# Mononizing Binocular Videos

WENBO HU, MENGHAN XIA, CHI-WING FU, and TIEN-TSIN WONG, The Chinese University of Hong Kong

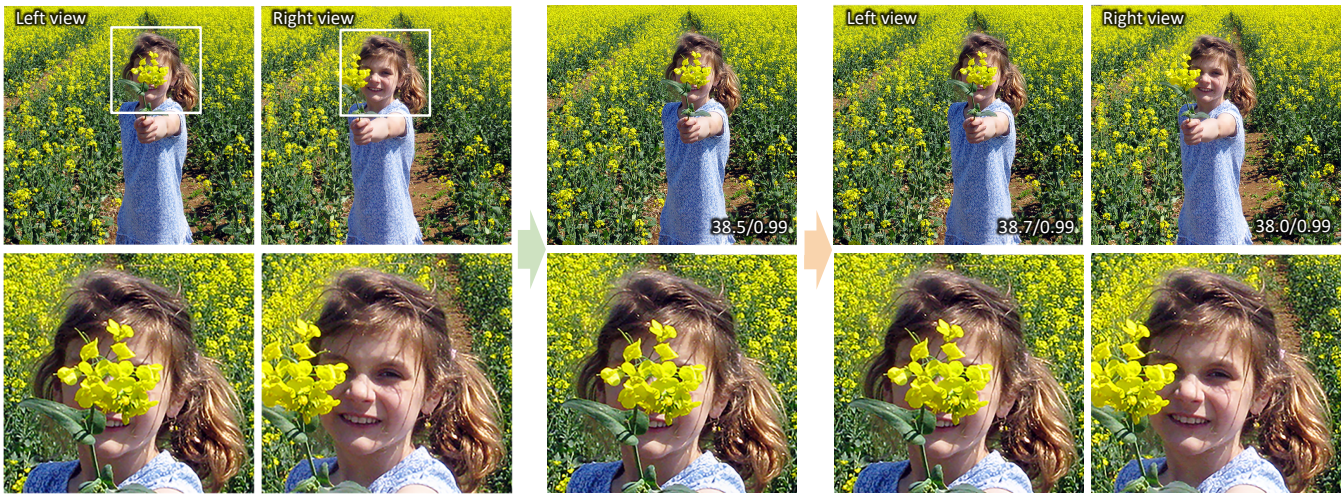



Fig. 1. Our method can effectively *mono-nize* a binocular video into a monocular video with the stereo information encoded in a nearly-imperceptible form. Note, the face of the little girl in the input right view (left); we can fuse it with the left view and encode (hide) it in the mononized frame (middle). Though the face is not observable in the mononized frame, we can restore it back in the restored binocular frame (right). We show PSNR/SSIM at the bottom of each result.

This paper presents the idea of *mono-nizing* binocular videos and a framework to effectively realize it. Mono-nize means we purposely convert a binocular video into a regular monocular video with the stereo information implicitly encoded in a visual but nearly-imperceptible form. Hence, we can impartially distribute and show the mononized video as an ordinary monocular video. Unlike ordinary monocular videos, we can restore from it the original binocular video and show it on a stereoscopic display. To start, we formulate an encoding-and-decoding framework with the pyramidal deformable fusion module to exploit long-range correspondences between the left and right views, a quantization layer to suppress the restoring artifacts, and the compression noise simulation module to resist the compression noise introduced by modern video codecs. Our framework is self-supervised, as we articulate our objective function with loss terms defined on the input: a monocular term for creating the mononized video, an invertibility term for restoring the original video, and a temporal term for frame-to-frame coherence. Further, we conducted extensive experiments to evaluate our generated mononized videos and restored binocular videos for diverse types of images and 3D movies. Quantitative results on both standard metrics and user perception studies show the effectiveness of our method.

CCS Concepts: • **Computing methodologies** → **Computer graphics**.

Authors' address: Wenbo Hu; Menghan Xia; Chi-Wing Fu; Tien-Tsin Wong, The Chinese University of Hong Kong, Hong Kong, [wbhu,mhxia,cwfu,ttwong]@cse.cuhk.edu.hk.


0730-0301/2020/12-ART228 $15.00
https://doi.org/10.1145/3414685.3417764

Additional Key Words and Phrases: Binocular video, machine learning

**ACM Reference Format:**
Wenbo Hu, Menghan Xia, Chi-Wing Fu, and Tien-Tsin Wong. 2020. Mononizing Binocular Videos. *ACM Trans. Graph.* 39, 6, Article 228 (December 2020), 16 pages. https://doi.org/10.1145/3414685.3417764

## 1 INTRODUCTION

While multi-camera smartphones become popular in the market, single-view platforms remain dominant. To migrate from single- to multi-view, re-design and re-implementation of existing software and hardware platforms are usually unavoidable, and may obsolete the existing single-view platforms. Hence, a backward-compatible solution is crucial, as demonstrated in the successful migration from black&white to color TV broadcasting during the 50's to 60's.

In this paper, we present a fully backward-compatible solution that is independent of the video coding standard and requires zero additional upgrade/installation on monocular TVs to cope with the stereoscopic data. To achieve the goal, we propose a brand new approach to *"mono-nize"* (convert) conventional binocular (stereoscopic) images/videos to monocular ones. The generated monocular images/videos, which we call *mononized images/videos*, look visually the same as one of the two views (say, the left view, without loss of generality) in the given binocular images/videos, and can be treated (stored, distributed, and displayed) as ordinary monocular images/videos. The only difference is that we can restore their binocular counterparts from them, whenever necessary. The mononization and restoration enable a fully backward-compatible solution for the migration from single- to multi-view (Figure 2).

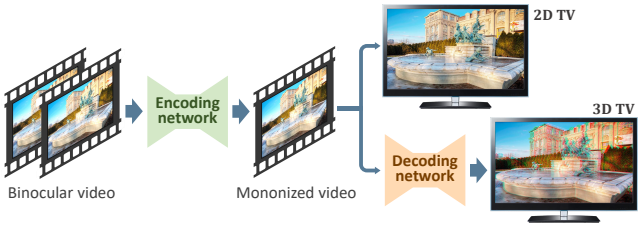



Fig. 2. Given a binocular video, our framework produces a mononized video that is visually no different from an ordinary monocular video, and can be distributed and displayed on conventional monocular-video platforms. If a 3D display is available, we can restore the original binocular video from the mononized video and provide stereo viewing on the video.

A possible solution is to just drop one of the two views, then use methods such as [Calagari et al. 2017; Leimkühler et al. 2018; Niklaus et al. 2019; Shih et al. 2020; Xie et al. 2016] to infer or estimate the other view from the remaining one. However, such approach cannot accurately predict the dropped view, due to the loss of occlusion and depth information. For instance, the face of the little girl shown in Figure 1 is mostly occluded in the left view; if we drop the right view, it will be hard to recover her face in the right view solely from the left one. Rather, we aim to implicitly encode the other view in a *visual but nearly-imperceptible form* in the mononized frame, such that we can restore from it a high-quality binocular frame (Figure 1). The key problem is how to achieve such an encoding.

Instead of handcrafting the encoding process, we propose to embed the encoding system via a convolutional neural network (CNN) [Goodfellow et al. 2016]. Our framework consists of (i) an *encoding neural network* to convert the input binocular video into a mononized video and (ii) a *decoding neural network* to restore the binocular video (Figure 2). This formulation enables self-supervised learning and bypasses the need of preparing manually-labeled training data, which is a common burden to many deep learning methods.

Unfortunately, it is hard for conventional CNNs to exploit correspondences between the left and right views, since the large disparities between views may exceed the spatial transformation capability of conventional CNNs [Jaderberg et al. 2015]. To overcome this, we present the pyramidal deformable fusion (PDF) module to explore long-range correspondences between the left and right views. Also, we adopt a quantization layer to suppress the artifacts caused by quantization errors, and formulate the compression noise simulation (CNS) module to resist the compression perturbation that could be introduced by the video codecs. Further, we design an objective function with three loss terms to train the networks: *monocular loss* to ensure the mononized video looks like the left view of the input video, *invertibility loss* to ensure the restored binocular video looks like the original, and *temporal loss* to ensure the temporal coherence in both the mononized videos and restored binocular videos.

To evaluate our method, we employed a collection of binocular images and 3D movies of various scene categories, and conducted extensive experiments, including a qualitative evaluation on the visual quality of our results, i.e., the mononized and restored binocular images/videos (Section 7.1), quantitative comparisons with methods [Baluja 2017; Niklaus et al. 2019; Xia et al. 2018] related to our application (Section 7.2), a quantitative evaluation of our results on frame quality and temporal coherence (Section 7.3), a quantitative evaluation on the compatibility of our method with common video codecs (Section 7.4); and a user study to evaluate the perceptual performance of our method (Section 7.5). All evaluated metrics confirm the effectiveness of our method to produce high-quality mononized and restored binocular images/videos. Also, the user study shows that our generated mononized videos and restored binocular videos look no different from their original counterparts. Further, the experimental results show that our mononized videos are friendly with common video codecs. In fact, encoding our mononized videos with standard codecs outperforms existing side-by-side and multi-view encoding methods at low bit-rate, while it achieves comparable compression performance at high bit-rate. This makes our approach a favorable alternative when storage is a concern.

Our contributions are summarized below.

- We offer a backward-compatible solution for distributing and storing binocular videos as monocular ones, that are fully compatible to common monocular platforms.
- We formulate this conversion-and-restoration problem as an encoding-and-decoding process embedded in deep neural networks that are trained in a self-supervised manner.
- We propose the pyramidal deformable fusion module to exploit the long-range correspondences between the left and right views, design a compression noise simulation module and adopt a quantization layer to resist noise in real-world video compression.
- We offer a favorable compression alternative, as our method outperforms existing solution at low bit-rate, and achieves a comparable performance at high bit-rate.

## 2 RELATED WORK

### 2.1 Stereo Image/video Synthesis

Novel view synthesis from a single image, e.g., [Leimkühler et al. 2018; Niklaus et al. 2019], usually starts by estimating a depth map, then performing depth-based image rendering. Deep neural networks have shown remarkable improvements on single-image depth estimation [Atapour-Abarghouei and Breckon 2018; Eigen et al. 2014; Fácil et al. 2019; Li et al. 2019a; Luo et al. 2018]. However, it remains very challenging to obtain accurate depth maps for general scenes. Several works [Cun et al. 2018; Liu et al. 2018; Xie et al. 2016] propose to integrate depth estimation and view synthesis into an end-to-end deep neural network. Recent works also propose to explicitly inpaint the occluded regions in image [Shih et al. 2020] or point cloud domain [Niklaus et al. 2019]. However, ensuring plausible results in the inpainted occluded regions is still very hard. Instead of estimating the novel views, this work solves a very different problem of *restoring* the stereo contents in the mononized video, in which the stereo information is implicitly encoded.

Besides, some works focus on retargeting stereo or multi-view images with preferred properties, e.g., stereo magnification [Zhou et al. 2018], novel view synthesis from multiple images [Flynn et al. 2016; Kellnhofer et al. 2017; Lombardi et al. 2019; Mildenhall et al. 2020; Srinivasan et al. 2019; Xu et al. 2019], and disparity manipulation [Didyk et al. 2011, 2012; Kellnhofer et al. 2016; Lang et al.

2010]. Recently, some works [Fukiage et al. 2017; Scher et al. 2013] point out that when a conventional stereoscopic display is viewed without stereo glasses, image blurs, or 'ghosts,' are visible due to the fusion of the stereo image pairs, which makes the stereoscopic display backward incompatibility. Also, they propose to synthesize ghost-free stereoscopic image pairs that can minimize the ghosts when viewed without stereo glasses and provide stereo viewing when stereo glasses is available. Note that ghost-free stereo images/videos are still binocular in nature. In contrast, our mononized images/videos are monocular in nature, and yet embed the stereo information in a nearly-imperceptible form.

### 2.2 Reversible Image Conversion

The reversible property has been explored in the stenography methods [Baluja 2017; Wang et al. 2019a; Wengrowski and Dana 2019; Zhu et al. 2018], which conceal secret information (or an image) within a reversible container image, from which the secret information can be recovered. For example, Baluja [2017] presents a method to hide an image within another one. However, the container image and secret information/image are usually unrelated, so it is hard to generate artifacts-free results for both hiding and recovering.

On the other hand, the reversible property is explored in some image processing procedures. For example, [Xia et al. 2018] formulate a neural network to generate a reversible grayscale from a color image, where colors can be restored from the grayscale image; [Li et al. 2019b] adopt a neural network to generate downsampled images with compactly-encoded higher-resolution details, and show that the high-resolution counterparts can be restored with a super-resolution neural network. Our work belongs to the category of reversible networks. Among them, [Xia et al. 2018] is a generic one. However, we cannot directly apply it to realize our application. Its network is designed for still images. Also, it cannot effectively hide a view (right view) in the other one (left view), since the large disparities between views may exceed the spatial transformation capability of conventional CNNs. Moreover, we need to handle binocular videos and account for the temporal coherence. Please refer to Section 7.2 for comparison with other methods.

### 2.3 Binocular Video Encoding

To store and distribute a binocular video using existing video encoding standards, one can simply concatenate each pair of left and right frames top-down or side-by-side as a single image [Vetro 2010], and regard the combined frames as a regular monocular video for encoding using existing video codecs. Doing so not only ignores the binocular coherence in the encoding, but also wastes the computation, if we simply want to playback the video on conventional monocular displays. It is because we first have to decode the whole side-by-side video before obtaining the individual frames and dropping the other half of the frames; we cannot skip the decoding of the other half, due to the nature of video encoding. Having said that, we need a special piece of software/hardware to handle the process, which is incompatible with regular monocular displays.

Another stream of approaches is to design multi-view extensions for existing monocular video codecs, e.g., the MVC extension[1] for H.264/MPEG4-AVC uses 2D plus Delta to encode binocular videos and the MV-HEVC[2] extension for H.265/HEVC supports the encoding of multiple views with inter-layer prediction. More multi-view extensions for H.264/MPEG4-AVC and for H.265/HEVC can be found in [Vetro et al. 2011] and [Tech et al. 2016], respectively. However, multi-view extensions work for specific codecs: each time a new codec comes out, the multi-view extension has to be re-designed and re-developed. Recently, [Mallik and Akbari 2016] and [Lai et al. 2017] present new multi-view extensions for H.265/HEVC that use 4D wavelets and frame interleaving to enhance the coding efficiency. Unfortunately, they are incompatible with diverse existing monocular displays. Different from the above methods, our framework does not rely on any specific video codec. We can readily restore the binocular video from the mononized one via a decoding network on top of the video codec, so our framework is fully compatible with the existing codecs, as demonstrated in Section 7.4.

## 3 OVERVIEW

*Image Mononization.* Before we present the full framework (Figure 3) for producing *mononized videos*, we first introduce our framework for producing *mononized images* to better state the insight in our approach and to give the notations.

Overall, our framework has two parts: *an encoding neural network* $E$ and *a decoding neural network* $D$, as shown in the middle "Time $t$" block of Figure 3 without the recurrent connections. Given a stereo image pair $\{\mathbf{I}_L, \mathbf{I}_R\}$ as input, the encoding network generates mononized image $\mathbf{O}_M$ that looks like $\mathbf{I}_L$ (without loss of generality), and at the same time, embeds $\mathbf{I}_R$ as nearly-imperceptible information in $\mathbf{O}_M$. Inside the decoding network, we first use a pair of feature extractors with shared weights to simultaneously extract *pyramidal left feature* $\mathbf{P}_L$ and *pyramidal right feature* $\mathbf{P}_R$, which are then fused together by the *pyramidal deformable fusion (PDF)* module to produce the *pyramidal mononized feature* $\mathbf{P}_M$. Finally, we feed $\mathbf{P}_M$ into the reconstructor to produce the mononized image $\mathbf{O}_M$.

On the other hand, the decoding network restores a stereo image pair $\{\mathbf{O}_L, \mathbf{O}_R\}$ from $\mathbf{O}_M$, such that $\{\mathbf{O}_L, \mathbf{O}_R\}$ look like $\{\mathbf{I}_L, \mathbf{I}_R\}$, respectively. Inside the decoding network, we first extract *pyramidal mononized feature* $\widetilde{\mathbf{P}_M}$, and transform it to simultaneously produce *pyramidal left feature* $\widetilde{\mathbf{P}_L}$ and *pyramidal right feature* $\widetilde{\mathbf{P}_R}$ by another *PDF* module. Finally, we feed $\widetilde{\mathbf{P}_L}$ and $\widetilde{\mathbf{P}_R}$ into a pair of reconstructors with shared weights to generate $\mathbf{O}_L$ and $\mathbf{O}_R$, respectively. Note, we drop superscript $t$ in the notations, since we now discuss the framework for image mononization. Later, we will put superscript $t$ back to the notations when we discuss video mononization. Mathematically, $E$ and $D$ are defined, respectively, as

$$\mathbf{O}_M = E(\{\mathbf{I}_L, \mathbf{I}_R\}) \quad (1)$$

$$\text{and } \{\mathbf{O}_L, \mathbf{O}_R\} = D(\mathbf{O}_M) = D(E(\{\mathbf{I}_L, \mathbf{I}_R\})). \quad (2)$$

To train $E$ and $D$, we define the *monocular loss* $\mathcal{L}_M$ to ensure $\mathbf{O}_M$ looks like $\mathbf{I}_L$ and the *invertibility loss* $\mathcal{L}_I$ to ensure $\{\mathbf{O}_L, \mathbf{O}_R\}$ look like $\{\mathbf{I}_L, \mathbf{I}_R\}$, respectively. Altogether, we jointly train the two networks to produce a pair of compatible encoder and decoder.

[1] https://en.wikipedia.org/wiki/Multiview_Video_Coding

[2] https://hevc.hhi.fraunhofer.de/mvhevc

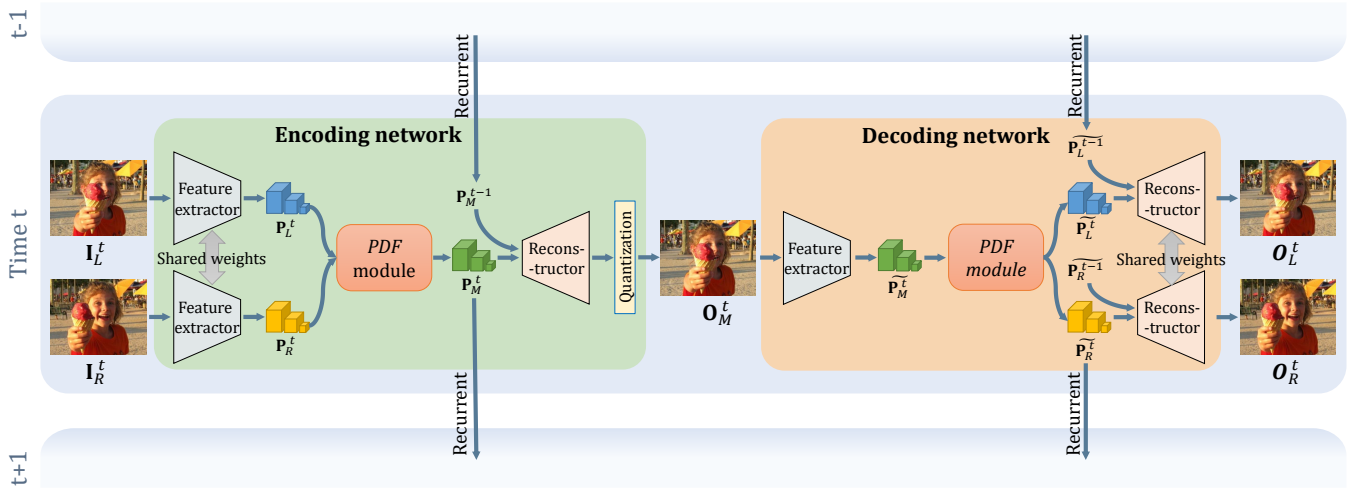



Fig. 3. Overview of our framework for producing mononized videos. Our framework has two parts: (i) an encoding network that produces mononized frame $\mathbf{O}_M^t$ at time $t$ from input binocular image pair $\{\mathbf{I}_L^t, \mathbf{I}_R^t\}$ and pyramidal mononized feature $\mathbf{P}_M^{t-1}$ from the previous time frame; and (ii) an decoding network that restores a binocular image pair $\{\mathbf{O}_L^t, \mathbf{O}_R^t\}$ from $\mathbf{O}_M^t$ and binocular pyramid feature pair $\{\widetilde{\mathbf{P}_L^{t-1}}, \widetilde{\mathbf{P}_R^{t-1}}\}$ from the previous time frame. Note that $\mathbf{P}_L^t$, $\mathbf{P}_R^t$, and $\mathbf{P}_M^t$ denote the pyramidal left, right, and mononized features, respectively; and the pyramidal deformable fusion (PDF) module is proposed to exploit the long-range correspondences between the left and right views to improve the encoding efficiency.

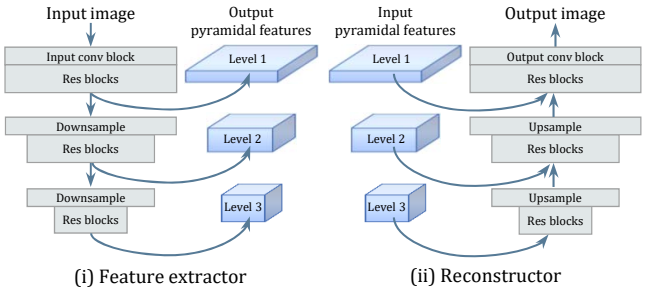



Fig. 4. The feature extractor (i) hierarchically extracts features in three levels, whereas the reconstructor (ii) is symmetric with the feature extractor and trainable skip connections are adopted.

*Video Mononization.* Independently processing video frames at each time $t$ may lead to temporal incoherence. So, we recurrently feed $\mathbf{P}_M^{t-1}$ and $\{\widetilde{\mathbf{P}_L^{t-1}}, \widetilde{\mathbf{P}_R^{t-1}}\}$ from previous time $t-1$ into the networks, and formulate the *temporal loss* $\mathcal{L}_T$ to drive the network to produce $\mathbf{O}_M^t$, $\mathbf{O}_L^t$, and $\mathbf{O}_R^t$ that are temporally coherent with $\mathbf{O}_M^{t-1}$, $\mathbf{O}_L^{t-1}$, and $\mathbf{O}_R^{t-1}$, respectively. Lastly, we put together the three terms to formulate the overall loss function $\mathcal{L}$, and train the whole framework end-to-end in a self-supervised manner:

$$\mathcal{L} = \mathcal{L}_M + \lambda_1 \mathcal{L}_I + \lambda_2 \mathcal{L}_T \,, \tag{3}$$

where $\lambda_1$ and $\lambda_2$ are weights. The details on the network architecture, the design of the loss terms, and the training scheme are presented in Sections 4, 5, and 6, respectively.

## 4 NETWORK ARCHITECTURE

### 4.1 Network Backbone

The feature extractors in both the encoding and decoding networks have the same architecture (Figure 4 (i)), which is a variant of ResNet [He et al. 2016]. We remove the batch normalization [Ioffe and Szegedy 2015] from the original residual blocks [He et al. 2016] as done in [Nah et al. 2017], since we found it performs better empirically. Also, to better abstract the features from low to high levels, we adopt a convolution with a stride of two to realize the downsampling instead of using max or average pooling. We hierarchically extract information from the input image in three levels to form pyramidal features. Intuitively, the feature extractors (left part in Figure 3) are mapping functions that embed the left and right images to feature space, so we share the weights of the two feature extractors in the encoding network to maintain their mapping uniformity.

The reconstructors in the encoding and decoding networks also have the same architecture (Figure 4 (ii)), which is symmetric with the feature extractor architecture. Specifically, the "upsample" block is achieved by a bilinear interpolation followed by a convolution. Except for the features in the third level, features in all the other levels are fed into the reconstructor using skip connection, then linearly combined with the upsampled features from a higher level via some trainable coefficients. Similarly, we share the weights of the reconstructors for the left and right views in the decoding network (right part in Figure 3). When extending this encoding-and-decoding framework to mononize binocular videos, we further feed the corresponding pyramid features from the previous time frame ($\mathbf{P}_M^{t-1}$, $\widetilde{\mathbf{P}_L^{t-1}}$, and $\widetilde{\mathbf{P}_R^{t-1}}$) into the corresponding reconstructors to form a recurrent neural network (Figure 3). Please refer to Supplemental material Section 1 for the detailed network architecture.

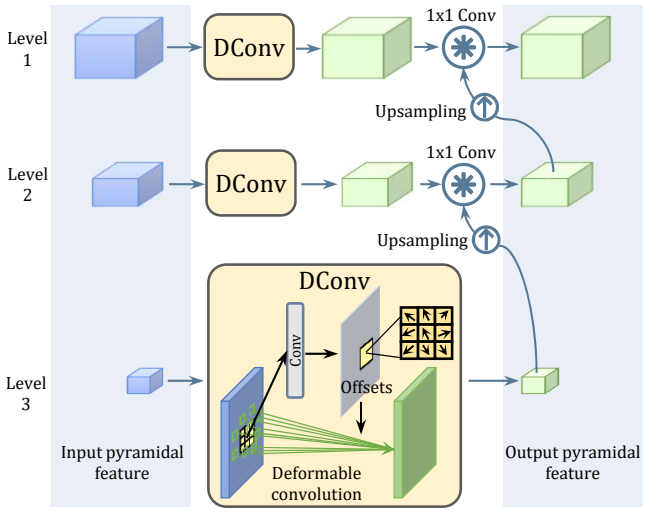



Fig. 5. The Pyramidal Deformable Fusion (PDF) module. Given the input pyramidal feature, the PDF module exploits the long-range correspondences and aggregates context information to generate the output pyramidal feature. The upsampling is achieved by a bilinear interpolation.

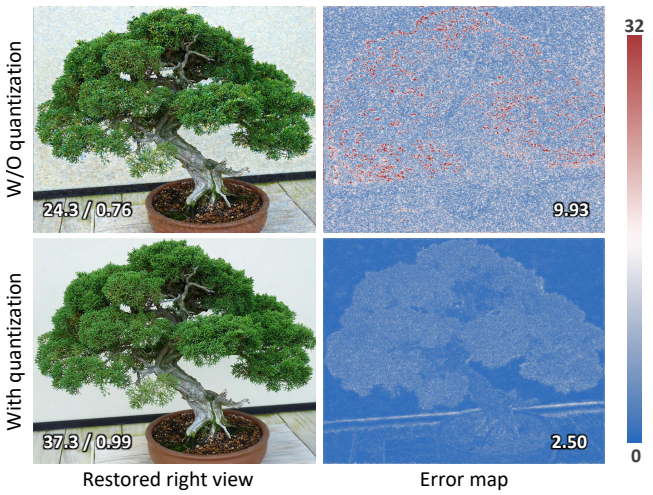



Fig. 6. Restored right views produced at the test phase using our network models trained without (top row) and with (bottom row) the quantization layer. Corresponding error maps (compared with the ground truth) are shown on the right-hand side, PSNR/SSIM (left column) and mean absolute error (right column) are shown inside the restored images.

## 4.2 Pyramidal Deformable Fusion (PDF) Module

First of all, the mononized pyramid feature $\mathbf{P}_M^t$ should contain information of both the left and right views (left part in Figure 3), such that the mononized frame $\mathbf{O}_M^t$ reconstructed from it can inherit the information of both the left and right views and the decoding network can later extract pyramidal feature $\widetilde{\mathbf{P}_M^t}$ and further reconstruct the left and right views (right part in Figure 3).

Hence, we should first fuse pyramidal features $\mathbf{P}_L^t$ and $\mathbf{P}_R^t$ from the left and right views to form $\mathbf{P}_M^t$. However, $\mathbf{P}_L^t$ and $\mathbf{P}_R^t$ may not align well due to the disparity between the left and right views. In practice, the disparity can be as large as ~300 pixels for objects that are close to the camera. For such cases, it will be hard for the CNNs to figure out the long-range correspondences, due to the limited spatial transformation capability in conventional CNNs [Jaderberg et al. 2015]. Similar challenge also exists in several image recognition tasks, e.g., semantic segmentation and object detection, in which the geometry deformation could lead to performance degeneration. To meet this challenge, [Dai et al. 2017] propose a deformable convolution operator to augment the spatial sampling locations with additional offsets and learn the offsets for semantic segmentation and object detection. Later, the deformable convolution v2 [Zhu et al. 2019] further extends the operator to improve the performance. Based on deformable convolution v2, we formulate the *Pyramidal Deformable Fusion (PDF)* module to examine the transformation between left and right views and exploit the long-range correspondences between views in a hierarchical manner.

After we concatenate the left and right pyramidal features along the channel dimension at each level, our *PDF* module implicitly explores the long-range correspondences among the feature channels to produce the fused pyramidal feature, as shown in Figure 5. Starting from the pyramidal feature at the third level, the *PDF* module first learns the offsets from the feature map, then applies the learned offsets to the deformable convolution and produces the output feature in the same level. For the second and first levels, we first obtain the convolved features through the same procedures as in the third level, and later aggregate the result with the bilinearly-upsampled feature from a higher level via a $1 \times 1$ convolution to produce the final output feature in the level. Note that, the employed deformable convolutions are with an offset group number of two, so the learned offset vectors for the pyramidal left and right features can be different. In other words, the sampled spatial locations can be different for the left and right feature maps. Altogether, we produce a three-level pyramidal feature from the input three-level pyramid features.

Without changing the feature dimension and size, our *PDF* module can be viewed as a feature transformation module, which exploits long-range correspondences and aggregates context information. Hence, it can also be applied to transform $\widetilde{\mathbf{P}_M^t}$ back to $\widetilde{\mathbf{P}_L^t}$ and $\widetilde{\mathbf{P}_R^t}$ with the newly learned offsets.

## 4.3 Quantization Layer

The mononized frame $\mathbf{O}_M^t$ in our framework is a regular monocular image in 8-bit pixel format per RGB channel. This means that we need to quantize the 32-bit floating point network-output values to 8-bit integers for producing $\mathbf{O}_M^t$. Such an operation is, however, not differentiable, since it hinders the network training with gradient descent. If we directly ignore the quantization process during the network training, the restored results could contain artifacts at the test phase (top row of Figure 6) due to the quantization error in the mononized view. Inspired by the works on propagating gradients through binarization [Hubara et al. 2016; Rastegari et al. 2017], image and network compression [Agustsson et al. 2017] and entropy coding [Ballé et al. 2017; Choi et al. 2019], we adopt a quantization layer (Figure 4 (ii)), which consists of quantization function $Q(x_{ijk})$

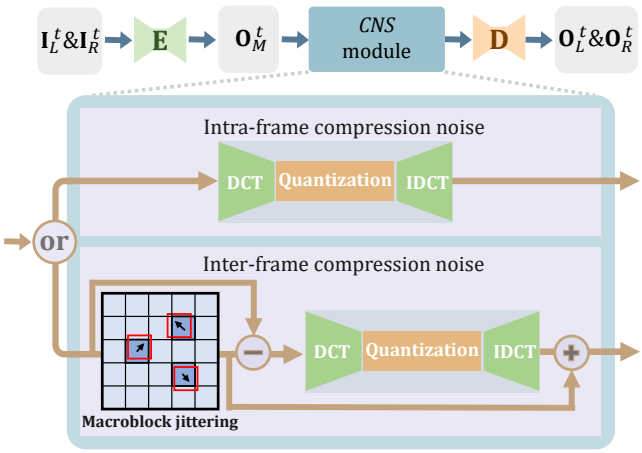



Fig. 7. Our extended framework with the *compression noise simulation (CNS)* module (pipeline on top). The *CNS* module between the encoder ($E$) and decoder ($D$) is designed to simulate the injection of intra- and inter-frame noise introduced by the video codec. DCT means Discrete Cosine Transform and IDCT is its inverse, while Macroblock means $16 \times 16$ pixels block.

and proxy function $\widetilde{Q}(x_{ijk})$:

$$\begin{cases} Q(x_{ijk}) &= round(x_{ijk}) \\ \widetilde{Q}(x_{ijk}) &= x_{ijk}\,, \end{cases} \tag{4}$$

where $Q(x_{ijk})$ is used in the forward pass and the gradient of $\widetilde{Q}(x_{ijk})$ is used in the backward propagation. By training the network with this quantization layer, we can better suppress the artifacts in the restored binocular frames (bottom row of Figure 6). Also, we explored other quantization strategies, e.g. universal quantization [Choi et al. 2019], to build the quantization layer, and found no significant difference in the performance. Please refer to Supplemental material Section 3 for the details.

### 4.4 Compression Noise Simulation (CNS) Module

For distribution to users, mononized videos could be streamed by using lossy video codecs, e.g., H.264/MPEG4-AVC [Wiegand et al. 2003], H.265/HEVC [Sullivan et al. 2012], VP9 [Mukherjee et al. 2015], and the newly issued AOMedia Video 1 (AV1) [Chen et al. 2018]. When streaming the mononized video at low bit-rates, the stereo information encoded in mononized videos may be distorted by the codecs, leading to bad restoration of the binocular video.

To resist the compression perturbation when collaborating with lossy video codecs, we further design an extended framework by inserting the *compression noise simulation (CNS)* module (Figure 7) into our framework when we train the whole framework. By introducing codec-like noise during the training, the framework can better learn to encode the stereo information in a compression-friendly mode, as well as to better restore the binocular frames from the distorted mononized frame. We design the *CNS* module to simulate the following two kinds of video codec noise:

(i) intra-frame noise. We employ the DCT quantization [Robertson and Stevenson 2005] to simulate the lossy operation inside a frame; and

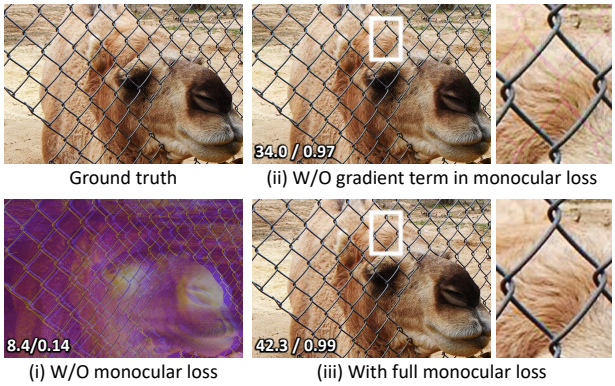



Fig. 8. Effect of the monocular loss $\mathcal{L}_M$. (i)-(iii) Mononized frames $\mathbf{O}_M^t$ produced with three different forms of monocular loss. The numbers in each result show the corresponding PSNR and SSIM values.

(ii) inter-frame noise. This kind of noise mainly comes from the quantization error of the residual signal due to the inaccurate motion compensation in conventional video codecs; we simulate it by macroblock ($16 \times 16$) jittering, as depicted in the lower half of Figure 7. The jittering probability is up to the variance of the estimated optical flow.

During the network training, we randomly choose to simulate intra- or inter-frame noise (Figure 7), corresponding to the cases of I and P frames in conventional video codecs.

## 5 LOSS FUNCTION

To drive the network training, we formulated three loss terms in the objective function (Eq. (3)), namely the monocular loss, invertibility loss, and temporal loss.

### 5.1 Monocular Loss

The monocular loss aims at producing an ordinary mononized frame $\mathbf{O}_M^t$ that looks like the input. Without loss of generality, we choose the input left view $\mathbf{I}_L^t$ as the reference. Therefore, we minimize the $L_2$ difference between $\mathbf{O}_M^t$ and $\mathbf{I}_L^t$. However, as $\mathbf{O}_M^t$ contains the visual information from both $\mathbf{I}_L^t$ and $\mathbf{I}_R^t$, using the $L_2$ alone is insufficient and may result in noticeable patterns, which are closely related to the image details in $\mathbf{I}_R^t$ (Figure 8 (top-right)).

To suppress the artifact, we further minimize the Charbonnier difference [Charbonnier et al. 1994] between the gradients of $\mathbf{O}_M^t$ and $\mathbf{I}_L^t$ in the monocular loss:

$$\mathcal{L}_M = \mathbb{E}_{\{\mathbf{I}_L^t, \mathbf{I}_R^t\} \in \mathcal{S}} \{ ||\mathbf{O}_M^t - \mathbf{I}_L^t||_2 + \alpha \cdot \rho(\nabla \mathbf{O}_M^t - \nabla \mathbf{I}_L^t) \}\,, \tag{5}$$

where $\mathbb{E}$ denotes the average expectation over $N$ frames in a sequence; $\alpha$ is a weight; $\nabla$ is the gradient; and $\rho(x) = \sqrt{x^2 + \epsilon^2}$ is the Charbonnier $L_1$ function [Charbonnier et al. 1994], where the constant $\epsilon$ is set to $1 \times 10^{-6}$. Figure 8 (bottom-right) shows an example restored frame produced with this monocular loss, in which the artifact has been greatly suppressed.

### 5.2 Invertibility Loss

The ability of restoring the binocular frames from mononized frame $\mathbf{O}_M^t$ is secured by the *invertibility loss* $\mathcal{L}_I$. It minimizes the difference between the restored binocular frame pair $\{\mathbf{O}_L^t, \mathbf{O}_R^t\}$ and the original inputs $\{\mathbf{I}_L^t, \mathbf{I}_R^t\}$:

$$\mathcal{L}_I = \mathbb{E}_{\{\mathbf{I}_L^t, \mathbf{I}_R^t\} \in \mathcal{S}} \{(1-\beta) \cdot ||\mathbf{O}_L^t - \mathbf{I}_L^t||_2 + \beta \cdot ||\mathbf{O}_R^t - \mathbf{I}_R^t||_2\}, \quad (6)$$

where $\beta$ is a weight to balance the quality of the restored left and right views. Restoring the right view is much harder than the left one, since the left view is taken as the reference for forming the mononized view. Hence, we set $\beta$ to 0.99 in practice. Note that $\mathcal{L}_I$ effectively imposes constraints over the parameters in both the encoding and decoding networks, as we jointly train the two networks to produce $\{\mathbf{O}_L^t, \mathbf{O}_R^t\}$ in our framework.

### 5.3 Temporal Loss

Mononizing binocular images can be achieved with the above two loss terms $\mathcal{L}_M$ and $\mathcal{L}_I$. However, using them alone is inadequate to ensure the temporal coherence when mononizing binocular videos. As mentioned in Section 3, we add recurrent connections in the framework to introduce the information of previous frame into the current one, but doing so cannot drive the model to learn how to utilize such information. Hence, we design the temporal loss to maintain the coherence between successive frames:

$$\begin{aligned}\mathcal{L}_T = \mathbb{E}_{\{\mathbf{I}_L^t, \mathbf{I}_R^t\} \in \mathcal{S}} \{ & ||\mathcal{W}(\mathbf{O}_M^{t-1}, \mathbf{F}_L^t) - \mathbf{O}_M^t||_2 \\ & + ||\mathcal{W}(\mathbf{O}_L^{t-1}, \mathbf{F}_L^t) - \mathbf{O}_L^t||_2 \\ & + \gamma \cdot ||\mathcal{W}(\mathbf{O}_R^{t-1}, \mathbf{F}_R^t) - \mathbf{O}_R^t||_2 \},\end{aligned} \quad (7)$$

where $\mathbf{F}_L^t$ is the estimated optical flow from $\mathbf{I}_L^{t-1}$ to $\mathbf{I}_L^t$; $\mathbf{F}_R^t$ is the estimated optical flow from $\mathbf{I}_R^{t-1}$ to $\mathbf{I}_R^t$; $\mathcal{W}(\mathbf{X}, \mathbf{F})$ produces a warped image of $\mathbf{X}$ by using the optical flow $\mathbf{F}$; and $\gamma$ is a weight. Since restoring a high-quality $\mathbf{O}_R^t$ is more challenging than $\mathbf{O}_L^t$, we put a relatively larger weight on the term for $\mathbf{O}_R^t$. Comprehensive qualitative and quantitative analysis will be presented in Section 7.3.2 to demonstrate the effect of the temporal loss.

## 6 TRAINING

*Training data.* Publicly-available datasets with binocular frame sequences such as *KITTI* [Geiger et al. 2013] are often too specific in genre. Hence, we compile a *3D movie* dataset that contains 122 3D movie sequences with 720p resolution (5,876 frames in total) collected from *Inria*[3] [Seguin et al. 2015] and *YouTube*[4]. Since some stereoscopic videos in YouTube were artificially produced by a naive mono to stereo conversion, we intentionally avoided them by estimating the disparity of each collected video and ignoring those with unnatural disparity. Overall, the dataset covers eight types of scenes, e.g, *animals and pets*, *architecture*, *cartoon*, and *natural scenery*; see Supplemental material Section 2 for the detailed description. Further, we employed the pre-trained *PWC-Net* [Sun et al. 2018] to estimate the optical flows between consecutive frame pairs in each movie sequence. Lastly, we randomly selected 69 sequences as the training set and used the remaining 53 sequences as the test set.

[3]Inria: https://www.di.ens.fr/willow/research/stereoseg/
[4]YouTube: https://www.youtube.com/

Besides, we employ a stereo image dataset, *Flickr1024* [Wang et al. 2019b], which contains 1,024 binocular images of various categories, to train the image version of our method for comparison with other methods. Here, we follow the official train/test split in Flickr1024.

*Training details.* We implemented our encoding and decoding networks using PyTorch [Paszke et al. 2019] and trained them jointly with the loss function defined in Eq. (3). Each mini-batch training samples contains $N \times B$ frames, where $N$ is the number of consecutive frames; $B$ is the number of instances; and each frame is randomly cropped into $256 \times 256$ resolution during the training. We empirically set $N = 4$ and $B = 16$ in the training.

We optimized our model by the Adam solver [Kingma and Ba 2015], in which the learning rate was initially set to 0.0001 and further reduced by a factor of 3.33 when the loss plateaus (known as ReduceLROnPlateau in PyTorch). For the image version of our model, we trained the networks on the training set of *Flickr1024* for 200 epochs, while for the video version, we trained the networks for 300 epochs on the training set of the compiled *3D movie* dataset. During the training, the video version of our framework was initialized by the image version of our trained framework model, since we empirically found that doing so improves the overall performance than simply initializing the network parameters from scratch. The hyper-parameters in the loss function are set as following: $\lambda_1 = 1.7$; $\lambda_2 = 1.3$; $\alpha = 0.1$; $\beta = 0.99$; and $\gamma = 10.0$. *Code is available at the following GitHub page: https://github.com/wbhu/Mono3D.*

## 7 RESULTS AND DISCUSSION

### 7.1 Qualitative Evaluation

Figure 17 at the end of this paper showcases four example results produced by our method, featuring indoor and outdoor contents with close-up objects and faraway scenery. In each example, we show a tabular figure of 2×4 images: the input left and right views (1st column); mononized view and its difference from the input left view (2nd column); restored left and right views (3rd column); and their differences from the corresponding inputs (4th column). The numbers in each result ($\mathbf{O}_M$, $\mathbf{O}_L$, and $\mathbf{O}_R$) show the PSNR and SSIM values compared with the corresponding input (ground truth). For the difference maps, we compute the absolute pixel value difference in the scale of [0,255] and color-code the difference value. The mean absolute differences are often very small (only around two).

In Figure 17, the top three examples are still pictures, whereas the bottom-most one is a video frame in the test dataset (more video results can be found in the supplemental video). Since video examples are usually less challenging due to small disparity, we pick three more challenging still pictures to show the capability of our network to handle occlusions and large disparity. See particularly the top example, the baby monkey on top of the right view ($\mathbf{I}_R$) is mostly occluded by the front monkey in the left view ($\mathbf{I}_L$), so it is visually absent in the mononized image ($\mathbf{O}_M$); *yet, our method can restore it ($\mathbf{O}_R$) solely from the mononized image ($\mathbf{O}_M$)*, just like the face of the little girl shown in Figure 1. Also, the pixel value differences in the occluded region, e.g., the baby monkey, are not obviously higher, as revealed in the difference map. These results demonstrate the capability of our method to implicitly encode the stereo information in a nearly-imperceptible form inside the mononized view and later


Fig. 9. From left to right: the input left and right views, followed by the mononized images produced by *DeepSteno* (from [Baluja 2017]), *InvertGray* (from [Xia et al. 2018]), and the image version of our framework (**Mono3D**$_{\text{img}}$). Note the PSNR/SSIM values shown in each result. Our mononized result does not have obvious artifacts, such as color shifting and traces of objects that come from the right view; see the blown-up views on the bottom row.

restore from it the binocular views. More visual comparison results can be found in Supplemental material Section 4.

## 7.2 Comparison with Other Methods

So far, no methods have been developed for mononizing binocular images and videos. Hence, to evaluate and demonstrate the quality of our method, we adopt the following three related works for comparison: (i) deep stenography [Baluja 2017] (denoted as *DeepSteno*), in which we take the right view as the secret image and use its preparing and hiding networks to conceal the right view in the left view, then further reconstruct the right view from the stenographed image by its reveal network; (ii) the reversible framework in [Xia et al. 2018] (denoted as *InvertGray*), in which we concatenate the left and right views along the channel dimension, feed the result into its encoding network to produce the mononized view, then use its decoding network to restore the left and right views; and (iii) further, we explore the possibility of dropping the right view and synthesizing it from the left one using the recent novel-view-synthesis method, *3D Ken burns* [Niklaus et al. 2019]. Clearly, view synthesis might not produce high-quality results; here, we take it as a baseline to see if our method can encode stereo information in the mononized (left) view for reconstructing a better right view.

Since the three methods are originally designed for still pictures, for a fair comparison, we adopted our framework for mononizing binocular images (denoted as **Mono3D**$_{\text{img}}$) without the recurrent connections, temporal loss, and CNS module, and trained it on the *Flickr1024* dataset (Section 6). For *DeepSteno*, we implemented its method according to its paper, as there is no public code. For *InvertGray*, we obtained code from its project webpage. Then, we re-trained their networks on *Flickr1024* with our loss function, since their original loss functions are not designed for mononizing binocular images. For *3D Ken burns*, we adopted their released trained model and manually tuned the camera pose to best align the synthesized right view with the ground-truth right view.

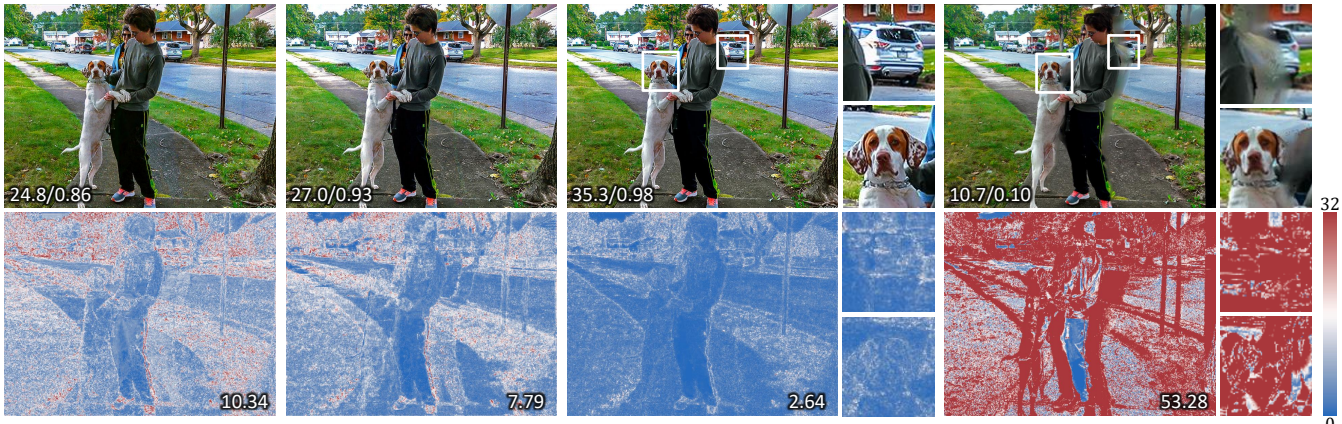
*7.2.1 Evaluation on the mononized views.* Figure 9 shows the mononized views produced by *DeepSteno*, *InvertGray*, and our framework, in which we can see color shifting problem in the results of *DeepSteno* and *InvertGray*. Also, we can observe obvious traces of objects from the right view; see the blown-up views on the bottom of Figure 9. Compared with our framework, *DeepSteno* does not take into account the relations between the container (left view) and secret (right view) images, whereas *InvertGray* cannot be directly extended for mononizing binocular images. *InvertGray* is designed to handle aligned luminance and chrominance in the invertible grayscale problem, so it cannot effectively harvest the long-range correspondences between the left and right views. Besides, both cannot maintain frame-to-frame coherence in the video inputs. Thanks to the architecture and the pyramidal deformable fusion module, our framework can learn to leverage the correspondences between left and right views to mononize binocular images. From Figure 9, we can see that our mononized view does not exhibit obvious traces from the right view, meaning that the stereo information can be implicitly encoded in a nearly-imperceptible form.

Table 1. Visual quality of the mononized views and restored binocular (left & right) views produced by *DeepSteno*, *InvertGray*, and our framework (**Mono3D**$_{\text{img}}$) on *Flickr1024*. Note that *DeepSteno* does not restore the left view, as its reveal network is only designed for restoring the secret image.

| Methods | Mono-view | | L. Bino-view | | R. Bino-view | |
|---|---|---|---|---|---|---|
| | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| *DeepSteno* | 26.1 | 0.81 | — | — | 27.9 | 0.88 |
| *InvertGray* | 28.0 | 0.89 | 28.7 | 0.92 | 30.7 | 0.92 |
| **Mono3D**$_{\text{img}}$ | **37.8** | **0.97** | **38.3** | **0.99** | **37.3** | **0.98** |

Further, we quantitatively compare the visual quality of the mononized views produced by various methods on the whole *Flickr1024* test set in terms of PSNR and SSIM. As shown in the Mono-view column of Table 1, the average PSNR and SSIM values of our mononized views (37.8 & 0.97) are far higher than those of *DeepSteno* (26.1 & 0.81) and *InvertGray* (28.0 & 0.81). These statistical results quantitatively demonstrate the effectiveness of our method.

*7.2.2 Evaluation on restored/synthesized views.* Next, we compare the restored right view produced by *DeepSteno*, *InvertGray*, and our framework. From the first three columns shown in Figure 10, we can see that our result has much higher PSNR and SSIM, whereas those




Restored by *DeepSteno* | Restored by *InvertGray* | Restored by our framework | Synthesized by [Niklaus et al. 2019]

Fig. 10. Restored/synthesized right views produced by *DeepSteno*, *InvertGray*, the image version of our framework (**Mono3D**$_{\text{img}}$), and the novel-view-synthesis method [Niklaus et al. 2019]. PSNR/SSIM values and mean absolute error (in the scale of 255) are shown in each result. We can see from the error maps on the bottom row that our restored view is significantly closer to the ground-truth results with very small pixel color differences.

of *DeepSteno* and *InvertGray* contain traces of objects from the left view, as can be seen in the error maps shown in the figure. Overall, the average PSNR of our restored right views are well above 35dB, compared with those of *DeepSteno* and *InvertGray*. The statistical results for the restored right views in Table 1 further confirm the findings. Note that similar statistical results are also obtained for the restored left views (Table 1). Here, we do not consider the restored left view when comparing with *DeepSteno*, since its reveal network is designed only for restoring the secret image, so it restores only the right view but not the left one from the stenographed image.

Further, the last column in Figure 10 shows the synthesized right view by *3D Ken burns*. As discussed earlier, it is very hard to estimate accurate depth and infer plausible results for the occluded regions, so the results of *3D Ken burns* tend to be blurry on the inpainted regions. We admit that this comparison is not entirely fair, as inferring and restoration are not directly comparable. Yet, the comparison gives evidence that our encoding-and-decoding approach is able to recover the stereo information not available in the left view.

## 7.3 Quantitative Evaluation

Next, we quantitatively evaluate our method on the test set of the *3D movie* dataset (Section 6). To verify the effectiveness of some key designs in our method, we consider five variants of our method:

- **Mono3D**$_{\text{video}}$: our full method for mononizing binocular videos;
- **Mono3D**$_{\text{video}}^{\text{single-scale}}$: we remove the first two levels of pyramidal features in **Mono3D**$_{\text{video}}$ and use only the third-level feature;
- **Mono3D**$_{\text{video}}^{\text{w/o DConv}}$: from **Mono3D**$_{\text{video}}$, we replace the deformable convolution in the PDF modules with conventional convolution;
- **Mono3D**$_{\text{video}}^{\text{w/o PDF}}$: from **Mono3D**$_{\text{video}}$, we remove the PDF modules, directly concatenate $\mathbf{P}_L^t$ and $\mathbf{P}_R^t$ into $\mathbf{P}_M^t$ in the encoding network, and separate $\widetilde{\mathbf{P}_M^t}$ into $\widetilde{\mathbf{P}_L^t}$ and $\widetilde{\mathbf{P}_R^t}$ simply along the channel dimension in the decoding network; and
- **Mono3D**$_{\text{img}}$: our method for mononizing binocular images, i.e., **Mono3D**$_{\text{video}}$ without the recurrent connections from previous frames, temporal loss, and CNS module (Figure 3).

Table 2. Frame quality of the mononized and restored binocular views produced by the five variants of our method (**Mono3D**$_{\text{img}}$, **Mono3D**$_{\text{video}}^{\text{single-scale}}$, **Mono3D**$_{\text{video}}^{\text{w/o DConv}}$, **Mono3D**$_{\text{video}}^{\text{w/o PDF}}$, and **Mono3D**$_{\text{video}}$) over the entire test set of *3D movie*.

| Method variants | Mono-frame | | L. Bino-frame | | R. Bino-frame | |
|---|---|---|---|---|---|---|
| | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
| **Mono3D**$_{\text{img}}$ | 38.6 | 0.97 | 39.5 | 0.98 | 37.9 | 0.97 |
| **Mono3D**$_{\text{video}}^{\text{single-scale}}$ | 29.7 | 0.90 | 30.1 | 0.92 | 30.5 | 0.92 |
| **Mono3D**$_{\text{video}}^{\text{w/o DConv}}$ | 36.6 | 0.94 | 36.8 | 0.95 | 35.7 | 0.93 |
| **Mono3D**$_{\text{video}}^{\text{w/o PDF}}$ | 34.1 | 0.92 | 34.9 | 0.94 | 33.9 | 0.93 |
| **Mono3D**$_{\text{video}}$ | **39.0** | **0.98** | **39.8** | **0.99** | **38.7** | **0.98** |

*7.3.1 Evaluation on frame quality.* We adopt PSNR and SSIM to measure the frame quality of our results relative to the inputs as the ground truths. Table 2 lists the PSNR and SSIM values of the mononized and restored binocular frames produced by the five variants of our method on the entire *3D movie* test set.

Comparing the statistical results of **Mono3D**$_{\text{img}}$ and **Mono3D**$_{\text{video}}$, we can see that both can produce high-quality results with PSNR well above 35dB, while **Mono3D**$_{\text{video}}$ performs slightly better, showing that the recurrent connections help introduce extra useful information from previous time frames to improve the performance. Note that PSNR and SSIM only measure the quality of individual frames without considering the temporal quality. More analysis on the temporal frame quality will be presented in Section 7.3.2.

Comparing the results of **Mono3D**$_{\text{video}}^{\text{single-scale}}$ and **Mono3D**$_{\text{video}}$ as shown in Table 2, we can see that **Mono3D**$_{\text{video}}$ performs significantly better. Since the pyramidal features help harvest both low- and high-level information, **Mono3D**$_{\text{video}}$ leads to better results for both the mononized and restored binocular frames.

Comparing the results of $\textbf{Mono3D}_{\text{video}}^{\text{w/o PDF}}$ and $\textbf{Mono3D}_{\text{video}}^{\text{w/o DConv}}$, we can see that $\textbf{Mono3D}_{\text{video}}^{\text{w/o DConv}}$ performs better, since the pyramidal fusion improves the encoding and decoding of stereo information to and from the mononized frames. More importantly, comparing the statistical results of $\textbf{Mono3D}_{\text{video}}^{\text{w/o DConv}}$ and $\textbf{Mono3D}_{\text{video}}$, we can see that $\textbf{Mono3D}_{\text{video}}$ performs even better. Such result quantitatively shows the effectiveness of the PDF module for exploiting long-range correspondences and fusing features from the left and right views.

*7.3.2 Evaluation on temporal coherence.* There is no standard way to measure the temporal coherence of a video. The most direct way is to show the video to humans and let them evaluate the temporal coherence. Readers are recommended to watch our supplement video for the evaluation. Besides, we extract a line of pixels at a fixed location in videos, and stack them over time as an image of *temporal profile*; see the right-hand side of Figure 11 for examples. Comparing the temporal profiles of the restored right frames from $\textbf{Mono3D}_{\text{img}}$ and $\textbf{Mono3D}_{\text{video}}$, as well as the raw frames from the ground-truth input (**GT**), we can see that the temporal profile of $\textbf{Mono3D}_{\text{video}}$ is much smoother than that of $\textbf{Mono3D}_{\text{img}}$, and it also looks more similar to the temporal profile of the ground truth.

To quantitatively evaluate the temporal coherence, we explore an observation that if a generated video ($\mathbf{O}_M^t$, $\mathbf{O}_L^t$, or $\mathbf{O}_R^t$) is temporally coherent, each frame in the video should be more predictable from the previous one via optical flow estimated between frames in the original video. Based on this idea, we first use *PWC-Net* [Sun et al. 2018] to estimate the optical flows between each pair of successive frames in the input, i.e., $\mathbf{F}_x^t$ from $\mathbf{I}_x^{t-1}$ to $\mathbf{I}_x^t$, where $x$ is $L$ or $R$ for left or right view, respectively. Then, we warp each frame in the generated videos, and compute the *warping deviation* between the warped frame ($\mathcal{W}(\mathbf{O}_x^{t-1}, \mathbf{F}_x^t)$) and next frame ($\mathbf{O}_x^t$) in the video:

$$\Delta_x^t = |\,\mathcal{W}(\mathbf{O}_x^{t-1}, \mathbf{F}_x^t) - \mathbf{O}_x^t\,|, \tag{8}$$

where $x$ is $M$, $L$, or $R$, and $\mathbf{F}_M^t$ is taken as $\mathbf{F}_L^t$. Likewise, we compute also the warping deviation for the input binocular videos:

$$\hat{\Delta}_x^t = |\,\mathcal{W}(\mathbf{I}_x^{t-1}, \mathbf{F}_x^t) - \mathbf{I}_x^t\,|, \tag{9}$$

where $x$ is $L$ or $R$, and $\hat{\Delta}_x^t$ are mostly zeros, except in areas that $\mathbf{I}_x^t$ cannot be warped from $\mathbf{I}_x^{t-1}$, e.g., occlusion and lighting changes. For temporally-coherent videos, $\Delta_x^t$ should be coherent with $\hat{\Delta}_x^t$. Hence, we define the *temporal deviation* of a generated video as the mean absolute relative error of $\Delta_x^t$ w.r.t. $\hat{\Delta}_x^t$ over the whole image:

$$\sigma^t = \Big(\prod_{i=1}^{W}\prod_{j=1}^{H}\prod_{c=1}^{3}\Big|\frac{\Delta_{x(i,j,c)}^t}{\hat{\Delta}_{x(i,j,c)}^t + \epsilon}\Big|\Big)^{\frac{1}{H\times W\times 3}}, \tag{10}$$

where $\sigma^t$ is a scalar that indicates the temporal deviation at time $t$, $W$ is image width, $H$ is image height, $(i, j)$ is pixel index, $c$ is RGB channel index, and $\epsilon$ is set as $1 \times 10^{-3}$ in practice to avoid division by zero. Note that $\sigma^t \approx 1$ shows high temporal coherence, and vice versa. It is because if a given video is not temporally coherent, its warping deviation ($\Delta_x^t$) will be very far from that of the input ($\hat{\Delta}_x^t$).

Figure 11 presents plots of $\sigma^t$ over time for the case of $\textbf{Mono3D}_{\text{video}}$ and $\textbf{Mono3D}_{\text{img}}$, as well as for the ground truth (**GT**), typically for the right binocular frames in a challenging video example. Here, if we compare the warping deviations of **GT** with itself, the resulting

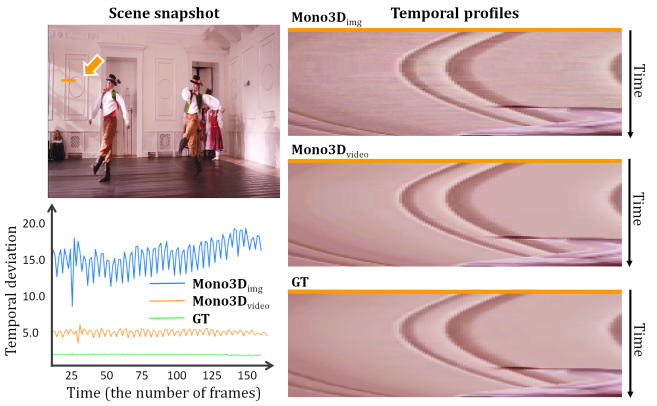



Fig. 11. Temporal coherence evaluation. The top-left image shows a snapshot of the scene; the right three images show the temporal profiles of $\textbf{Mono3D}_{\text{img}}$, $\textbf{Mono3D}_{\text{video}}$, and **GT** (ground truth) along the pixels in the orange line segment marked by the orange arrow in the scene snapshot; and the bottom-left plot shows the temporal deviation of the right binocular frames restored by $\textbf{Mono3D}_{\text{img}}$ and $\textbf{Mono3D}_{\text{video}}$, and of the raw frames directly from the ground truth, i.e., the input binocular video.

Table 3. Temporal coherence evaluation via temporal deviation (Eq. (10)) on the *3D movie* test set. Better temporal coherence should be closer to one.

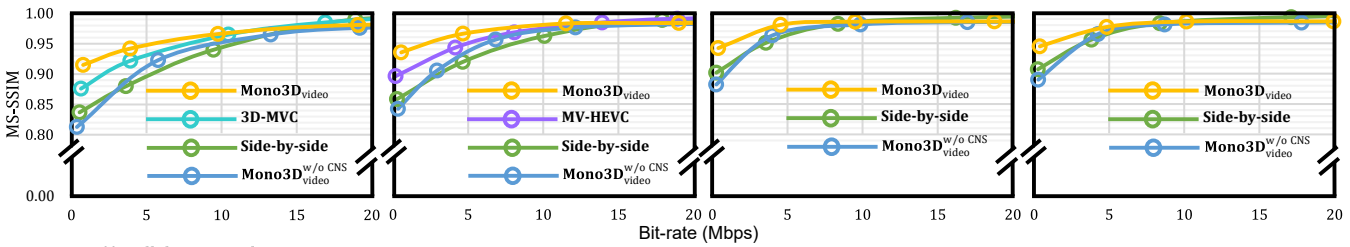
| Method variants | Mono-frame | L. Bino-frame | R. Bino-frame |
|---|---|---|---|
| $\textbf{Mono3D}_{\text{img}}$ | 1.90 | 1.55 | 4.73 |
| $\textbf{Mono3D}_{\text{video}}$ | 1.24 | 1.10 | 1.22 |

Table 4. Timing statistics of our method for one frame (in milliseconds).

| Resolution | Encoding | Decoding |
|---|---|---|
| $480 \times 720$ | 10.9 | 10.5 |
| $720 \times 1280$ | 18.6 | 18.5 |
| $1080 \times 1920$ | 27.3 | 26.9 |

$\sigma^t$ values are almost one, as shown in the plot for **GT**. More importantly, we can see that the plot of $\textbf{Mono3D}_{\text{video}}$ is always closer to the plot of **GT** and lower than the plot of $\textbf{Mono3D}_{\text{img}}$, thus showing that $\textbf{Mono3D}_{\text{video}}$ produces more temporally-coherent videos than $\textbf{Mono3D}_{\text{img}}$. To statistically confirm the results, we compute the geometric mean $\sigma$ values over time for all the videos in the *3D movie* test set. From the geometric mean $\sigma$ values shown in Table 3, we can see that the temporal deviation of all the results produced by $\textbf{Mono3D}_{\text{video}}$ are very close to one, compared with $\textbf{Mono3D}_{\text{img}}$, thus demonstrating how temporal coherence is improved by having the temporal loss and recurrent connections in our framework.

*7.3.3 Timing performance.* We implemented our method using *PyTorch* and ran all experiments on a PC equipped with a Titan Xp GPU and Intel Core i9-7900X@3.30GHz CPU. We evaluated the time performance of our method for one frame in multiple resolutions. Table 4 shows the timing statistics, where we exclude the time to transfer data between the GPU and CPU, since it has nothing to do with the method and can be optimized using memory cache and pipeline techniques. From the results, we can see that our method performance can support real-time applications on *1080p* videos.




(i) Collaborate with H.264 (ii) Collaborate with H.265 (iii) Collaborate with VP9 (iv) Collaborate with AV1

Fig. 12. Rate-distortion (R-D) curves of **Mono3D**$_{video}$, **3D-MVC**, **Side-by-side**, and **Mono3D**$^{w/o\ CNS}_{video}$. The distortion is measured by MS-SSIM and bit-rate is represented as megabits per second (Mbps). All the restored binocular videos have $1280 \times 720$ resolution and 25 FPS for both the left and right views.

## 7.4 Compatibility with Video Codecs

Next, we evaluate the compatibility of our framework with common video codecs: (i) H.264/MPEG4-AVC [Wiegand et al. 2003] and (ii) H.265/HEVC [Sullivan et al. 2012], (iii) VP9 [Mukherjee et al. 2015], which is commonly-used in web browsers, and (iv) AOMedia Video 1 (AV1) [Chen et al. 2018], which aims to be the next-generation codec. Overall, we passed the mononized videos produced by our framework to these codecs for encoding and decoding, thus emulating the use of our mononized videos as regular monocular videos in common video platforms. Then, we fed the mononized videos decoded by the codecs into our decoder network to restore the binocular videos, and measured the quality of the restored binocular videos relative to the originals.

To the best of our knowledge, there is no multi-view codec extension that is backward compatible with multiple monocular video codecs, while our method is fully backward-compatible. So far, there are only codec-specific multi-view extensions as discussed in Section 2.3. Here, our comparison includes the following methods:

- **Mono3D**$_{video}$: our full method for mononizing binocular videos;
- **Mono3D**$^{w/o\ CNS}_{video}$: our **Mono3D**$_{video}$ without the CNS module;
- **Side-by-side**: concatenate each pair of left and right frames side-by-side [Vetro 2010], and encode them using each of the four video codecs mentioned above;
- **3D-MVC**[5]: an multi-view extension of H.264/MPEG4-AVC; and
- **MV-HEVC**[6]: an multi-view extension of H.265/HEVC.

In this experiment, we followed existing video compression research to use MS-SSIM [Wang et al. 2003] to measure the quality of the restored video over the test dataset, and plotted the rate-distortion (R-D) curves (Figure 12) to explore video quality vs. bit-rate.

Comparing the R-D curves of **Side-by-side**, **3D-MVC**, **MV-HEVC** and our **Mono3D**$_{video}$ in Figure 12, we can see that **Mono3D**$_{video}$ outperforms all others when encoding with low bit-rates (< 10 Mbps) for all the four codecs. Importantly, the multi-view extensions, e.g., **3D-MVC** and **MV-HEVC**, all depend on the monocular codecs (for instance, we cannot apply MV-HEVC to the H.264/MPEG4-AVC codec), the **Side-by-side** method is not compatible with existing monocular TVs, whereas our **Mono3D**$_{video}$ is fully compatible with the existing monocular codecs and TV systems. When encoding with high bit-rates (> 15 Mbps), we can see that **Mono3D**$_{video}$ performs slightly worse than **3D-MVC**, **MV-HEVC** and **Side-by-side**. Yet, the overall video quality is very close and comparable with others, as shown in the plots, for all the four codecs.

Comparing the R-D curves of **Mono3D**$_{video}$ and **Mono3D**$^{w/o\ CNS}_{video}$, we can see that **Mono3D**$_{video}$ significantly outperforms **Mono3D**$^{w/o\ CNS}_{video}$ at low bit-rates for all the four codecs. These results show that our CNS module helps the framework to learn to encode the stereo information in a compression-friendly manner by introducing codec-like noise in the network training. Also, **Mono3D**$_{video}$ and **Mono3D**$^{w/o\ CNS}_{video}$ perform similarly at high bit-rates, since the compression perturbation of modern codecs becomes too trivial at high bit-rates (with more storage resource in the encoding). Moreover, the R-D curve trends of **Mono3D**$_{video}$ are similar for all the four plots, showing that our method is not sensitive to specific codecs.

Overall, the experimental results show the backward-compatibility of our framework with various common video codecs. We can encode and stream our mononized videos on existing monocular platforms, just as the regular monocular videos. On top of this, we can restore binocular videos from the streamed mononized videos for stereoscopic viewing, if a 3D display is available.

## 7.5 User Study

Further, to emulate the use of our mononized videos in existing video platforms, we conducted a user study to evaluate the perceptual quality of the mononized videos and restored binocular videos produced by **Mono3D**$_{video}$, in combination with the H.264 video codec. Here, we chose H.264, since our method has slightly lower performance with H.264 (Figure 12). Also, H.264 is the most common codec nowadays. For the mononized videos, we evaluate the frame quality, temporal smoothness, and overall video quality. For the restored binocular videos, besides the above three quantities, we additionally evaluate the depth perception of the binocular videos; see questions Q1-Q7 listed in Figure 13 (right).

*Preparation.* To start, we prepared four types of videos:

(i) *Ground truth.* We randomly selected eight video samples (720*p*, 25 FPS), one per category from the *3D movie* test set, as the ground-truth binocular videos, and simply regarded the left views as the ground-truth monocular videos;
(ii) *Ours.* From these ground truths, we generated the mononized videos using **Mono3D**$_{video}$, then encoded and decoded each

[5] Available at https://www.videohelp.com/software/FRIM
[6] Available at https://github.com/listenlink/3D-HEVC

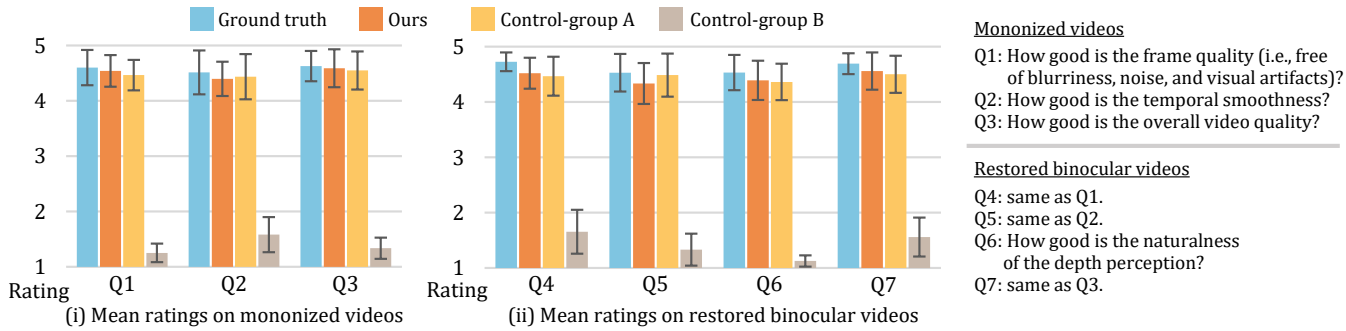



Fig. 13. User study results. Mean ratings given by the participants on the mononized videos (i) and restored binocular videos (ii) for the cases of ground truth, our framework, control-group A, and control-group B (with slight noise, temporal flicker, etc.). The error bar on each column indicates the standard deviation.

mononized video by the H.264 codec at ~5 Mbps[7], and finally restored from them the binocular videos using **Mono3D**$_{\text{video}}$.

(iii) *Control-group A.* We encoded and decoded the ground truths by the multi-view extension of H.264, 3D-MVC, at ~5 Mbps to get back the binocular videos; here, we also regarded the left views as the monocular videos.

(iv) *Control-group B.* From the ground truths, we encoded and decoded the left view of them by the H.264 codec at low bit-rate (~1 Mbps) to slightly inject some visual artifacts, such as noise and blurriness in the video frames; then, we randomly removed some frames in the videos to create slight temporal discontinuity as the monocular videos. Further, we shifted the generated monocular videos some pixels to the left (as shown in Figure 8 in the supplementary material), to act as the right view videos, and regarded the monocular videos together with the shifted videos as the binocular videos.

Control-group A is set for simulating the conventional video quality of monocular and binocular platforms, while control-group B is set for checking whether the users carefully participate in the study. Note also that the videos in control-group B are not obviously bad; see Supplemental material Section 5 for examples.

Altogether, we prepared eight sets of videos (ground truths, our framework (ours), control-group A, and control-group B) for monocular videos, and another eight sets for binocular videos. Concerning the participants, we recruited 25 participants: 10 females and 15 males, all with normal vision, and aged 24 to 28.

*Procedure.* Our user study has three sessions. The first one is a tutorial session. When a participant came to our lab, we first showed to him/her some normal videos and some control videos (like those in control-group B) for both monocular and binocular. This was to ensure that the participants knew the meaning of visual artifacts, temporal smoothness, and depth perception, and could perceive depth, when watching the binocular videos. Here, we employed the *Bino Player*[8] to show binocular videos on a 27" polarized 3D display, with the resolution of 1920 × 1080 and the peak luminance of $250cd/m^2$.

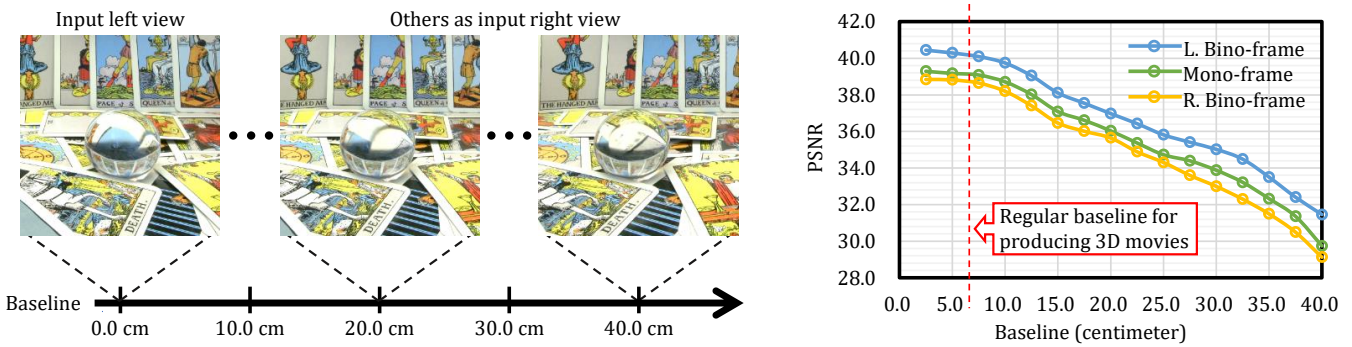
The second session focused on the eight sets of monocular videos. Here, we showed the videos to each participant set by set, in which the four video types in each set (i.e., ground truths, ours, control-group A, and control-group B) were shown in random order. To avoid bias, we use different random orders for different sets. After seeing all the four videos in a set, the participant might go back and forth in the playlist to carefully examine the four videos again before they gave a rating in the scale of one (poor quality) to five (excellent quality) per video for questions Q1 to Q3 listed in Figure 13.

The third session followed a similar procedure as in the second session, but we showed the other eight sets of binocular videos on the 3D display and asked for ratings on questions Q4 to Q7 (Figure 13). After the three sessions, we obtained a total of "25 participants × 8 video samples × 4 video types" per question.

*Analysis of the results.* Figure 13 (left) summarizes the mean participant ratings for questions Q1 to Q3 on the four types of monocular videos. Comparing the ratings on control-group B vs. others, we can clearly see that the participants could distinguish the control-group B videos from others. Comparing the ratings on ground truths, ours, and control-group A, we can see that their rating distributions are very similar for all the three questions Q1 to Q3. To statistically compare them, we performed an equivalence test using the two-one-sided t-test (TOST) [Schuirmann 1987], because t-test can only help to examine significant difference between two groups of data, while equivalence test can tell whether the two groups are equivalent under a given bound, which is specified to denote the smallest effect size of interest. Here, the upper and lower equivalence bounds in TOST are set to be 0.5 and −0.5, respectively, since the participant ratings are all integers and 0.5 is half the interval size. The result of the TOST shows that the ratings on ours and control-group A for questions Q1-Q3 are equivalent with confidence values 99.2%, 99.5%, and 99.7%, respectively. On the other hand, the confidence values are 99.4%, 98.2%, and 99.4%, respectively, for the ratings on ours and ground truths for Q1-Q3. Hence, the results suggest that the ratings on ours and ground truths for Q1-Q3 are equivalent, meaning that there are no obvious perceptual differences between the original (left view) videos and our mononized videos.

Next, we look at Figure 13 (right) for the binocular videos. Again, the control-group B videos can be recognized by the participants, and the rating distributions for ground truths, ours, and control-group A are similar for questions Q4 to Q7. Using TOST with the same setting as before, we found that the ratings on ours and control-group A for Q4 to Q7 are equivalent with confidence values 99.2%,

[7]The recommended bit-rate of YouTube is 5 Mbps for 720p 25 FPS monocular videos, https://support.google.com/youtube/answer/1722171.
[8]Bino Player: https://bino3d.org/




(i) Set up of producing stereo image pairs of increasing baseline

(ii) Performance curves under different baselines

Fig. 14. Stress test on our method using stereo image pairs of increasing "baseline" distances. (i) illustrates the set up, in which we treated the left-most image as the input left view and other images as the input right view. (ii) plots the quality of our results for stereo image pairs of increasing baseline.

98.7%, 99.9%, and 99.6%, respectively, whereas the ratings on ours and ground truths for Q4 to Q7 are equivalent with confidence values 97.9%, 98.6%, 99.1%, and 98.5%, respectively. Hence, there are no obvious perceptual differences, both between ours and control-group A and between ours (our restored binocular videos) and ground truths. The detailed questionnaire and example frames can be found in Section 5 of the supplemental material.

## 7.6 Discussion

*How does the performance vary with inter-camera distance?* The interpupillary distance, or the distance between the centers of the pupils of the eyes, varies from 5.1 to 7.7 cm for adults, whereas the average is 6.2 cm for females and 6.4 cm for males[9]. Hence, for general 3D movies that aim to reproduce natural human vision, the distance between camera centers are often set to a "normal" baseline of 5 to 8 cm[10]. Obviously, the task of mononizing binocular videos would become more challenging when the baseline increases and when some objects are close to the cameras. It is because doing so will increase the disparity between the left and right views, thus making it harder to encode the two views into one. We performed a stress test on our method by setting up the scene shown in Figure 14(i). It is a very challenging scene, since the crystal ball shown introduces strong non-Lambertian reflections and discontinuity, while being close to the viewpoints. Then, we tested the performance of our method on a sequence of stereo image pairs captured at viewpoints of increasing baselines, in which we treated the left-most image (at 0.0 cm) as the input left view and other images of increasing baselines from the left as the input right views.

Figure 14 (ii) plots the quality of the mononized and restored binocular frames produced by our method on stereo image pairs of increasing baselines. We can see from the plots that the quality of our results stays very high (> 38 dB) and varies only slightly for baseline less than 10 cm. When the baseline increases from 10 to 35 cm, the quality smoothly decreases but is still well above 30 dB. Further, when the baseline reaches 40 cm, the quality of the mononized and restored right views drops below 30 dB, for which the audiences may be able to aware of the artifacts.

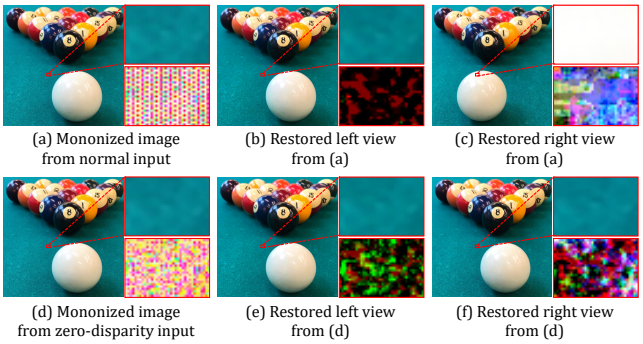



Fig. 15. The first row shows the mononized view (a) produced from a normal binocular input and left & right views (b & c) restored from it. The second row shows the mononized view (d) produced from a zero-disparity input and left & right views (e & f) restored from it. The right-hand sides of (a)-(f) present the blown-up regions (top) and difference images from the ground truths (bottom), which have been scaled-up "100" times for viewing.

*How is the stereo information encoded?* In our investigation, we first extensively zoom into (~30×) a small region in a mononized view that is occluded in the corresponding right view (Figures 15 (a) & (c)). We choose such a region, since the left and right views in the region are substantially different, so more stereo information has to be encoded in the region inside the mononized view. However, we cannot observe any abnormal pattern in the blown-up view of this small region (top-right image in Figure 15 (a)). Hence, we further compute the difference image in the region between the mononized view and input left view, and scale up the difference "100" times for better visualization. The bottom-right image in Figure 15 (a) shows the result, in which a regular dot pattern is revealed.

Intuitively, we suppose the stereo information is carried by the dot pattern. After the restoration, the dots no longer exist in the restored left and right views (b & c), since our decoding network has consumed them in the restoration process. To further verify our supposition, we feed a zero-disparity image pair (i.e., left view = right view) to our *encoding* network, to produce a mononized view

[9]https://en.wikipedia.org/wiki/Pupillary_distance
[10]https://en.wikipedia.org/wiki/Stereo_photography_techniques

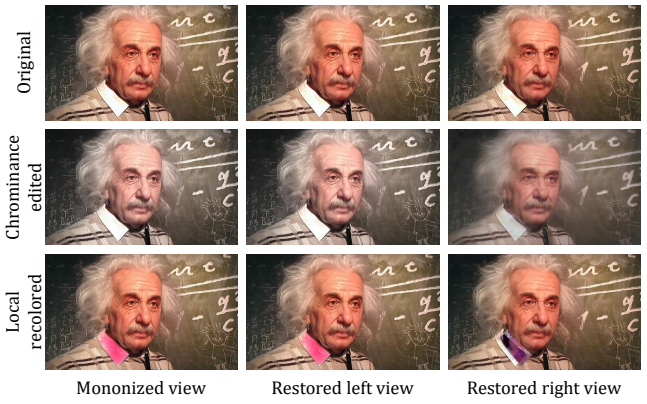



Fig. 16. Effects of manipulating the mononized view. Left column shows the original and manipulated mononized views, while the middle and right columns show the corresponding restored left and right views, respectively.

(d), as well as restoring from it a pair of left and right views (e & f). The dot patterns are absent in these results, thus revealing that the regular dots indeed encode the stereo information.

*Limitation.* Manipulations on the mononized view can hurt the restorability. For example, if we edit the chrominance in the mononized image, the manipulation effect could be somehow transferred to the restored left view but the restored right view could suffer from blurriness (second row in Figure 16). Similarly, locally re-coloring objects in the mononized image could lead to improper color changes in the corresponding image region in the right view (third row in Figure 16). It is because the manipulations ruin the stereo information visually-encoded in the mononized view, thus interfering the restoration of the binocular views by the decoding network.

## 8 CONCLUSION

We presented an innovative idea of *mono-nizing* binocular videos and a framework to effectively realize it by implicitly encoding the stereo information in a visual but nearly-imperceptible form inside the mononized videos. Our mononized videos allow us not only to impartially distribute and show them as ordinary monocular videos on existing video platforms but also to decode them back to binocular videos for stereo viewing, when a 3D display is available. Our technical contributions include an encoding-and-decoding framework with the pyramidal deformable fusion module to exploit long-range correspondences between the left and right views, a quantization layer to suppress the restoring artifacts, and the compression noise simulation module to resist the compression noise introduced by modern video codecs. Our framework is self-supervised. We formulate the objective with a monocular term, an invertibility term, and a temporal term, which are defined on the input binocular video to guide the network to produce the mononized and binocular videos. Extensive experiments were performed to show the quality of our results and backward compatibility of our method. Particularly, our mononized videos and restored binocular videos look no different from the original ones, and our mononized videos are compatible with various common video codecs, as demonstrated in the user study and experiments.

In the future, we would like to further extend our framework to mononize general multi-view videos acquired on the various recently-launched multi-camera phones. Also, we are interested in exploring editable mononized images/videos, such that global and local image manipulations applied on the mononized view can be transferred to all of the restored multi-views. Having editable mononized images/videos could enable the manipulation of multi-view scenes in a coherent and efficient manner.

## ACKNOWLEDGMENTS

This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under RGC General Research Fund (Project No. CUHK14201017 and CUHK14201918). The original binocular photo in Figure 1 is from *3D shoot* (Flickr); it is licensed under CC BY-NC-SA 2.0.

## REFERENCES

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V. Gool. 2017. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*.

Amir Atapour-Abarghouei and Toby P. Breckon. 2018. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2017. End-to-end Optimized Image Compression. In *International Conference on Learning Representations (ICLR)*.

Shumeet Baluja. 2017. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*.

Kiana Calagari, Mohamed Elgharib, Piotr Didyk, Alexandre Kaspar, Wojciech Matusik, and Mohamed Hefeeda. 2017. Data Driven 2-D-To-3-D Video Conversion for Soccer. *IEEE Transactions on Multimedia* 20, 3 (2017), 605–619.

Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. 1994. Two deterministic half-quadratic regularization algorithms for computed imaging. In *IEEE International Conference on Image Processing (ICIP)*.

Yue Chen, Debargha Mukherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, Ching-Han Chiang, Yunqing Wang, Paul Wilkins, Jim Bankoski, Luc N. Trudeau, Nathan E. Egge, Jean-Marc Valin, Thomas Davies, Steinar Midtskogen, Andrey Norkin, and Peter De Rivaz. 2018. An Overview of Core Coding Tools in the AV1 Video Codec. In *IEEE Picture Coding Symposium (PCS)*.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. 2019. Variable rate deep image compression with a conditional autoencoder. In *International Conference on Computer Vision (ICCV)*.

Xiaodong Cun, Feng Xu, Chi-Man Pun, and Hao Gao. 2018. Depth-Assisted Full Resolution Network for Single Image-Based View Synthesis. *IEEE Computer Graphics and Applications* 39, 2 (2018), 52–64.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable convolutional networks. In *International Conference on Computer Vision (ICCV)*.

Piotr Didyk, Tobias Ritschel, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. 2011. A perceptual model for disparity. *ACM Transactions on Graphics (SIGGRAPH)* 30, 4 (2011), 96:1–96:10.

Piotr Didyk, Tobias Ritschel, Elmar Eisemann, Karol Myszkowski, Hans-Peter Seidel, and Wojciech Matusik. 2012. A luminance-contrast-aware disparity model and applications. *ACM Transactions on Graphics (SIGGRAPH Asia)* 31, 6 (2012), 184:1–184:10.

David Eigen, Christian Puhrsch, and Rob Fergus. 2014. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *Advances in Neural Information Processing Systems*.

José M. Fácil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. 2019. CAM-Convs: Camera-Aware Multi-Scale Convolutions for Single-View Depth. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. DeepStereo: Learning to predict new views from the world's imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Taiki Fukiage, Takahiro Kawabe, and Shin'ya Nishida. 2017. Hiding of phase-based stereo disparity for ghost-free viewing without glasses. *ACM Transactions on Graphics (SIGGRAPH)* 36, 4 (2017), 147:1–147:17.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research (IJRR)* 32, 11 (2013), 1231–1237.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Advances in Neural Information Processing Systems*.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. In *Advances in Neural Information Processing Systems*.

Petr Kellnhofer, Piotr Didyk, Karol Myszkowski, Mohamed M. Hefeeda, Hans-Peter Seidel, and Wojciech Matusik. 2016. GazeStereo3D: Seamless disparity manipulations. *ACM Transactions on Graphics (SIGGRAPH)* 35, 4 (2016), 1–13.

Petr Kellnhofer, Piotr Didyk, Szu-Po Wang, Pitchaya Sitthi-Amorn, William Freeman, Fredo Durand, and Wojciech Matusik. 2017. 3DTV at home: Eulerian-Lagrangian stereo-to-multiview conversion. *ACM Transactions on Graphics (SIGGRAPH)* 36, 4 (2017), 146:1–146:13.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Yi Lai, Qian Wang, and Yin Gao. 2017. Content-Based Scalable Multi-View Video Coding Using 4D Wavelet. *International Journal of Hybrid Information Technology* 10, 8 (2017), 91–100.

Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus H. Gross. 2010. Nonlinear disparity mapping for stereoscopic 3D. *ACM Transactions on Graphics (SIGGRAPH)* 29, 4 (2010), 75:1–75:10.

Thomas Leimkühler, Petr Kellnhofer, Tobias Ritschel, Karol Myszkowski, and Hans-Peter Seidel. 2018. Perceptual real-time 2D-to-3D conversion using cue fusion. *IEEE Transactions on Visualization & Computer Graphics* 24, 6 (2018), 2037–2050.

Yue Li, Dong Liu, Houqiang Li, Li Li, Zhu Li, and Feng Wu. 2019b. Learning a Convolutional Neural Network for Image Compact-Resolution. *IEEE Transactions on Image Processing (TIP)* 28, 3 (2019), 1092–1107.

Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T. Freeman. 2019a. Learning the depths of moving people by watching frozen people. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Miaomiao Liu, Xuming He, and Mathieu Salzmann. 2018. Geometry-aware deep network for single-image novel view synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Stephen Lombardi, Tomas Simon, Jason M. Saragih, Gabriel Schwartz, Andreas M. Lehrmann, and Yaser Sheikh. 2019. Neural volumes: learning dynamic renderable volumes from images. *ACM Transactions on Graphics (SIGGRAPH)* 38, 4 (2019), 65:1–65:14.

Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. 2018. Single view stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bruhanth Mallik and Akbar Sheikh Akbari. 2016. HEVC based multi-view video codec using frame interleaving technique. In *International Conference on Developments in eSystems Engineering (DeSE)*.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv:2003.08934 [cs.CV]

Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, and Yaowu Xu. 2015. A technical overview of VP9—The latest open-source video codec. *SMPTE Motion Imaging Journal* 124, 1 (2015), 44–54.

Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. 2017. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 2019. 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (SIGGRAPH Asia)* 38, 6 (2019), 184:1–184:15.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2017. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision (ECCV)*.

Mark A. Robertson and Robert L. Stevenson. 2005. DCT quantization noise in compressed images. *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)* 15, 1 (2005), 27–38.

Steven Scher, Jing Liu, Rajan Vaish, Prabath Gunawardane, and James Davis. 2013. 3D+2DTV: 3D displays with no ghosting for viewers without glasses. *ACM Transactions on Graphics (SIGGRAPH)* 32, 3 (2013), 21:1–21:10.

Donald J. Schuirmann. 1987. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics* 15, 6 (1987), 657–680.

Guillaume Seguin, Karteek Alahari, Josef Sivic, and Ivan Laptev. 2015. Pose Estimation and Segmentation of People in 3D Movies. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)* 37, 8 (2015), 1643–1655.

Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 2020. 3D Photography using Context-aware Layered Depth Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the Boundaries of View Extrapolation with Multiplane Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)* 22, 12 (2012), 1649–1668.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gerhard Tech, Ying Chen, Karsten Müller, Jens-Rainer Ohm, Anthony Vetro, and Ye-Kui Wang. 2016. Overview of the multiview and 3D extensions of high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)* 26, 1 (2016), 35–49.

Anthony Vetro. 2010. Frame compatible formats for 3D video distribution. In *IEEE International Conference on Image Processing (ICIP)*.

Anthony Vetro, Thomas Wiegand, and Gary J. Sullivan. 2011. Overview of the stereo and multiview video coding extensions of the H. 264/MPEG-4 AVC standard. *Proc. IEEE* 99, 4 (2011), 626–642.

Yingqian Wang, Longguang Wang, Jungang Yang, Wei An, and Yulan Guo. 2019b. Flickr1024: A Large-Scale Dataset for Stereo Image Super-Resolution. In *International Conference on Computer Vision Workshops*.

Zihan Wang, Neng Gao, Xin Wang, Ji Xiang, Daren Zha, and Linghui Li. 2019a. HidingGAN: High Capacity Information Hiding with Generative Adversarial Network. *Computer Graphics Forum* 38, 7 (2019), 393–401.

Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers*. IEEE.

Eric Wengrowski and Kristin Dana. 2019. Light Field Messaging With Deep Photographic Steganography. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology (CSVT)* 13, 7 (2003), 560–576.

Menghan Xia, Xueting Liu, and Tien-Tsin Wong. 2018. Invertible grayscale. *ACM Transactions on Graphics (SIGGRAPH Asia)* 37, 6 (2018), 246:1–246:10.

Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Deep3D: Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks. In *European Conference on Computer Vision (ECCV)*.

Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. 2019. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (SIGGRAPH)* 38, 4 (2019), 76:1–76:13.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (SIGGRAPH)* 37, 4 (2018), 65:1–65:12.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. HiDDeN: Hiding Data With Deep Networks. In *European Conference on Computer Vision (ECCV)*.

Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. 2019. Deformable ConvNets v2: More deformable, better results. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

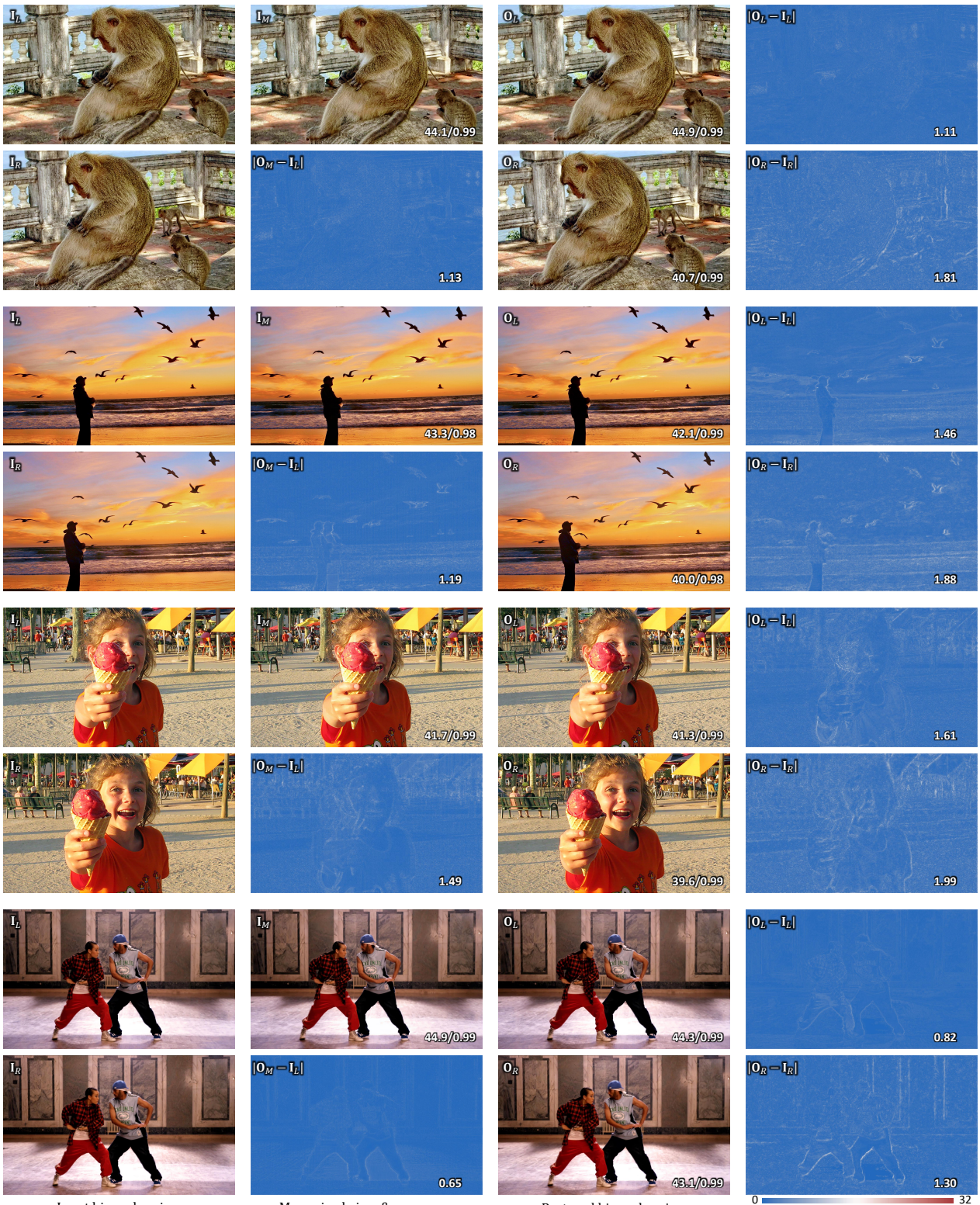



Fig. 17. Four sets of example results (every pair of rows) produced by our method. In each result, we show the input left & right views (1st column), generated mononized view and its difference map from the input left view (2nd column), restored binocular views (3rd column), and their difference maps from the inputs (4th column). In each result, the numbers show PSNR and SSIM, while in each error map, the number shows the mean absolute difference (scale of [0,255]).