inversion algorithms, high-resolution images can still be faithfully decoded from the highly ill-posed, low-cost BMVC encoder.

2.2. DL for SCI inversion

The inverse problem of CS is ill posed, as there are more unknown parameters to be estimated than known measurements. Toward this end, different priors have been employed as

regularizations to help solve the CS inverse problem. Widely used priors include sparsity[23] and piece-wise smoothness such as total variation (TV) and low rankness of similar image patches.

Implicit regularizations have also been explored using standard denoising algorithms (such as NLM[24] and BM3D[25]) as PnP priors[26]. Other algorithms have also been used to solve the SCI reconstruction, such as TwIST[27], GAP-TV[28], and DeSCI[29].

Recently, DL has been used for solving inverse problems in computational imaging systems for high-quality and high-speed reconstruction. Specially, existing DL-based inversion algorithms can be categorized into three different classes[11]: (1) end-to-end convolutional neural networks (E2E-CNNs) for high-speed reconstruction[14,16,30–33], (2) deep unfolding/unrolling networks[34–40] with interpretability, and (3) the PnP algorithms[41,42] using pretrained denoising networks as implicit priors to regularize inverse problems.

For the BMVC decoding case considered in this work, due to the large scale of the data, we employ two decoding algorithms: a BMVC-PnP and a BMVC-E2E. Regarding the data dimension, we consider the HD data of size 1080×1920, which is decomposed into blocks of various sizes with a wide range of Cr from 24 to 150. We use 1080×1920 as an example to demonstrate the idea of BMVC due to its wide usage. BMVC is ready to be used in other-sized images and videos. In fact, there are no constraints on the input size of the BMVC codec. Given such a wide Cr range, it is challenging to train a deep unfolding network with multiple CNNs for this large-scale data due to the limited GPU memory. For applications where real-time reconstruction is not necessary, we first develop an iterative BMVC decoder based on the PnP algorithm. The BMVC-PnP decoder employs a pretrained FFDNet[43] as the denoiser to regularize the inversion, which makes the PnP framework flexible with different Crs. On the other hand, there are still exist many applications that require real-time decoding. To achieve a more practically usable BMVC codec, we further develop a noniterative, feed-forward decoder based on an end-to-end CNN. It is worth mentioning that even for the end-to-end approach, the training of BMVC-E2E is not trivial, since it requires an ultra-deep CNN that cannot be fit in the memory of a single GPU. Fortunately, recent work on reversible networks[44] has shown to reduce the memory requirement during the training phase, and thus we build our BMVC-E2E decoder around 3D reversible convolution modules[31]. Finally, the BMVC-E2E decoder is capable of decoding images at a frame rate of 4–14 frames per second (fps) depending on the Cr with a single decent GPU.

3. BMVC

In this section, we elaborate on the mathematical details of the BMVC pipeline. In particular, we use gray-scale images as an example to derive the mathematical model. As mentioned before, BMVC is ready to handle RGB color images and videos, where we can conduct BMVC on all RGB channels or YUV (or YCbCr) channels. In our experiments, we have further found that decent decoding results can be obtained by only performing BMVC on the Y channel and a simple downsampling/upsampling can be used for the U and V channels.

Let $\mathbf{X} \in \mathbb{R}^{N_h \times N_w}$ denote the input HD image [left-middle in Fig. 1(a)]. The first step is to divide the input image $\mathbf{X}$ into N_b nonoverlapping small blocks of size $\mathbf{X}_b \in \mathbb{R}^{B_h \times B_w}$, where $B_h < N_h$ and $B_w < N_w$. It is straightforward to show that the number of blocks equals the compression ratio $N_b = \mathrm{Cr}$. Then the image blocks are element-wise modulated by the binary masks, which gives us

$$\tilde{\mathbf{X}}_b = \mathbf{X}_b \odot \mathbf{M}_b, \qquad \forall\ b = 1, \ldots, N_b, \tag{1}$$

where $\odot$ denotes the element-wise multiplication and $\mathbf{M}_b \in \{0,1\}^{B_h \times b_w}$ denote the binary masks shown in the middle-bottom in Fig. 1(a).

Note that in practice, Eq. (1) can be performed by summations according to a look-up table rather than any multiplication. As a result, the actual computational pipeline of the BMVC encoder requires only additions and no multiplications. Detailed analysis of the BMVC complexity can be found in Sec. 4.3.

Generally, the number of blocks N_b can be obtained by

$$N_b = \left\lceil \frac{N_h}{B_h} \right\rceil \left\lceil \frac{N_w}{B_w} \right\rceil, \tag{2}$$

where $\lceil a \rceil$ denotes the minimum integer no smaller than a. For optimal BMVC encoding and decoding performance, we choose both B_h and B_w to be divisors of N_h and N_w so that we can maximize the compression ratio by having nonoverlapping blocks. Note that we only need to save the indices of each block and then stitch them together to get the full-sized image back in the decoder.

3.1. Compressed measurement

After blocking and binary modulation, the next step is to sum all these modulated blocks to yield a single compressed measurement, i.e.,

$$\mathbf{Y}_0 = \sum_{b=1}^{N_b} \tilde{\mathbf{X}}_b = \sum_{b=1}^{N_b} \mathbf{X}_b \odot \mathbf{M}_b. \tag{3}$$

This $\mathbf{Y}_0 \in \mathbb{R}^{B_h \times B_w}$ is the noise-free compressed measurement that we are going to transmit.

Before sending the measurement, the last step is bit quantization, which imposes additional quantization error to the actual compressed signal $\mathbf{Y}$. In this case, we have the forward model of

$$\mathbf{Y} = \sum_{b=1}^{N_b} \mathbf{X}_b \odot \mathbf{M}_b + \mathbf{E}_q, \tag{4}$$

where $\mathbf{E}_q \in \mathbb{R}^{B_h \times B_w}$ denotes the quantization error. In our experiments, we performed 8–16 bit quantization and found the BMVC pipeline is robust against quantization error and provides consistent reconstruction quality.

3.2. Forward model in vectorized formulation

For the sake of describing the reconstruction algorithm employed in the decoder, we hereby introduce the vectorized formulation of Eq. (3). Let

$$\boldsymbol{x}_b = \mathrm{Vec}(\mathbf{X}_b), \tag{5}$$

$$\boldsymbol{m}_b = \mathrm{Vec}(\mathbf{M}_b), \tag{6}$$

$$\boldsymbol{y} = \mathrm{Vec}(\mathbf{Y}), \tag{7}$$

where the Vec$(\cdot)$ operator vectorizes the ensued matrix by stacking its columns.

After vectorization, Eq. (3) can be reformulated as

$$\boldsymbol{y}_0 = [\mathrm{Diag}(\boldsymbol{m}_1), \mathrm{Diag}(\boldsymbol{m}_2), \cdots, \mathrm{Diag}(\boldsymbol{m}_{N_b})]_{\overset{\text{def}}{=}\boldsymbol{\Phi}\in\mathbb{R}^{B_hB_w\times(B_hB_wN_b)}} \times \underbrace{\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{N_b} \end{bmatrix}}_{\overset{\text{def}}{=}\boldsymbol{x}\in\mathbb{R}^{B_hB_wN_b}}, \tag{8}$$

where Diag$(\cdot)$ embeds its input vector to the main diagonal of a square diagonal matrix. Given the structure of the forward model $\boldsymbol{\Phi}$, note that $\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathbf{T}}$ is a diagonal matrix as

$$\mathbf{R}\overset{\text{def}}{=}\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \mathrm{Diag}(r_1, \ldots, r_{B_hB_w}), \tag{9}$$

$$r_i = \sum_{b=1}^{N_b} m_{i,b}^2, \quad \forall\ i = 1, \ldots, B_hB_w, \tag{10}$$

where $m_{i,b}$ is the ith element in m_b.

When the quantization error e_q is considered, the forward model in Eq. (4) can be written as

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} + e_q. \tag{11}$$

It is worth noting that Eq. (11) has the formulation of CS, but with a sensing matrix $\boldsymbol{\Phi}$ being a concatenation of diagonal matrices. The performance bound of this special sensing matrix was analyzed in Ref. [45].

Though there is a matrix multiplication in Eq. (11), during the encoding process, there is only masking (implemented by a look-up table) and summation as in Eq. (3). Note that, as mentioned in the introduction, only a single full-sized mask is needed to be predefined and stored for both encoder and decoder. This mask can be performed on consequent frames in a video sequence. This mask can also be designed for different user applications or designed for encryption purposes, for which only an authorized person can access the mask and then decode the video, while the compressed measurement is actually encrypted. This provides another benefit of BMVC.

3.3. PnP-based decoder of BMVC

So far, we have shown that the BMVC encoder has an extremely low complexity, which makes it perfect for use cases on resource-limited mobile platforms. On the other hand, the decoding process of BMVC becomes highly challenging due to the huge dimensionality mismatch (up to N_b or Cr times) between the encoded block and the original HD image.

Here we first discuss an optimization-based decoder that utilizes the PnP[6,7] algorithm. When the decoder receives the encoded image block from the encoder, the goal is to reconstruct the desired image, provided the modulation masks. Following the formulation in Eq. (11), the decoding is an ill-posed problem and thus similar to CS; priors need to be employed in the optimization,

$$\hat{\boldsymbol{x}} = \arg\min_x \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}\|_2^2 + \tau R(\boldsymbol{x}), \tag{12}$$

where $R(\boldsymbol{x})$ denotes the prior term to regularize the inverse problem. As mentioned in the background knowledge, various priors have been developed for CS and thus can be used for BMVC decoding. Recently, the deep prior learned by neural networks has been used to solve the SCI inversion, leading to the PnP algorithm[8].

Since the specific structure of the BMVC encoding operator $\boldsymbol{\Phi}$ [shown in Eq. (8)] shares the same mathematical form as SCI modality, the BMVC-PnP decoder has a similar structure as the PnP-GAP[8] framework to solve the inversion problem of BMVC decoding, and here GAP denotes generalized alternating projection[28,46].

To be concrete, the BMVC-PnP is an iterative decoder. Starting from $\boldsymbol{v}^{(0)}$, where the superscript denotes the iteration number, the BMVC-PnP is composed of the following two steps:

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{v}^{(j)} + \boldsymbol{\Phi}^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}})^{-1}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{v}^{(j)}), \tag{13}$$

$$\boldsymbol{v}^{(j+1)} = \mathrm{Denoise}(\boldsymbol{x}^{(j+1)}). \tag{14}$$

Due to the diagonal structure of $\mathbf{R} = \boldsymbol{\Phi}\boldsymbol{\Phi}^T$ shown in Eq. (9), Eq. (13) can be computed efficiently (an element-wise operation). Regarding Eq. (14), the key idea of PnP is that various denoisers can be used to regularize the outcome from the linear step Eq. (13) to achieve better results. Recently, it has been demonstrated that the fast and flexible denoising network, FFDNet[43], is flexible in terms of noise levels and can provide excellent results for different image-processing tasks. Therefore, in this work, we adopt the PnP framework with FFDNet as the BMVC-PnP decoder, as shown in Fig. 2.

Though the CNN-based FFDNet is very efficient for denoising, the BMVC-PnP decoder is still an iterative algorithm, and thus it cannot provide real-time results. For instance, in the experimental results, we let the BMVC-PnP decoder run for 60 iterations while gradually decrease the σ value in the FFDNet step ($\sigma = [5, 10, 20]$, each for 20 iterations). Finally, the runtime of the BMVC-PnP decoder is about 1 min per HD image.

3.4. End-to-end CNN decoder of BMVC

To address the speed issue and enable real-time BMVC applications, we present a second BMVC decoder that employs an end-to-end CNN architecture for faster BMVC decoding. In the following context, we will term the CNN-based end-to-end BMVC decoder as BMVC-E2E.

In order to make the BMVC-E2E robust and interpretable, we design the feed-forward CNN architecture based on unrolling the PnP optimization framework to a few stages[47], as shown in Fig. 3. Similarly, each stage now contains a linear projection operator to account for the BMVC forward encoding process, and a CNN to serve as an implicit regularization. Note that though the BMVC-E2E follows the structure of an unrolled optimization, it has two unique features, making it different from the PnP approach. First, unlike the CNN in the PnP approach, which is independently trained as an ad hoc denoiser, the BMVC-E2E decoder is trained in an end-to-end (multiple stage jointly) fashion to perform direct decoding. Second, remember that the whole purpose of the BMVC-E2E decoder

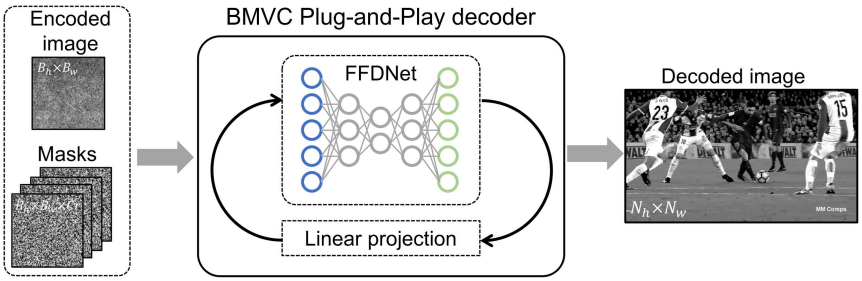

Fig. 2 PnP optimization-based decoding algorithm for BMVC-PnP. The encoded image block along with the modulation binary masks are fed into the BMVC-PnP decoder as inputs. The BMVC-PnP iteratively performs a linear projection step to account for the BMVC encoding process and a DL-based denoising step as an implicit prior. We use a pretrained FFDNet43 as the denoising CNN for its flexibility and robustness against various noise levels.

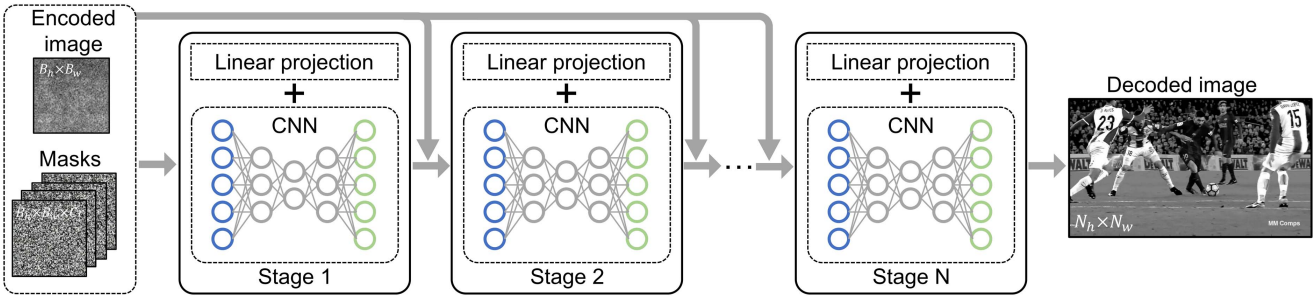

Fig. 3 E2E neural-network-based decoding algorithm for BMVC-E2E. The encoded image block along with the modulation binary masks are fed into the BMVC-E2E decoder as inputs. The feed-forward BMVC-E2E decoder consists of several stages, where each stage contains a linear projection step and a convolutional neural network. All BMVC-E2E decoders are trained in an E2E fashion. We use 2D-U-Net and 3D-CNN with reversible blocks (RevSCI) to facilitate memory-efficient training.

is to accelerate the inference time towards real-time decoding. The BMVC-E2E needs only a few stages (usually 2–3 stages are enough), while the BMVC-PnP generally requires ≥60 iterations to converge. These unique features of the end-to-end decoder make real-time BMVC decoding possible on standard GPU servers.

The linear projection step follows the identical derivation, as in Eq. (13). While the general structure of deep unfolding is easy to understand, how to design the network in each stage is extremely challenging for efficient BMVC decoding. As mentioned before, different from other applications of SCI, in BMVC, each block is not necessarily correlated with others. Therefore, it is important to extract nonlocal information in different blocks during BMVC decoding. Toward this end, after extensive experiments, we have found that 3D-CNN architecture is powerful enough to conduct this task by extracting information across nonlocal blocks. However, this introduces the challenge of running time, since a 3D-CNN model usually needs 12×−25× longer than its 2D-CNN counterparts.

Therefore, a trade-off between speed and quality has to be made. To balance between decoding quality and running time, we conducted extensive experiments on 2D- and 3D-CNN structures and how many stages we unroll the network into. Finally, we identified a few key observations to have fast and high-quality decoding. First, the BMVC-E2E decoder needs to incorporate at least two stages. This is because the first stage mainly conducts an initial inpainting of the missing pixels and generates blurry results. Additional stages are essential to retrieve the high-resolution features. Second, 3D-CNNs are shown to be more efficient at recovering fine features from nonlocal blocks than 2D structures. Especially for high Cr (Cr > 50) cases, a stack of dozens of 2D-CNNs no longer provides high-resolution decoded results. In these cases, we use two-stage 3D-CNNs in the decoder, aiming for high-quality reconstruction, though the speed is relatively slow,

(1) Network Structure: Empirically, after testing on a large number of images and videos under different scenes, we have found the following network settings in BMVC-E2E leading to a good trade-off between quality and speed:

i) For Cr ≤ 50, the BMVC-E2E decoders have two 2D-CNN (U-Net[48] structure) stages and one 3D-CNN (RevSCI[31] structure) stage. The U-Net stage has {32, 64, 128, 256, 512} filters on its downsampling branch and {256, 128, 64, 32, 16} filters on its upsampling branch. The RevSCI stage consists of 10 3D reversible blocks with 32 filter channels. After optimizing the inference phase by reducing to float16 precision, we eventually can achieve a 70 ms runtime (>14 fps) when decoding a 1080 × 1920 image on a server with a single GPU (Nvidia A6000).

ii) For Cr $>$ 50, the BMVC-E2E decoders have two 3D-CNN (RevSCI) stages, each with 11 3D reversible blocks of 64 filter channels. Trained BMVC-E2E decoders can process a 1080 $\times$ 1920 image with a runtime of 252 ms ($\sim$4 fps) per frame on the same GPU server. The increase of decoder runtime for a high Cr is because of the additional 3D-CNN stage in the E2E decoder, which guarantees better decoding quality, though at the cost of a longer inference time.

(2) Training Details: All BMVC-E2E decoders are trained end-to-end on the GoPRO data set[49] and the DAVIS HD data set[50] using Pytorch. For Cr $\leq$ 50 cases, the BMVC-E2E decoders are trained on a GPU server with 4 Nvidia GeForce Titan XP GPUs (each one with 12 GB RAM). The high Cr BMVC-E2E decoders (Cr $>$ 50) are trained on GPU servers with two Nvidia Tesla P40 GPUs (each one with 24 GB RAM).

Note that more stages with 3D-CNN can be used; this will lead to higher decoding quality but further increase the runtime. In addition, when the spatial size of the image changes, for instance when BMVC is being used for QHD, 4 K or 8 K resolutions, BMVC also shows the characteristics of efficient encoding. But attention is needed for the decoding computational complexity. Higher resolutions inherently increase the computational demands of decoding. BMVC can be optimized for these resolutions by employing parallel processing techniques suitable for multicore CPUs or leveraging the capabilities of modern GPUs for accelerated decoding. To summarize, the algorithm procedure of BMVC involves processing on either RGB or YUV (or YCbCr) channels. Taking YUV as an example, the input frame's Y channel is initially segmented into nonoverlapping blocks of identical size. These blocks undergo modulation via element-wise multiplication with corresponding binary masks of the same dimensions. The resulting modulated blocks are then summed and quantized to a user-defined bit depth (8–16 bit). Subsequently, the quantized blocks are transmitted to the receiver and decoded using the proposed decoder network. For the U and V channels, a simple downsampling/upsampling technique is applied.

4. Evaluation Result Background Knowledge

We consider the HD video with a spatial resolution of 1080 pixel $\times$ 1920 pixel. The peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM)[51] are employed as the metrics to evaluate decoded images. For RGB videos, we first transform them to YUV videos and conduct BMVC encoding and decoding on the Y channel while only performing downsampling for U and V channels in the encoder. In the decoder, BMVC-PnP- or BMVC-E2E-based algorithms are used for the Y channel, and bicubic interpolations are used for the U and V channels, respectively. Metrics are only calculated for the restored Y channel compared with the ground truth. We evaluate only the quality of the restored Y channel to isolate the effect of the BMVC pipeline from the interpolations of color channels. We benchmark the BMVC pipeline along with other compression methods on static frames from the UVG data set[52] and other standard images. Exemplar test images are shown in Fig. 4, which covers diverse scenes.

Examples of decoded videos using the BMVC pipeline at a wide range of Crs are presented in the Supplementary Materials (Video 1 and Video 2). Decoded gray-scale and RGB videos show the flexibility of BMVC in terms of video format.

4.1. BMVC decoder evaluation at various Crs

Since the compression ratio of BMVC depends on the block size, we consider the following block sizes for the HD video; the corresponding Crs are depicted in Table 1. The BMVC pipeline is tested for a wide range of Cr, ranging from 24 to 150. The average PSNR and SSIM of the decoded images (with two different BMVC decoders) in the test set are shown in Table 1 along with other related image compression methods, detailed in the next subsections.

Selected decoded images at representative Crs using BMVC-PnP and BMVC-E2E are presented in Fig. 5, where we can see that for a wide range of Crs, both BMVC-PnP and BMVC-E2E decoders are consistently able to provide decent decoded images with fine details. As expected, the resolution from BMVC starts to degrade as Cr increases. This decreasing trend can be seen from the texts in the zoom-in panels of the "bookcase" and "jockey" examples in Fig. 5. In addition, we also notice the reconstruction artifacts are different for the BMVC-PnP and BMVC-E2E due to different network structures being used.

Fig. 4 Test data set (set 13) we used to evaluate the BMVC pipeline and other compression methods.

Table 1 PSNR (top line in each cell in dB) and SSIM (bottom line in each cell) Performance for Different Compression Methods at a Wide Range of Crs on the HD Image of Size 1080 × 1920. The highest performance of CS-based methods is bold-faced.

Cr (N_b) BMVC: block size	150 108 × 128	120 108 × 160	100 108 × 192	80 108 × 240	72 120 × 240	60 216 × 160	50 216 × 192	40 216 × 240	32 270 × 240	24 270 × 320
BMVC-PnP	22.89, 0.682	24.32, 0.707	25.97, 0.745	27.30, 0.781	27.96, 0.797	28.83, 0.822	**29.87**, 0.848	**31.10**, **0.877**	**32.23**, **0.895**	**33.58**, **0.913**
BMVC-E2E	**26.59**, 0.802	**27.38**, 0.815	**27.80**, 0.821	**28.81**, 0.837	**29.31**, **0.849**	**30.74**, **0.871**	29.08, 0.839	29.23, 0.843	29.98, 0.854	31.05, 0.871
Random DS	8.95, 0.354	9.56, 0.401	10.10, 0.431	10.88, 0.460	11.33, 0.472	12.17, 0.491	13.30, 0.516	14.87, 0.555	16.61, 0.609	18.58, 0.690
Block CS	**26.59**, **0.819**	26.93, **0.826**	27.65, **0.837**	27.88, **0.843**	28.20, **0.849**	28.70, 0.857	29.38, **0.865**	29.83, 0.870	30.70, 0.876	31.79, 0.884
JPEG2000	30.30, 0.852	30.94, 0.862	31.42, 0.869	31.99, 0.877	32.80, 0.888	33.41, 0.897	33.84, 0.902	34.60, 0.911	36.45, 0.931	37.51, 0.941

4.2. BMVC versus other compression methods

We further compare the BMVC pipeline with other image and video compression algorithms: random downsampling (random DS), block-wise CS (block CS), and JPEG2000 compression.

(1) **Random Downsampling**: Since the BMVC encoding process is essentially aiming to compress the images, it is natural to first compare BMVC with random DS, which is an extreme case of CS by naively and randomly decimating some pixels in the image. Naive downsampling with low-pass prefiltering essentially loses high-frequency components, while CS preserves high-frequency features and further retrieves them by the inverse algorithms. Therefore, we only compare BMVC with random DS. The encoder of random DS is easily implemented by randomly sampling only $\frac{1}{\mathrm{Cr}}$ pixels to match with the Cr of BMVC and discard all the rest of the pixel values, which is equivalent to element-wise multiplication with a sparse binary mask. The decoder of random DS is implemented using

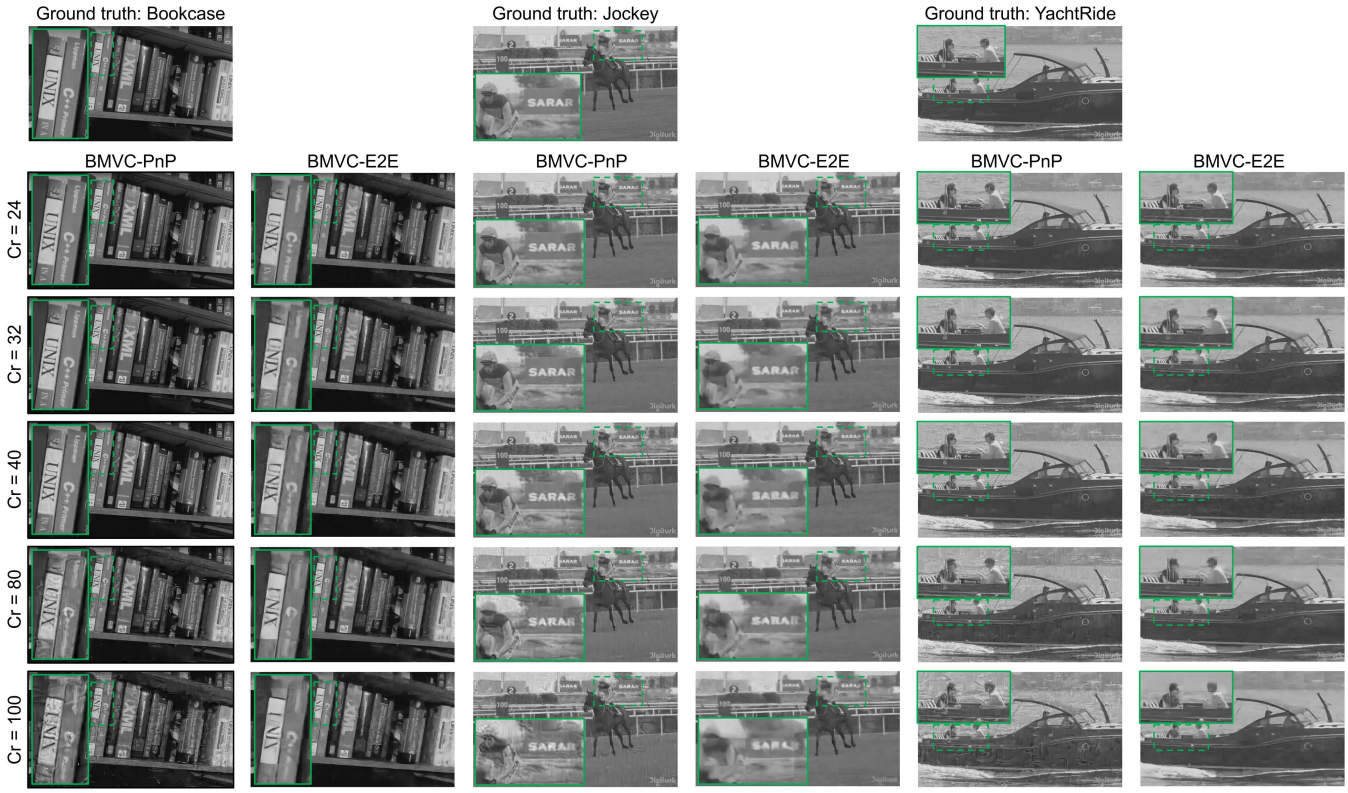

Fig. 5 Decoded image results at various Crs with the proposed BMVC-PnP and BMVC-E2E approaches. The BMVC-E2E results consistently have good decoding quality at both low and high Crs. The BMVC-PnP decoder provides higher image quality for low Crs while producing some denoising artifacts at high Crs.

the same PnP algorithm (PnP-FFDNet). Due to the nature of random DS, the decoder is essentially solving an image inpainting task with extremely low counts of measured pixels (0.6%–4%). Shown in Fig. 6, at a relatively low Cr = 32 case, the random DS PnP decoder can restore some of the spatial features but already suffers from massive reconstruction artifacts. At a higher Cr = 80 case, the random DS approach entirely fails. In terms of quantitative metrics, the PSNR performance of random DS is also the lowest for all Crs (Fig. 7, Table 1). We do notice that the recent advanced algorithms have been developed for image inpainting (thus random DS)[53]; however, these approaches are usually slow and only provide results with larger than 10% sampling rates. Compared with BMVC cases considered here, the Crs of random DS are low.

(2) **Block CS**: Another related CS algorithm is block CS, which utilizes the same CS principle. Unlike the traditional CS approach, which gathers compressed measurements from global image pixels (such as the single-pixel camera[23,54]), block CS computes its compressed measurements only on local blocks[55,56]; 24×24 blocks and a binary-sensing matrix composed of {0,1} are used in our experiments to be consistent with BMVC. Note the sensing matrix is dense for block CS

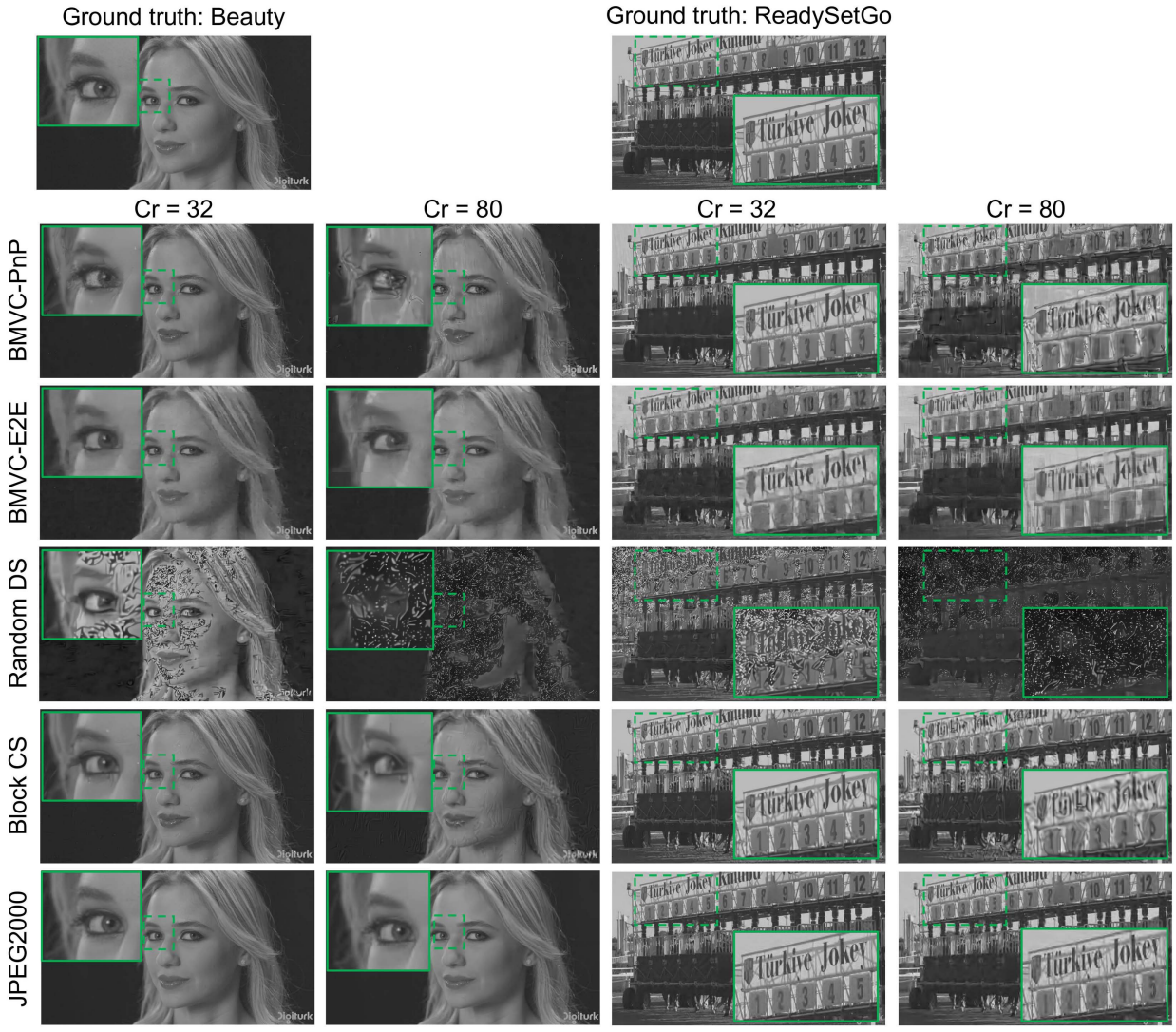

Fig. 6 Comparison of the BMVC pipeline with other image compression methods: random DS, block CS, and JPEG2000 compression. For the random DS and block CS experiments, we implemented their decoders based on the PnP algorithm with FFDNet as the flexible denoiser. Results are shown with a low Cr = 32 and a high Cr = 80. The two BMVC decoders provide consistently high-quality images. The random DS method fails because, in principle, the random DS decoder is solving an image inpainting task with only <3% pixels available. Block CS also shows equally good decoding results, but we will show later that block CS will deteriorate after aggressive data quantization. As expected, JPEG2000 compression gives the best image quality at Cr = 32 and 72 with minor blurriness, since JPEG2000 utilizes an optimal set of wavelet basis but with the cost of increased encoding computation cost.

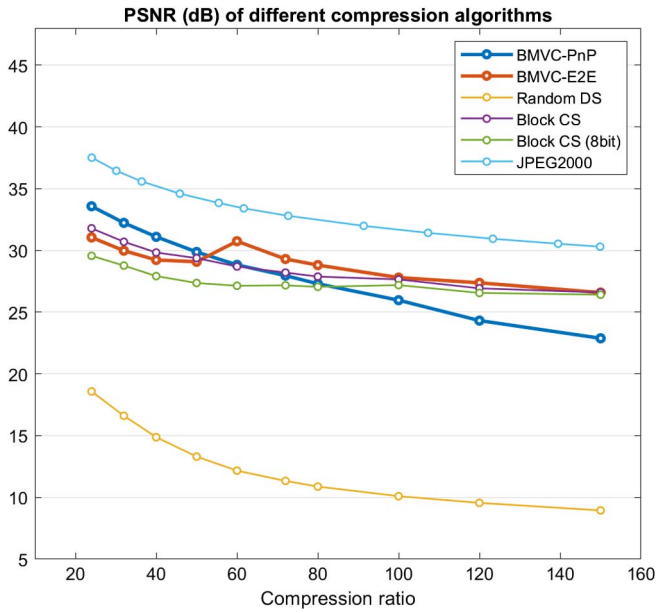

Fig. 7 PSNR performance of different compression methods at a wide range of Crs. PSNR value is computed for Y channels only. The BMVC-E2E has a PSNR increase at Cr = 60. This is because the BMVC-E2E decoder has an additional stage of 3D-CNN for all Cr ≥ 60.

(each block can be considered as a small image in the single-pixel camera), and we employ the same sensing matrix for different blocks.

For a fair comparison, we implement the block CS[57] decoder with the same PnP algorithm (PnP-FFDNet). We also notice both DL-based algorithms[58] and optimization-based algorithms[59] have been proposed for block CS recently. However, comparing these algorithms is beyond the scope of this work, and later we show that BMVC outperforms block CS on the robustness of bit quantization.

Figure 6 shows that the block CS approach and its PnP-based decoder perform well in both low and high Crs. The block CS pipeline provides visually on-par results with the BMVC pipeline.

Quantitative results in Fig. 7 and Table 1 also show its marginal improvement over BMVC at low Crs and slightly worse scores at high Crs. However, a potential pitfall of block CS is bit quantization. Later in Sec 4.4, we will show that BMVC is more robust to quantization bits and can provide more consistent performance after low-bit quantization than block CS. In addition, the choice of block size in block CS is also important. As the block size increases, the dynamic range of compressed data increases as well, which makes the approach more sensitive to quantization.

(3) **JPEG2000 compression**: For decades, JPEG/JPEG2000 compression and other MPEG-based video codecs have been gold standards, providing extraordinary compression performance. Here we test how JPEG2000 compression performs at similar Crs of BMVC. Unlike BMVC, random DS, and block CS, whose Cr can be explicitly defined and controlled, the Cr of JPEG2000 compression can be more complicated and includes the effects from both wavelet transform and entropy coding.

Note that even lossless JPEG2000 uses entropy coding to compress the file; to isolate the effect of entropy coding, we define the Cr of JPEG2000 compression as the ratio of lossless JPEG2000 file size over lossy compressed file size. As shown in Fig. 6 and Table 1, JPEG2000 indeed is a robust and efficient image-compression algorithm in terms of both quality and Cr. JPEG2000 consistently gives the highest PSNR performance across a wide range of tested Crs, from 24 up to 150. This performance of JPEG2000 is expected to be better than other CS-based methods because JPEG2000 sorts and saves the dominant coefficients in the wavelet domain, which is known to be an optimal basis set for natural images.

Despite its superior performance, we would like to highlight that its encoding complexity is much higher than BMVC and other CS-based methods. In the next section, we will provide a detailed analysis and comparison of the encoding cost of these compression methods, where we show the BMVC balances between the low-cost encoder and decent decoding quality.

We summarize the decoding results of various methods in Table 1 and Fig. 7. Note that the small peak in the BMVC-E2E PSNR plot (Fig. 7, red) is because the BMVC-E2E decoders have an additional 3D-CNN stage for high Crs. The SSIM metrics of the BMVC pipeline and other methods (in Table 1) follow a similar trend as the PSNR plot. We can see that, at all Cr levels, JPEG2000 performs best, but with the price of a higher encoding cost. BMVC-PnP provides a higher PSNR than BMVC-E2E when Cr ≤ 50, but with a longer running time. Due to the 3D-CNN stages used in the end-to-end decoder, BMVC-E2E performs better than BMVC-PnP when Cr > 50; random DS always performs the worst among these methods. This is expected because unlike BMVC and other methods which compress the raw image, random DS only samples a small proportion of information. Block CS performs consistently similar to BMVC. However, as the block size increases, the dynamic range of the compressed data increases as well. In Sec. 4.4, we show that block CS is sensitive to low-bit quantization. The BMVC pipeline is more consistent against different quantization bits. Selected decoded results at various Crs and from different methods are presented in Fig. 6. The visual quality and quantitative metrics (Table 1) both show the performance of the BMVC pipeline decreases as Cr increases. This is expected, and we note that a Cr above 100 is extremely challenging, not to mention that our Cr does not include entropy coding, which is part of our future work.

4.3. Computation cost: encoder and decoder

A key advantage of the BMVC pipeline and other CS-based methods over transform-based (DCT, DWT) image compression is its ultralow-cost encoder, which makes it perfect on resource-limited mobile platforms. Here we evaluate the computation cost of the encoder from the compression methods discussed above and shown in Table 2. The computation cost is evaluated by the number of required additions and multiplications when encoding a single image of N pixels. Consider the original image of size N_x rows and N_y columns. The total number of pixels is $N = N_x \times N_x$.

For the BMVC pipeline, let N_b denote the number of blocks, which is also the Cr when there is no overlapping between blocks. Given that the modulation binary mask has equal probability being $\{0,1\}$, the encoder locates the $\frac{N}{2}$ pixel readouts according to the look-up table (predefined mask) and performs

Table 2 Computation Cost of Different Compression Methods.

Codec	Encoder Cost	Dynamic Range of Measured Data	Mask/Basis Size	Decoder	Comment
BMVC	# addition: $\frac{(N_b-2)N}{2N_b}$ # multiplication: 0	$\frac{N_b}{2}$	N	PnP: iterative network E2E: feedforward network	Cr = N_b, N_b: # blocks N: # pixels
Random DS	# addition: 0 # multiplication: 0	1	N	PnP: iterative network	Cr = $\frac{N}{N_s}$, N_s: # sampled pixels
Block CS	# addition: $\frac{N}{2}-N_b$ # multiplication: 0	$\frac{N}{2N_b}$	$\frac{MN}{N_b}$	PnP: iterative network	Cr = $\frac{N}{MN_b}$, M: # measurements per block
JPEG2000	# addition: $(N-N_b)\log_2\frac{N}{N_b}$ # multiplication: $N\log_2\frac{N}{N_b}$	1	flexible	Discrete wavelet transform (DWT)	$N_b = \frac{N}{B^2}$, B: block size

additions to reduce to $\frac{N}{N_b}$ entries. Therefore, the BMVC encoder requires $\frac{N}{2}-\frac{N}{N_b}=\frac{(N_b-2)N}{2N_b}$ additions and 0 multiplication. Considering that the pixel values are normalized to [0, 1], we can expect the encoded block to have all its values bounded between [0, $\frac{N_b}{2}$]. The corresponding BMVC decoder cost depends on the number of iterations of the PnP approach and the CNN structure in the E2E approach. The PnP-based decoders (BMVC-PnP, random DS, block CS) use FFDNet with 2D convolutional layers and runs n_{iter} iterations. In all reported FFDNet-PnP approaches (BMVC-PnP, random DS, block CS), we use $n_{\text{iter}} = 60$. The BMVC-E2E decoder has a different network structure depending on the Cr. As mentioned in Sec 3.4, for low Cr, the BMVC-E2E decoder deploys two 2D-CNNs and one 3D-CNN, while for Cr (>50), the BMVC-E2E decoder deploys two 3D-CNNs for better decoding quality.

The random DS approach has the lowest possible encoder complexity, which does not include any calculation but is simply a reading out of a subset of pixel values from the detector. Such a simple encoding pipeline requires no addition or multiplication.

For block CS, let M denote the number of compressed measurements recorded for each block. Assume the sensing matrix is binary with equal probability; the number of required additions per measurement per block is statistically: $\frac{N}{2N_b}-1$; thus the total number of additions for encoding an image of N pixels is: $(\frac{N}{2N_b}-1)\times M\times N_b = M(\frac{N}{2}-N_b)$. Similar to BMVC, no multiplication is needed for block CS encoding. The compressed measurement is expected to be upper-bounded by half the number of pixels in each block, which equals $\frac{N}{2N_b}$.

JPEG2000 compression is based on wavelet transform with a flexible basis size. The encoding process of JPEG2000 consists of color space conversion, color channel downsampling, wavelet transform, quantization, and entropy coding. For a fair comparison, we analyze the most computationally intense bulk part of the encoder: 2D wavelet transforms on small image blocks (Y channel only). When using a block size of B, the total number of blocks is $N_b = \frac{N_h\times N_w}{B^2}=\frac{N}{B^2}$. Each entry of the block after transform requires B^2 multiplications and B^2-1 additions. By implementing a fast algorithm (similar to butterfly-based FFT), the number of required additions and multiplications per block are: $(B^2-1)\times\log_2 B^2$ and $B^2\times\log_2 B^2$. Then the total number of additions and multiplications for the entire image can be computed as $(B^2-1)\times\log_2 B^2\times N_b = (N-N_b)\log_2\frac{N}{N_b}$ and $B^2\times\log_2 B^2\times N_b = N\log_2\frac{N}{N_b}$. The decoder of JPEG2000 reverses its encoding process and employs inverse wavelet transform for a similar computation cost.

Note that JPEG has the encoder cost of JPEG2000 by setting $B = 8$ in Table 2, since JPEG uses 8×8 DCT basis. We also find it challenging to increase the Cr (defined as the ratio of lossless JPEG file size and lossy JPEG file size) of JPEG to above 20, so we only report the quantitative results of JPEG2000 in Fig. 7.

4.4. Robustness to quantization bits

In real-world applications, the encoded data will have to be quantized before sending them to the receiver.

High dynamic range data are prone to be degraded after bit quantization. As shown in the "dynamic range" column in Table 2, our BMVC-encoded blocks range between 0 and $\frac{N_b}{2}$, which is upper-bounded by half the Cr, while in block CS, each entry of the compressed measurement may have a dynamic range up to $\frac{N}{2N_b}$, which is half the number of pixels in each block. Under general BMVC and block CS settings, we expect the BMVC pipeline to provide more consistent decoding performance than block CS due to its lower dynamic range.

To test how robust the BMVC pipeline is to quantization bits, we conducted 8–16 bit quantization to the encoded/compressed blocks before sending them to the two BMVC decoders and the block CS decoder. Decoded results are evaluated by the PSNR metric. Table 3 and Fig. 8 illustrate how BMVC and block CS perform under different quantization bits. As expected, the two BMVC decoders are robust to quantization bits. The difference in PSNR is generally below 0.1 dB, with only one exception for the BMVC-PnP at Cr = 24 (0.163 dB degradation). The BMVC-E2E decoder is even more consistent against quantization bits, with maximum PSNR degradation—only 0.037 dB. Block CS has much more significant deterioration after low-bit quantization. At Cr = 24, block CS performance decreases by 3.59 dB when quantized to 8-bit data. A similar degradation of SSIM in block CS is observed. Specially, block CS at 8-bit quantization has more reconstruction artifacts after the PnP-based decoder. In general, block CS requires the compressed measurements to be at least 12-bit to secure a consistent

Table 3 PSNRs of BMVC-PnP, BMVC-E2E, and Block CS under Different Quantization Bits.

	BMVC-PnP (dB)				BMVC-E2E (dB)				Block CS (dB)			
Bit/Cr	100	80	50	24	100	80	50	24	100	80	50	24
8-bit	25.921	27.274	28.829	33.451	27.785	28.796	29.060	31.023	27.190	27.049	27.361	29.563
10-bit	25.981	27.305	29.882	33.592	27.805	28.813	29.082	31.056	27.634	27.957	29.361	31.931
12-bit	25.981	27.301	29.889	33.611	27.803	28.818	29.087	31.059	27.800	28.123	30.090	33.132
14-bit	25.982	27.303	29.892	33.613	27.802	28.819	29.088	31.060	27.813	28.132	30.099	33.149
16-bit	25.982	27.304	29.879	33.576	27.803	28.818	29.088	31.060	27.814	28.133	30.102	33.151
ΔPSNR↓	0.0606	0.0315	0.0629	0.1627	0.0202	0.0230	0.0282	0.0370	**0.6238**	**1.0844**	**2.7415**	**3.5872**

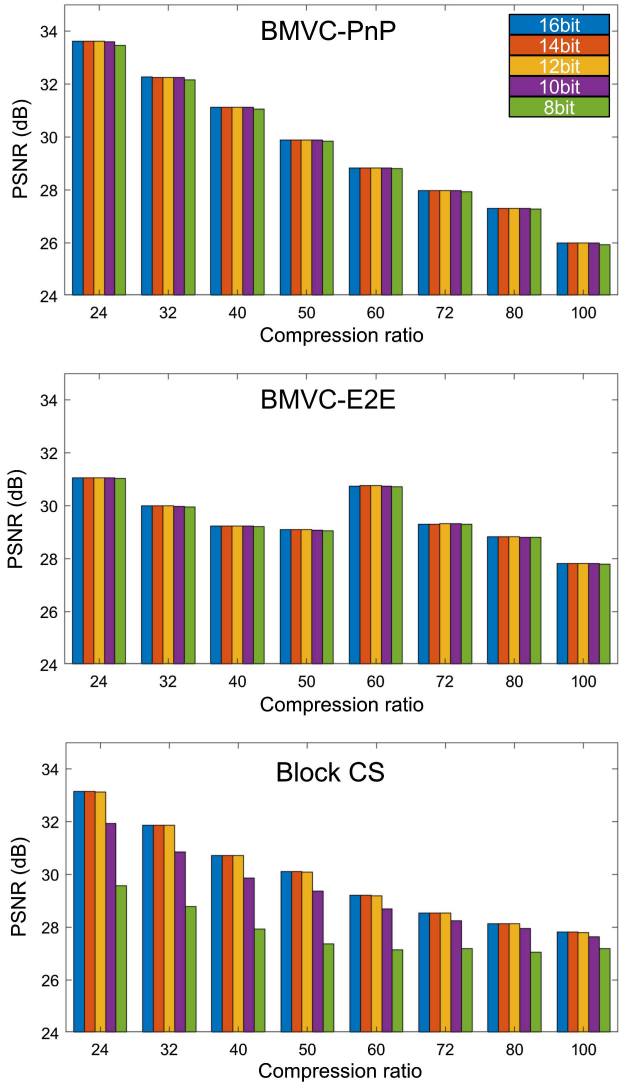

Fig. 8 Evaluation of robustness to quantization bits. BMVC and block CS both show high PSNR performance when the dynamic range of the data is intact. In practice, quantization will affect the codec performance in real-world video signal transmission. The bar plots indicate how the three decoders (BMVC-PnP, BMVC-E2E, and block CS) perform under different quantization bits. BMVC decoders have consistent performance regardless of data quantization. However, block CS has noticeable decreases in PSNR at 10-bit and 8-bit quantization.

Table 4 Ablation Study of BMVC-E2E (Cr ≤ 50).

Decoder Structure	PSNR (dB)	Runtime (ms)
2 × U-Net + 2 × 3D-CNN (RevSCI) + 1 × deeper 3D CNN (RevSCI)	31.720	~255
2 × U-Net + 2 × 3D-CNN (RevSCI)	31.449	~130
2 × U-Net + 1 × 3D-CNN (RevSCI)	**29.228**	~70
2 × U-Net	17.930	~10
1 × U-Net	10.308	~5

performance. That suggests under the same Cr, block CS (12-bit) will cost an extra 50% bandwidth than the BMVC pipeline (8-bit).

4.5. Ablation study of BMVC-E2E

We conduct an ablation study of the BMVC-E2E decoder structure to figure out the optimal design that balances well between decoder runtime and quality. Using Cr = 40 as an example, we trained multiple BMVC-E2E decoders with the structures in Table 4. After training, decoded results on the testing data set are evaluated by PSNR and reported in Table 4. As we employ more stages in the BMVC-E2E decoder, the decoding quality indeed improves, but at the cost of longer decoding runtime. Significant improvement is observed when adding the first stage of 3D-CNN, as it is capable of efficiently gathering information from the stack of nonlocal blocks. However, 3D-CNNs largely slow down the decoder runtime. Finally, we decided to aim at >10 fps real-time decoding for low Crs and >4 fps for high Crs. That gives us the BMVC-E2E decoder structures discussed in Sec. 3.4, which balances between decoding quality and runtime. Notably, we also empirically found that adding more 2D-CNNs does not provide on-par results as their 3D-CNN counterparts, and high Crs (Cr > 50) benefits from using only 3D-CNNs.

5. Discussion and Conclusions

In recent years, DL-based codecs have made significant advancements. The delay cognizant video coding (DCVC) framework, in particular, has shown promising results. However, when comparing it with BMVC, there are notable differences in their application scenarios and respective strengths and weaknesses. DCVC represents a sophisticated conditional coding

framework that leverages temporal context features as conditional inputs to enhance both encoding and decoding processes. By utilizing these temporal context features, DCVC effectively taps into the potential of conditional coding to improve encoding efficiency and reconstruction quality. Specifically, the inclusion of high-dimensional features helps in preserving high-frequency details in reconstructed videos, leading to enhanced visual fidelity. However, the integration of high-dimensional features and conditional coding mechanisms may impose higher computational demands. More specifically, for the encoding process, the latest DCVC work, DCVC-FM, first employs an optical flow network to estimate the motion vector. Then the motion vector is used to extract the temporal context for both encoding and decoding. In contrast, BMVC adopts a frame-independent pipeline with the primary objective of minimizing computation costs on encoder hardware. BMVC's encoding process can be efficiently implemented as additions of pixel readouts according to a predefined look-up table, by which BMVC can achieve robust performance across diverse video content while maintaining real-time encoding capabilities, crucial for applications with stringent computational constraints.

To conclude, we have proposed a brand-new BMVC codec that has an ultralow-cost encoder, which is perfect for resource-limited platforms such as robotics and drones. We have also developed two BMVC decoders, based on PnP optimization and E2E neural networks. The BMVC-PnP decoder is an iterative algorithm that has proved convergence. It is also flexible and robust to different compression ratios. The BMVC-E2E decoder has an unrolled structure of the PnP approach with very few stages. Its feed-forward nature enables real-time decoding of 1080p image sequences on a single GPU, achieving >14 fps for Crs under 60 and 4 fps for higher Crs.

Unlike traditional image and video compression algorithms that embed prior knowledge in the encoding process via optimal basis selection, the BMVC pipeline takes a different design philosophy. We only fully utilize the prior knowledge in the decoding process while keeping the encoding process as simple as possible. As a result, we can keep the computation cost of the encoder as low as possible so that we can save power, computation, and bandwidth on resource-limited platforms. On the other hand, the decoding process of BMVC is ill-posed so that it requires strong prior knowledge about natural images to reliably retrieve the desired frames. This is achieved via either a PnP framework with an image-denoising CNN or an E2E neural network trained on massive data sets.

One aspect we have not explored is using temporal information across frames to further compress the data stream. It is true that state-of-the-art video codecs take advantage of the temporal redundancy of video data. We chose to make the BMVC pipeline a frame-independent codec for three reasons: low complexity, low latency, and constant bandwidth. To compute the differential signal between frames will cost extra power consumption, which is not in line with our original purpose of developing BMVC. Another issue raised by processing a bunch of frames is that it will increase the latency of the E2E pipeline. The third advantage of the frame-independent BMVC pipeline is its constant bit rate, which differs from the content-dependent bandwidth of MPEG-based video codecs.

Recall again that BMVC is not designed to replace the current codec standard, but to provide an alternative option for platforms under extreme power/computation constraints. In fact, we have managed to implement the BMVC encoding pipeline on a drone platform with a simple Raspberry Pi development board. Future work will be around building the E2E BMVC pipeline to achieve optimal performance (both speed and quality) and minimal latency. With this ultralow-cost encoder design and two options for BMVC decoders, we envision the BMVC pipeline could be a revolutionary technique for video signal compression and transmission on robotic platforms.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62271414), the Zhejiang Provincial Outstanding Youth Science Foundation (No. LR23F010001), the Zhejiang "Pioneer" and "Leading Goose" R&D Program (Nos. 2024SDXHDX0006 and 2024C03182), the Key Project of Westlake Institute for Optoelectronics (No. 2023GD007), and the 2023 International Sci-tech Cooperation Projects under the purview of the "Innovation Yongjiang 2035" Key R&D Program (No. 2024Z126). The works of Yujia Xue and Waleed Tahir were conducted when they were summer interns at Nokia Bell Labs.

References

1. D. Le Gall, "MPEG: a video compression standard for multimedia applications," *Commun. ACM* **34**, 46 (1991).
2. R. Puri *et al.*, "Distributed video coding in wireless sensor networks," *IEEE Signal Process Mag.* **23**, 94 (2006).
3. W. Shi *et al.*, "Edge computing: vision and challenges," *IEEE Internet Things J.* **3**, 637 (2016).
4. S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: fixed-point convergence and applications," *IEEE Trans. Comput. Imaging* **3**, 84 (2017).
5. E. K. Ryu *et al.*, "Plug and-play methods provably converge with properly trained denoisers," in *International Conference on Machine Learning (ICML)* (2019), p. 5546.
6. S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *IEEE Global Conference on Signal and Information Processing* (2013), p. 945.
7. S. Sreehari *et al.*, "Plug-and-play priors for bright field electron tomography and sparse interpolation," *IEEE Trans. Comput. Imaging* **2**, 408 (2016).
8. X. Yuan *et al.*, "Plug-and-play algorithms for large-scale snapshot compressive imaging," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), p. 1447.
9. E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory* **52**, 489 (2006).
10. D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory* **52**, 1289 (2006).
11. X. Yuan, D. J. Brady, and A. K. Katsaggelos, "Snapshot compressive imaging: theory, algorithms, and applications," *IEEE Signal Process Mag.* **38**, 65 (2021).
12. P. Llull *et al.*, "Coded aperture compressive temporal imaging," *Opt. Express* **21**, 10526 (2013).
13. X. Yuan *et al.*, "Low-cost compressive sensing for color video and depth," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), p. 3318.
14. M. Qiao *et al.*, "Deep learning for video compressive sensing," *APL Photonics* **5**, 030801 (2020).
15. A. Wagadarikar *et al.*, "Single disperser design for coded aperture snapshot spectral imaging," *Appl. Opt.* **47**, B44 (2008).
16. Z. Meng, J. Ma, and X. Yuan, "End-to-end low cost compressive spectral imaging with spatial-spectral self-attention," in *European Conference on Computer Vision (ECCV)* (2020), p. 187.

17. M. Qiao, X. Liu, and X. Yuan, "Snapshot spatial-temporal compressive imaging," *Opt. Lett.* **45**, 1659 (2020).
18. P. Llull *et al.*, "Image translation for single shot focal tomography," *Optica* **2**, 822 (2015).
19. T.-H. Tsai *et al.*, "Spectral-temporal compressive imaging," *Opt. Lett.* **40**, 4054 (2015).
20. Y. Sun, X. Yuan, and S. Pang, "High-speed compressive range imaging based on active illumination," *Opt. Express* **24**, 22836 (2016).
21. M. Qiao *et al.*, "Snapshot coherence tomographic imaging," *IEEE Trans. Comput. Imaging* **7**, 624 (2021).
22. Y. Sun, X. Yuan, and S. Pang, "Compressive high-speed stereo imaging," *Opt. Express* **25**, 18182 (2017).
23. M. F. Duarte *et al.*, "Single-pixel imaging via compressive sampling," *IEEE Signal Process Mag.* **25**, 83 (2008).
24. A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (2005), p. 60.
25. K. Dabov *et al.*, "Image denoising with block-matching and 3D filtering," *Proc. SPIE* **6064**, 606414 (2006).
26. Y. Lai *et al.*, "Single-shot ultraviolet compressed ultrafast photography," *Laser Photonics Rev.* **14**, 2000122 (2020).
27. J. Bioucas-Dias and M. Figueiredo, "A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Trans. Image Process.* **16**, 2992 (2007).
28. X. Yuan, "Generalized alternating projection based total variation minimization for compressive sensing," in *IEEE International Conference on Image Processing (ICIP)* (2016), p. 2539.
29. Y. Liu *et al.*, "Rank minimization for snapshot compressive imaging," *IEEE Trans. Pattern Anal. Mach. Intell.* **41**, 2990 (2019).
30. X. Miao *et al.*, "λ-net: Reconstruct hyperspectral images from a snapshot measurement," in *IEEE/CVF Conference on Computer Vision (ICCV)* (2019), p. 4059.
31. Z. Cheng *et al.*, "Memory-efficient network for large-scale video compressive sensing," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), p. 16246.
32. Y. Xue *et al.*, "Reliable deep-learning-based phase imaging with uncertainty quantification," *Optica* **6**, 618 (2019).
33. Y. Li, Y. Xue, and L. Tian, "Deep speckle correlation: a deep learning approach toward scalable imaging through scattering media," *Optica* **5**, 1181 (2018).
34. J. Ma *et al.*, "Deep tensor ADMM-Net for snapshot compressive imaging," in *IEEE/CVF Conference on Computer Vision (ICCV)* (2019), p. 10223.
35. L. Wang *et al.*, "Hyperspectral image reconstruction using a deep spatial-spectral prior," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), p. 8024.
36. Y. Li *et al.*, "End-to-end video compressive sensing using Anderson-accelerated unrolled networks," in *IEEE International Conference on Computational Photography (ICCP)* (2020), p. 1.
37. T. Huang *et al.*, "Deep Gaussian scale mixture prior for spectral compressive imaging," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), p. 16216.
38. K. Greger and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)* (2010), p. 399.
39. Y. Yang *et al.*, "Deep ADMM-Net for compressive sensing MRI," in *Advances in Neural Information Processing Systems*, Vol. **29** (2016), p. 10.
40. C. A. Metzler, A. Mousavi, and R. G. Baraniuk, "Learned D-AMP: principled neural network based compressive image recovery," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)* (2017), p. 1770.
41. S. Zheng *et al.*, "Deep plug-and-play priors for spectral snapshot compressive imaging," *Photonics Res.* **9**, B18 (2021).
42. X. Yuan *et al.*, "Plug-and-play algorithms for video snapshot compressive imaging," *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7093 (2021).
43. K. Zhang, W. Zuo, and L. Zhang, "FFDNet: toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.* **27**, 4608 (2018).
44. M. Kellman *et al.*, "Memory-efficient learning for large-scale computational imaging," *IEEE Trans. Comput. Imaging* **6**, 1403 (2020).
45. S. Jalali and X. Yuan, "Snapshot compressed sensing: performance bounds and algorithms," *IEEE Trans. Inf. Theory* **65**, 8005 (2019).
46. X. Liao, H. Li, and L. Carin, "Generalized alternating projection for weighted-2.1 minimization with applications to model-based compressive sensing," *SIAM J. Imag. Sci.* **7**, 797 (2014).
47. J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: model-based inspiration of novel deep architectures," arXiv:1409.2574 (2014).
48. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Vol. **9351** of Lecture Notes in Computer Science (Springer, 2015), p. 234.
49. S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), p. 3883.
50. F. Perazzi *et al.*, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 724.
51. Z. Wang *et al.*, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.* **13**, 600 (2004).
52. A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120 fps 4k sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference* (2020), p. 297.
53. Z. Zha *et al.*, "Image restoration via reconciliation of group sparsity and low-rank models," *IEEE Trans. Image Process.* **30**, 5223 (2021).
54. X. Yuan and R. Haimi-Cohen, "Image compression based on compressive sensing: End-to-end comparison with jpeg," *IEEE Trans. Multimedia* **22**, 2889 (2020).
55. J. Zhang and B. Ghanem, "ISTA-Net: interpretable optimization-inspired deep network for image compressive sensing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), p. 1828.
56. X. Yuan and Y. Pu, "Parallel lensless compressive imaging via deep convolutional neural networks," *Opt. Express* **26**, 1962 (2018).
57. Lu Gan, "Block compressed sensing of natural images," in *15th International Conference on Digital Signal Processing* (2007), p. 403.
58. D. You *et al.*, "Coast: controllable arbitrary-sampling network for compressive sensing," *IEEE Trans. Image Process.* **30**, 6066 (2021).
59. Z. Zha *et al.*, "Triply complementary priors for image restoration," *IEEE Trans. Image Process.* **30**, 5819 (2021).