Invertible Image Signal Processing

Yazhou Xing*, Zian Qian*, Ji Zhou, and Qifeng Chen, *Member, IEEE*

Abstract—Unprocessed RAW data is a highly valuable image format for image editing and computer vision. However, since the file size of RAW data is huge, most users can only get access to processed and compressed sRGB images. To bridge this gap, we design an Invertible Image Signal Processing (InvISP) pipeline, which not only enables rendering visually appealing sRGB images but also allows recovering nearly perfect RAW data. Due to our framework's inherent reversibility, we can reconstruct realistic RAW data instead of synthesizing RAW data from sRGB images without any memory overhead. We also integrate a differentiable JPEG compression simulator that empowers our framework to reconstruct RAW data from JPEG images. Extensive quantitative and qualitative experiments on two DSLR cameras demonstrate that our method obtains much higher quality in both rendered sRGB images and reconstructed RAW data than alternative methods. We further extend our invertible ISP to videos where we show our proposed video normalizing flow method and video compression simulator can achieve desirable balance among RGB video rendering, RAW video reconstruction, and temporal consistency. We collect a raw video dataset and evaluate our InvISP with different designs under different video compression settings. Our source codes are publicly available at https://github.com/yzxing87/Invertible-ISP.

Index Terms—Image Signal Processing, Invertible Neural Networks, Raw Image Reconstruction

1 Introduction

Professional photographers can choose to process RAW images by themselves instead of RGB images to produce images with better visual effects as the RAW data captures unprocessed scene irradiance at each in 12-14 bits by a camera. Due to its linear relationship with scene irradiance, raw sensor data is also a better choice than RGB images for many image editing and computer vision tasks, such as photometric stereo, intrinsic image decomposition, image denoising, reflection removal, and image super resolution [4], [9], [22], [35], [39], [50], [51], [63]. However, accessing RAW images can be quite hard due to their memory-demanding property: RAW images may be discarded during the process of data storing, transferring, and sharing. In this paper, we are interested in the question: can users get access to the real RAW data without explicitly storing it?

Due to the great advantages of RAW images, there have been many approaches to provide the mapping from sRGB images to their RAW counterparts [2], [9], [37], [40], [46], [61]. Nguyen et al. [40] suggest explicitly storing the parameters of sRGB-RAW mapping functions into the JPEG metadata for the prospective RAW reconstruction. Brooks et al. [9] use the prior information of the cameras (e.g., color correction matrices and digital gains) to reverse the ISP step-by-step. Another line of work [37], [46], [61] follows the inverse order of ISP and proposes learning-based methods to synthesize RAW data from sRGB images. However, these methods still rely on the underlying lossy in-camera ISP pipeline, and the recovered RAW images are inaccurate and may be different from the original ones.

In this work, we propose a novel and effective learned solution by redesigning the camera image signal processing pipeline as an invertible one, which can be aptly called *invertible ISP (InvISP)*. Our learning-based InvISP enables rendering visually appealing RGB images in the forward process, and recovering nearly perfect quality raw sensor data from compressed RGB images through the inverse process. Our reconstructed RAW data is nearly identical with real RAW data and enables computer vision applications, such as image retouching and HDR reconstruction, as shown in Figure 1.

Designing an invertible ISP is not a trivial task for at least three reasons. First, some steps in the traditional ISP, such as denoising, tone mapping, and quantization, can lead to inevitable information lost from wide-range (12-bit or 14-bit) raw sensor data to 8-bit RGB images. Second, the invertible ISP should not produce visual artifacts such as halo and ghosting artifacts [24]. To render visually appealing sRGB images, denoising, demosaicing, color correction, white balance gain, tone mapping, and color enhancement must be designed carefully in ISP. Third, modern digital cameras store RGB images in the JPEG format, where the lossy compression process makes reconstructing high-quality RAW data highly challenging.

To overcome these challenges, we take advantage of the inherent reversibility of normalizing-flow-based models [16], [33] and design both the RAW-to-RGB and RGB-to-RAW mapping in our invertible ISP with one single invertible neural network. We deeply analyze the properties of traditional ISP and design specific modules that can not only well approximate the camera ISP but also reconstruct almost identical RAW data with the camera RAW data. Specifically, we design our model with the composition of a stack of affine coupling layers and utilize the invertible 1×1 convolution as the learnable permutation function between the coupling layers. Besides, to empower our model to recover realistic RAW data from JPEG images, we integrate a differentiable JPEG simulator into our invertible neural network. We leverage the idea from Fourier transforma-

• *Y. Xing, Z. Qian, J, Zhou, and Q. Chen (corresponding author) are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China.*
E-mail: {yxingag, zqianaa, jzhoubl}@connect.ust.hk, cqf@ust.hk
**Equal contribution*

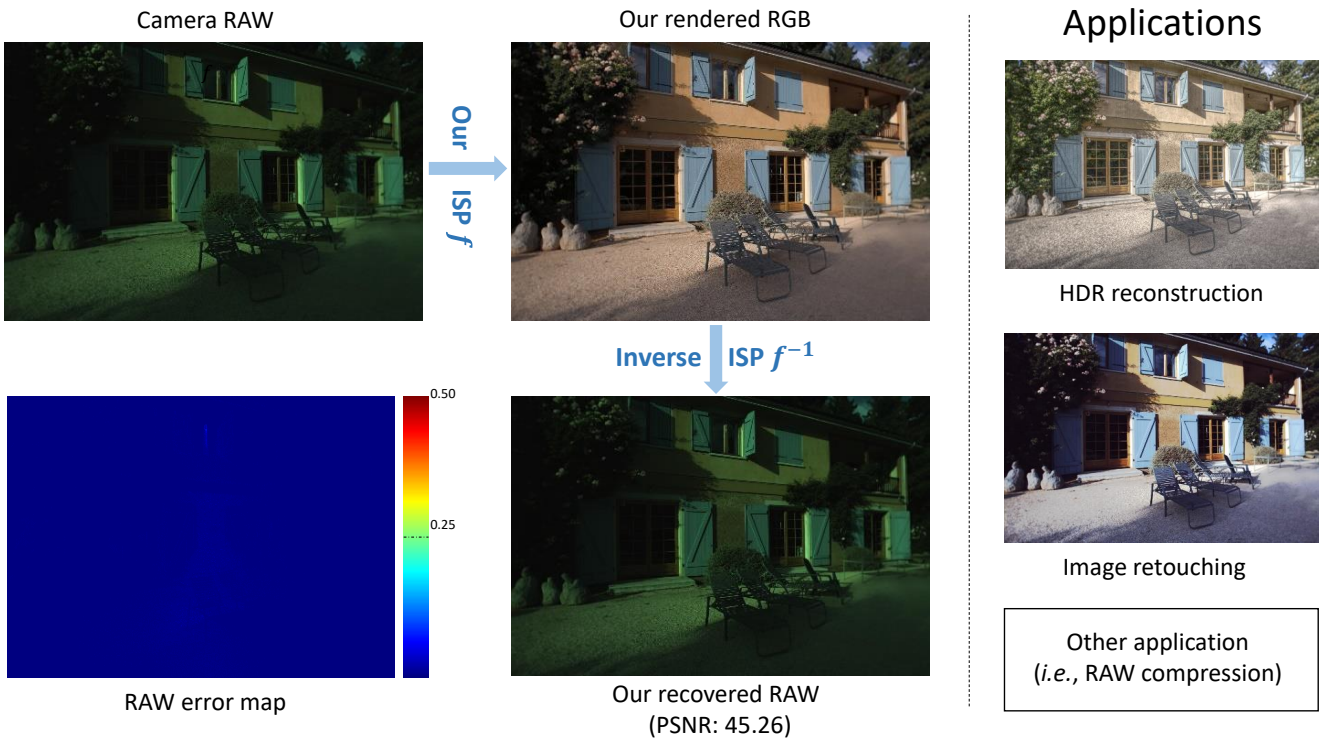

Fig. 1. Our ISP model can not only render visually pleasing RGB images but also recover RAW images that are nearly the same as the original RAW data. The recovered RAW data are valuable for photographers and benefit a number of computer vision tasks such as HDR reconstruction [42], image retouching [29], and RAW compression. Here, the RAW images are visualized with bilinear demosaicing.

tion to replace the non-differentiable quantization step in JPEG compression. Thus, our end-to-end InvISP framework bypasses traditional ISP modules and minimizes the information loss for the RAW data and RGB image conversion. We bidirectionally train our network to optimize the RGB and RAW reconstruction process jointly. We experimentally prove that our framework can recover much better RAW data than state-of-the-art baselines without sacrificing the RGB reconstruction performance.

In comparison to the conference version of this work [59], we present an extra technical contribution to RAW video reconstruction. We extend our invertible ISP to videos, where our method can achieve both high-quality RGB video rendering and RAW video reconstruction. Since there is no existing public RAW video dataset, we first collect a high-quality RAW video dataset with Sony RX 100M VI camera. Our RAW video dataset consists of 180 sequences of RAW videos that cover a diverse set of scenes. We preprocess the RAW videos through LibRAW to obtain high-quality RGB videos. Second, we propose a novel normalizing flow framework to enable invertible video processing. Our method can provide both leverage multiple frame information and preserve great level of temporal consistency. Third, we also integrate an effective video codec simulation module to handle the information loss during the video compression process. Our compression simulator can simulate both the H.264/AVC [55] and H.265/HEVC [52] video codecs. Moreover, we provide extensive ablation studies to analyze the key modules of our proposed invertible ISP framework, which provide valuable insights for further research.

Our contributions can be summarized as:

- We make the first attempt for RAW data reconstruction from the perspective of redesigning the camera ISP as an invertible one.
- We firstly introduce the normalizing flow into ISP modeling to enable both high quality RGB rendering and RAW reconstruction. Our method can address the information loss issue in ISP modules and is robust to the JPEG compression step. We demonstrate the effectiveness of our method on two DSLR cameras and show that our method outperforms state-of-the-art baselines to a large extent.
- We provide a detailed analysis of our invertible ISP key designs, including the influence of invertible block numbers, JPEG qualities, and loss functions.
- We extend our invertible ISP to videos where our proposed video normalizing flow method can achieve desirable balance among RGB video rendering, RAW video reconstruction, and temporal consistency. Our invertible framework is robust to the information lost due to video compression in digital cameras. To our best knowledge, we propose the first framework for accurate RAW video reconstruction from compressed video files.
- We exhibit three potential applications of our method, including RAW data compression, image retouching, and HDR reconstruction.

2 Related Work

2.1 RAW Image Reconstruction

Recovering RAW from sRGB images has been well-studied [2], [9], [37], [39], [40], [46], [61]. Nguyen et al. [40] encode the parameters in ISP into JPEG metadata with 64KB overhead and use them to reconstruct RAW from JPEG images. Brooks et al. [9] propose to invert the ISP pipeline step by step with camera priors. CIE-XYZ Net [2] proposes to recover RAW from sRGB images to the camera independent CIE-XYZ space. CycleISP [61] proposes to model the RGB-RAW-RGB data conversion cycle for synthesizing RAW from sRGB images. Unlike previous methods, we aim to fundamentally solve the RAW reconstruction problem by re-designing the camera ISP into an invertible one.

2.2 Image Signal Processing (ISP)

Image signal processing pipeline (ISP) aims at converting raw sensor data to human-readable RGB images [12], [13], [14], [22], [26], [34], [36], [50], [60], [63]. Heide et al. [26] merge the steps in the traditional ISP pipeline to avoid the accumulative error. Gharbi et al. [22] propose a method with end-to-end networks to learn RAW demosaicing and denoising jointly. Hasinoff et al. [25] propose a low-light imaging system for mobile devices. Other works [13], [50] focus on learning low-light enhancement ISP pipelines with CNNs. Zhang et al. [63] process RAW for super-resolution task with U-net [49] to preserve high-frequency information. CameraNET [36] splits the ISP into two learning stages for CNN. Unlike the encoder-decoder style network adopted in previous work, we demonstrate that invertible neural networks own great potential for ISP pipeline and enable accurate RAW reconstruction.

The ISP on videos concerns about the conversion process of raw data on videos [10], [38], [41], [53]. Buades et al. [10] present the joint demosaicking and denoising algorithm for raw video sequences with a spatio-temporal patch method. Tassano et al. [53] introduce a FastDVDnet based video denoising algorithm to achieve fast runtimes and the ability to handle different noise levels. Paliwal et al. [41] focus on denoising low-light raw videos in multiple stages with the adversarial loss and gradient mask. Maggioni et al. [38] propose a new multi-stage video denoising algorithm to reduce the computational complexity and memory requirements with recurrent spatio-temporal fusion.

2.3 Invertible Neural Networks

Normalizing flow-based invertible neural networks [15], [16], [27], [33] have become a popular choice in image generation tasks. Normalizing flow transforms a simple posterior distribution to a complex real-world distribution through a series of invertible transformations. NICE [15] is the first learning-based normalizing flow framework with the proposed additive coupling layers. RealNVP [16] modifies the additive coupling layer to both multiplication and addition, and composes the coupling layer in an alternating pattern such that all the inputs can be altered with equal chance. Kingma et al. [33] propose ActNorm layer and generalize channel-shuffle operations with invertible 1×1 convolution. Flow++ [27] modifies the affine coupling layer to logistics mixture CDF coupling flows and applies self-attention module.

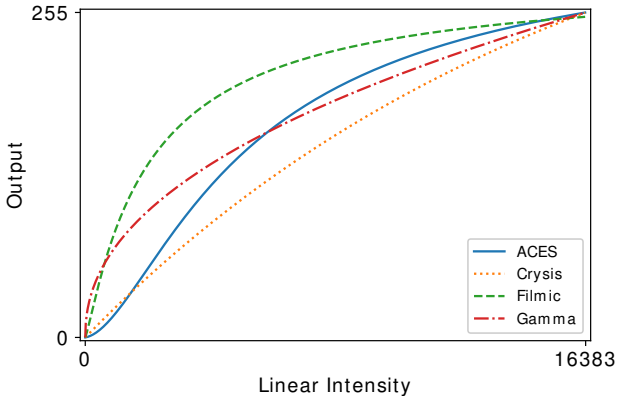

Fig. 2. Some popular tone mapping curves used in games and industries [7], [47]. Although the tone mapping function itself is lossless, the following quantization causes a great loss of information in over-exposed and under-exposed pixels. For instance, in a 14-bit linear RAW image, the pixel intensity lies in [16313, 16383] will all be quantized to the maximum pixel intensity 255 of an 8-bit RGB image.

2.4 Video Compression Codecs

Video codecs compress or decompress digital videos for better transmitting or recording [5], [6], [19], [21], [48], [56]. Bestagini et al. [5], [6] aim at identifying coding-based footprints of codecs in double compressed videos to recover the compression history. Wu et al. [56] focus on repeated deep image interpolation and generation in the video compression process and raise an end-to-end codec based on deep learning methods. Rippel et al. [48] proposes a relatively new video codec algorithm with a novel video compression architecture and spatial rate control framework based on machine learning. Esakki et al. [19] raise a VMAF-driven adaptive video encoding method for a large field of video codecs to maximize the quality of videos with comprehensive performance evaluation and subjective evaluations.

3 Traditional ISP Analysis

Modern digital cameras apply a series of operations, which form the image signal processing pipeline (ISP), to render RAW data to human-readable RGB images. These operations include white balance, demosaicing, denoising, color space transformation, tone mapping, and others [31]. Traditionally, every step of an ISP needs labor-intensive tuning for specific cameras, and inverting the traditional ISP steps is quite challenging. In this section, we analyze the existing modules with information loss in the traditional ISP. We show that the lossy steps in traditional ISP restrict the RAW reconstruction performance of a series of works [9], [40], [61] that aim at synthesizing RAW from sRGB images. Different from previous works, we re-design the ISP into an end-to-end invertible one that can bypass the traditional modules to minimize information loss during the RAW data and JPEG image (or MP4 video) conversion, which further enables recovering high-quality RAW data.

3.1 Quantization and tone mapping

Some ISP steps like demosaicing and gamma compression may involve float-point operations, and thus quantization is inevitable to transform the data into the integer type. For instance, the rounding function can bring $(-0.5, 0.5)$ intensity error to a pixel in theory. In the context of ISP, however, the tone mapping step can enlarge the intensity error much greater than ± 0.5. The tone mapping curve is usually designed as S-curve that compresses the high-intensity value and low-intensity value more than mid-intensity values [7], [47]. As illustrated in Figure 2, for a 14-bit raw image, gamma compression makes pixel intensity at $[16313, 16383]$ all be rounded to the max intensity 255 after normalized to $(0, 255)$. This step may cause a 0.004 RMSE error at this single pixel. Thus, it is challenging for existing works [9], [40], [61] to directly synthesize the 14-bit RAW data from its 8-bit sRGB counterparts, especially at the over-exposed regions. We show the comparison of our recovered RAW with previous works in Figure 6. Our method can preserve much more detail of RAW data, even at high-intensity pixels.

3.2 Out-of-range value clipping

Value clipping is a common step to normalize the raw value within a reasonable range, which may happen after color space transformation, demosaicing, denoising, and tone mapping [1], [18], [20], [44]. Most commonly used value clipping operation is like $\min(\max(x, 0), 1)$, which will discard the out-of-range pixels at over- and under-exposed regions. Note that this restricts the image capacity for further adjustment. Moreover, traditional ISPs are manually tuned in isolation by experts, which accumulates the clip error among ISP steps to bring further information lost. Our end-to-end pipeline jointly optimizes all the ISP steps and alleviates the clip error accumulation problem to recover more realistic RAW images.

3.3 JPEG compression and video compression

Modern digital cameras store RGB images in JPEG format, whose information loss further brings challenges to RAW image reconstruction. JPEG encoding pipeline consists of four main steps: color space transformation, discrete cosine transformation (DCT), quantization, and entropy encoding [43]. In reality, quantization is the only lossy and non-differentiable step in JPEG compression. Note that the JPEG information loss is quite hard to reverse. Thus we take a compromised step by integrating the JPEG compression procedure into our network optimization process to alleviate the information loss. To achieve this, we design a differentiable JPEG simulator by carefully simulating the JPEG compression procedure and replacing the quantization step with differentiable Fourier transformations.

Similarly, due to the great storage and bandwidth usage, videos are usually stored in compressed format. H.264/AVC [55] and H.265/HEVC [52] are the two most commonly used video compression methods. The information loss of these algorithms also brings challenges to RAW video reconstruction. H.264/AVC and H.265/HEVC share a similar compression pipeline, which mainly includes: I, P, B frame selection, macroblock dividing, inter-frame prediction (motion prediction and motion compensation), intra-frame prediction, discrete cosine transformation (DCT), quantization, and entropy encoding. Similar to JPEG compression, quantization is the only step that involves the information loss in H.264/AVC and H.265/HEVC. Since the full H.264/AVC and H.265/HEVC algorithms are hard to fully reproduce in our learning-based framework, we design an algorithm to simulate the key step in H.264/AVC and H.265/HEVC. Specifically, we propose a quality-aware compression simulation module to simulate the inter-frame prediction, intra-frame prediction, transformation, and quantization. By utilizing the CNS module, our method is able to adapt to the information loss by itself.

4 METHOD

4.1 Invertible Image Signal Processing (InvISP)

We denote the RAW data space as $\mathcal{X}$ and sRGB data space as $\mathcal{Y}$. Our goal is to find the invertible and bijective function which can map the data point from RAW data space to sRGB data space, denoted as $f : \mathcal{X} \to \mathcal{Y}$. To achieve this, classical neural networks need two separate networks to approximate $\mathcal{X} \to \mathcal{Y}$ and $\mathcal{Y} \to \mathcal{X}$ mappings respectively, which leads to inaccurate bijective mapping and may accumulate the error of one mapping into the other. We take an alternative method and use the affine coupling layers in [16], [33] to enable invertibility of one single network. We design our invertible ISP with the compostition of a stack of invertible and tractable bijective functions $\{f_i\}_{i=0}^{k}$, i.e., $f = f_0 \circ f_1 \circ f_2 \circ \cdots \circ f_k$. For a given observed data sample $\mathbf{x}$, we can derive the transformation to target data sample $\mathbf{y}$ through

$$\mathbf{y} = f_0 \circ f_1 \circ f_2 \circ \cdots \circ f_k(\mathbf{x}), \tag{1}$$

$$\mathbf{x} = f_k^{-1} \circ f_{k-1}^{-1} \circ \cdots \circ f_0^{-1}(\mathbf{y}). \tag{2}$$

The bijective model f_i is implemented through affine coupling layers. In each affine coupling layer, given a D dimensional input $\mathbf{m}$ and $d < D$, the output $\mathbf{n}$ is calculated as

$$\mathbf{n}_{1:d} = \mathbf{m}_{1:d}, \tag{3}$$

$$\mathbf{n}_{d+1:D} = \mathbf{m}_{d+1:D} \odot \exp\left(s\left(\mathbf{m}_{1:d}\right)\right) + t\left(\mathbf{m}_{1:d}\right), \tag{4}$$

where s and t represent scale and translation functions from $R^d \mapsto R^{D-d}$, and $\odot$ is the Hadamard product. Note that the scale and translation functions are not necessarily invertible, and thus we realize them by neural networks.

As stated in [16], the coupling layer leaves some input channels unchanged, which greatly restricts the representation learning power of this architecture. To alleviate this problem, we firstly enhance [58] the coupling layer (3) by

$$\mathbf{n}_{1:d} = \mathbf{m}_{1:d} + r(\mathbf{m}_{d+1:D}), \tag{5}$$

where r can be arbitrary function from $R^{D-d} \mapsto R^d$. The inverse step is easily obtained by

$$\mathbf{m}_{d+1:D} = \left(\mathbf{n}_{d+1:D} - t\left(\mathbf{n}_{1:d}\right)\right) \odot \exp\left(-s\left(\mathbf{n}_{1:d}\right)\right), \tag{6}$$

$$\mathbf{m}_{1:d} = \mathbf{n}_{1:d} - r(\mathbf{m}_{d+1:D}). \tag{7}$$

Next, we utilize the invertible 1×1 convolution proposed in [33] as the learnable permutation function to

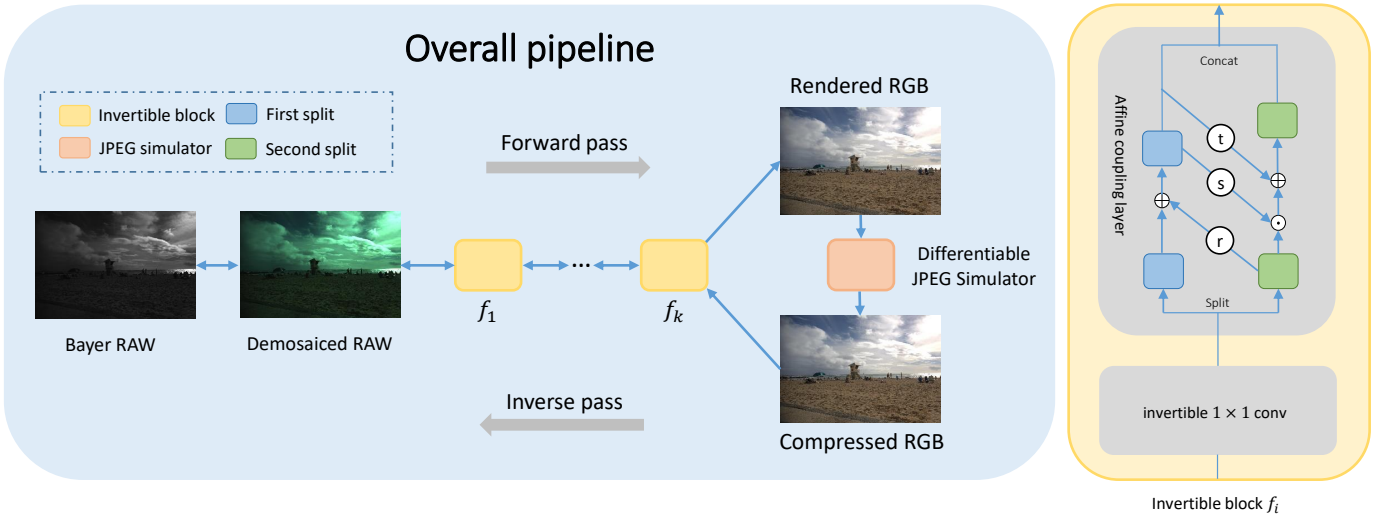

Fig. 3. Our invertible ISP (InvISP) framework. InvISP is composed of both forward and inverse passes. In the forward pass, the Bayer RAW is first bilinearly demosaiced and then transformed to an RGB image by a stack of bijective functions $\{f_i\}_{i=0}^{k}$. Our model integrates a differentiable JPEG simulator to account for compression information lost. During the training time, to invert the ISP, the backward pass takes a compressed RGB image as input and reverses all the bijective functions and the bilinear demosaicing to obtain the original RAW image. Note that the backward pass takes real JPEG images as input at test time. We illustrate the details of the invertible block on the right. r, s, and t are transformations defined in the bijective functions $\{f_i\}_{i=0}^{k}$.

reverse the order of channels for the next affine coupling layer.

We remove the spatial checkerboard mask as it brings no evident performance improvement [33]. We follow the implementation of [13] and disable batch normalization [30] and weight normalization used in [16]. For our image-to-image translation task, we directly learn the RAW-to-RGB mapping without explicitly modeling the latent distribution to stabilize the training process.

Note that the input size of invertible neural networks must be identical to the output size. Thus, we take the bilinear demosaiced RAW data as input, which will not destroy the RAW data quality, and reversing the bilinear demosaicing is trivial [9]. For the affine coupling layer, we split the input into two parts. We note that although three-channel input cannot be split evenly, the invertible 1×1 convolution ensures that unchanged components are updated in the next invertible block. Thus R, G, and B channels are still treated equally. We also do an online gamma correction (*i.e.*, without storing on disk) to RAW data to compress the dynamic range for faster convergence speed.

The forward pass of our InvISP produces the sRGB images, and the reverse pass aims at recovering realistic RAW data. We conduct bi-directional training with L_1 loss to optimize our framework:

$$L = ||f(\mathbf{x}) - \mathbf{y}||_1 + \lambda||f^{-1}(\mathbf{y}) - \mathbf{x}||_1, \tag{8}$$

where λ is the hyper-parameter used to balance the accuracy between RGB and RAW reconstruction. We set λ to 1 in our main experiments.

4.2 Invertible ISP on Videos

Compared with invertible ISP on images, invertible ISP on videos is more challenging since accurate RAW reconstruction and RGB rendering become harder and the network needs to maintain the temporal consistency at the same time. Single-image normalizing flow network performs unsatisfactory balance between the two goals since it ignores the temporal information which is vital for video reconstruction.

In this work, we propose a novel normalizing flow pipeline for invertible RAW video reconstruction and RGB video rendering. Specifically, instead of taking one frame each time as input for rendering and reconstruction, we feed W consecutive RAW frames to the forward process and W consecutive RGB frames to the inverse process, as shown in Figure. 4. We use this sliding-window scheme to encourage the network learn temporal information from the consecutive frames. Please note that in our design, we only aim at optimizing the center RGB and RAW frames within the sliding window. Thus, during training, we only optimize the loss on the center RGB and RAW frames and replace the other frames with ground-truth frames for computation. During inference, we use the reconstructed frames to act as the ground-truth frames. We utilize the bi-directional supervision to train the network:

$$L_{video} = \sum_{t=1}^{T} ||f(\mathbf{x_t}) - \mathbf{y_t}||_1 + \lambda||f^{-1}(\mathbf{y_t}) - \mathbf{x_t}||_1. \tag{9}$$

4.3 Quality-aware Compression Simulator

4.3.1 *JPEG compression*

Our goal is to train a robust invertible ISP that can tolerate the distortion by JPEG compression to recover accurate RAW. However, the JPEG compression algorithm is not differentiable, which can not be directly integrated into our end-to-end framework. Thus, we propose a differentiable JPEG simulator to enable our network robust to the JPEG compression through the optimization process. Since entropy encoding is lossless and goes after quantization, we

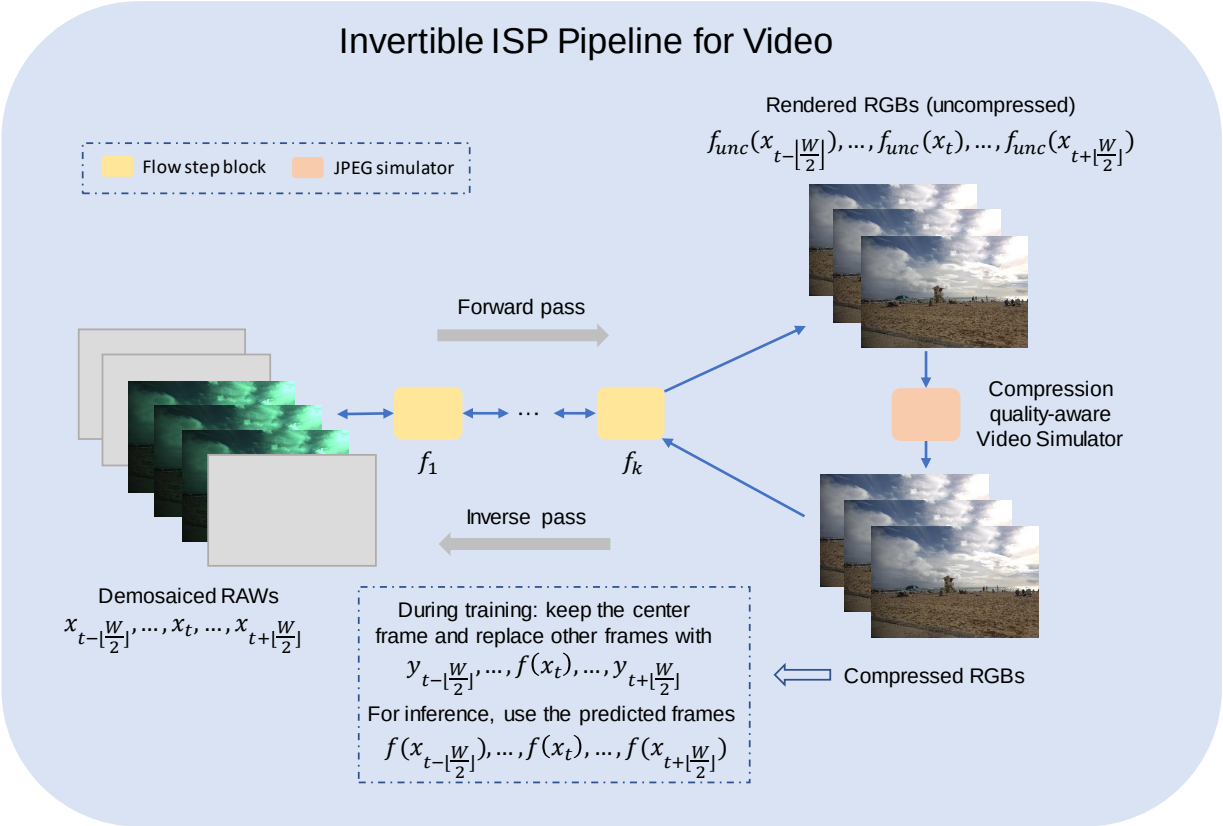

Fig. 4. Our invertible ISP framework for RGB video rendering and reconstruction. Different from its image counterpart, our invertible video pipeline needs to consider both the reconstruction quality and temporal consistency. Thus, for the forward pass, we sequentially feed K frames to the network to obtain the RGB frames. We also propose a more proper compression simulator for video codecs like H.264 and H.265. During training, in the reverse process, we replace the not-centered frames with the ground-truth RGB images to maintain the symmetric property of the sliding window. During inference phase, we use the predicted RGB frames as input to the reverse process.

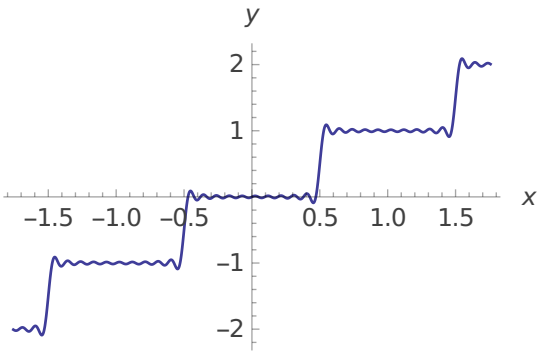

Fig. 5. The curve of our approximation rounding function for quantization in our differentiable JPEG simulator.

skip this step and only simulate color space transformation, DCT, and quantization steps.

To simulate the DCT process, we compute the DCT coefficients and split the input into 8×8 blocks. Then each block is multiplied by DCT coefficients to get the DCT map. In JPEG compression, the DCT map is divided by quantization tables and rounding to the integer type. Since the rounding function is not differentiable, we design a differentiable rounding function base on the Fourier series, which can be defined as

$$Q(I) = I - \frac{1}{\pi}\sum_{k=1}^{K}\frac{(-1)^{k+1}}{k}\sin(2\pi kI), \tag{10}$$

where I is the input map after divided by quantization tables in JPEG compression, and K is used for the tradeoff between approximation accuracy and computation efficiency. As K increases, the simulation function is closer to the real round function, but the running time will also increase. We empirically set K to 10. The rounding process is illustrated in Figure 5.

In the decoding phase of JPEG compression, I is multiplied by the quantization table. The inverse DCT and color space transformation are then applied to reconstruct the simulated JPEG images.

Discussion Differentiable rounding function is widely used in network quantization research. To fairly prove the effectiveness of our proposed rounding function, we also compare with the rounding function in [23], as shown in Table 1. Our method can achieve a better balance between RGB rendering and RAW reconstruction.

4.3.2 *Simulator for H.264 and H.265 compression*

Most modern digital cameras store captured RGB videos in a compressed format, which is usually streamed with lossy video codecs such as H.264/AVC [55] and H.265/HEVC [52]. H.264/AVC and H.265/HEVC consist of following steps:

Fig. 6. The qualitative comparison among UPI [9], CycleISP [61] and our method. UPI and CyleISP synthesize RAW data from 8-bit compressed RGB, which is inevitable to suffer from the information loss of traditional ISP. Unlike theirs, our model forms a RAW-RGB-RAW cycle and is inherently reversible to recover the realistic RAW image. The GT RAW image is visualized through bilinear demosaicing, and other RAW images are visualized through error maps. This figure is best viewed in the electronic version.

- I, P, B frame selection,
- inter-frame prediction,
- intra-frame prediction,
- DCT transformation and other transformations,
- quantization, and
- entropy encoding.

To account for the above key steps of video codecs, we aim to design a compression-quality-aware simulator. Specifically, we divide our simulator into inter-frame simulation and intra-frame simulation. We note that the frame selection of I, P, and B frame depends on the frame content and the video codecs design. To simplify this process, we perform randomized alternation between the inter-frame and intra-frame simulation. For intra-frame simulation, we mainly follow our design of a JPEG simulator and adjust the compression ratio via DCT tables. For inter-frame simulation, we use random motion jittering to simulate the optical flow estimation process. For each macroblock in an image, we sample a random dominant motion from [0,M], where M is a hyperparameter that represents the maximum motion. We then do some random jittering based on the random dominant motion between [0, 0.1]. We warp the macroblock to obtain the new ones that simulate the warped ones in intra-frame prediction. At last, we conduct compression on the estimated residual.

While our simulation is similar to the CNS module proposed in Hu et al. [28], we highlight the difference between our method and theirs.

- Instead of adopting the same compression simulator for all compression ratios, we adjust the simulator to adapt to different compression ratios by varying the DCT table.
- We adopt dominant motion to simulate the optical flow estimation process, which shows performance improvement over simulating the pixel-wise movement in CNS [28].

We quantatitively compare our method with CNS in Table 2.

5 Experiments

5.1 Datasets

RAW Image Datasets. We collect the Canon EOS 5D subset (777 image pairs) and the Nikon D700 subset (590 image pairs) from the MIT-Adobe FiveK dataset [11] as the training and test data. We train our model for each camera separately. We randomly split each of the two sets (Canon, Nikon) into training and test sets with a ratio of 85:15. We use the LibRaw library to process the RAW images to render ground-truth sRGB images. In general, LibRaw conducts most representative ISP steps in modern digital cameras to render sRGB images, including color space conversion, demosaicing, denoising, white balancing, exposure compensation, gamma compression, and global tone mapping.

RAW Video Datasets. Since there is no public large-scale RAW video dataset, we contribute a new dataset with 160 video frame sequences from different scenes captured

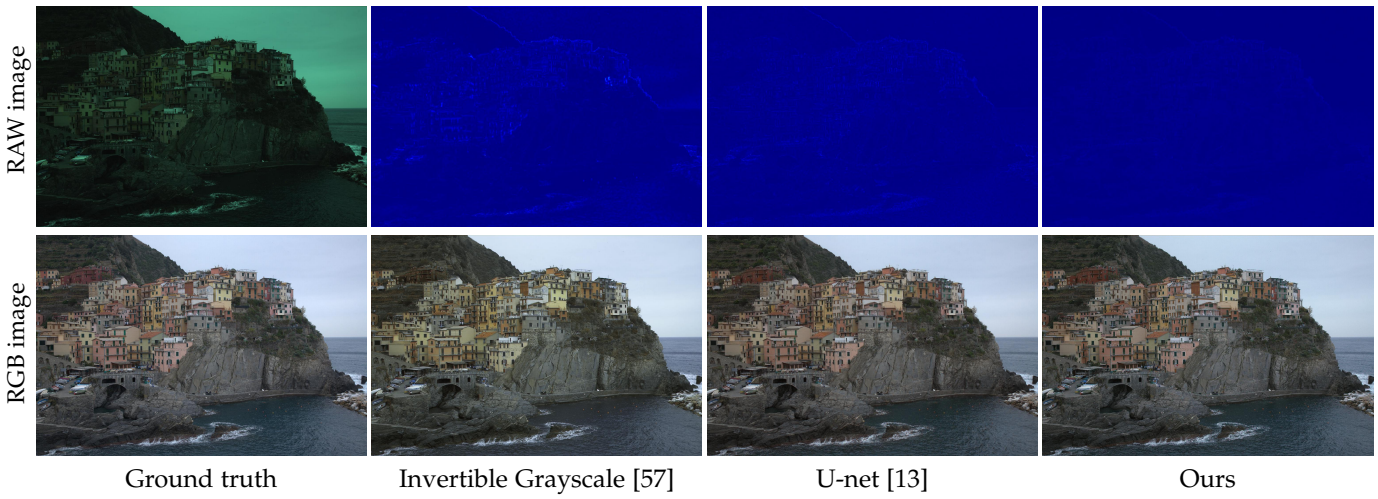

Fig. 7. Comparison with baselines. Invertible Grayscale [57] fails at learning a good balance between RGB rendering and RAW recovering, which results in relatively poor performance in both RGB and RAW images. The U-net [13] can render comparable RGB performance with ours but perform worse at RAW recovering. Our invertible ISP can both render visually pleasing RGB images and reconstruct realistic RAW data. The GT RAW is visualized through bilinear demosaicing, and other RAW images are visualized through error maps. This figure is best viewed in the electronic version.

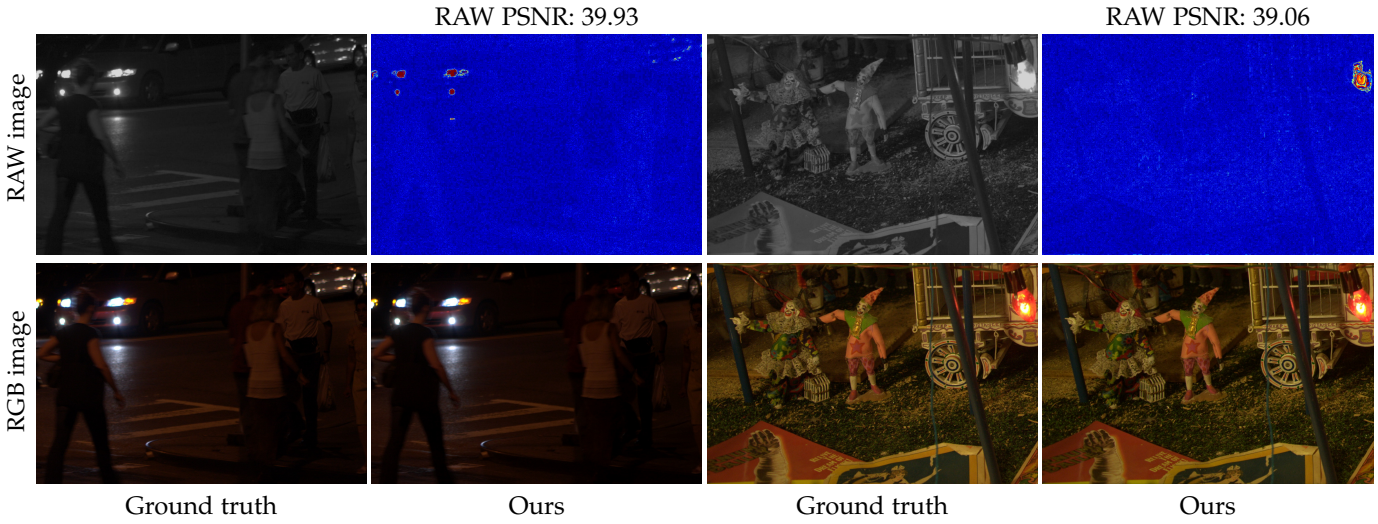

Fig. 8. Performance in low-light environments. Our approach also performs well in low-light environments.

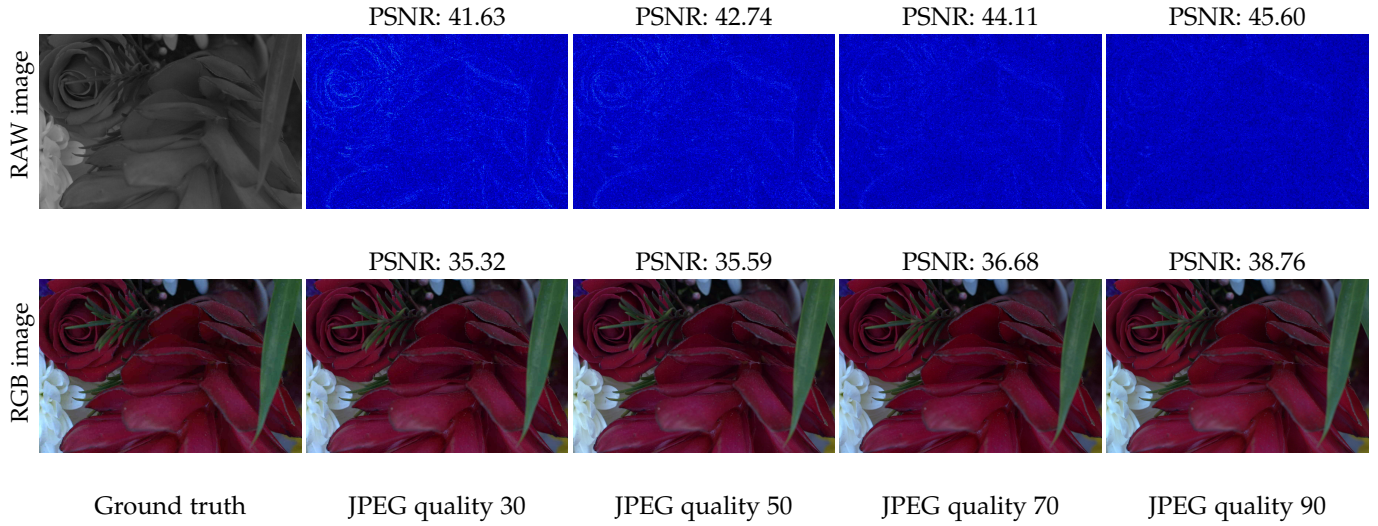

Fig. 9. Set different JPEG quality values for both the preprocess of the datasets and the training process. The JPEG quality of the ground-truth RGB image is 90. Low JPEG quality value will lead to unavoidable noise and loss of details, while a higher value can better the results for RAW and RGB images. This figure is best viewed in the electronic version.

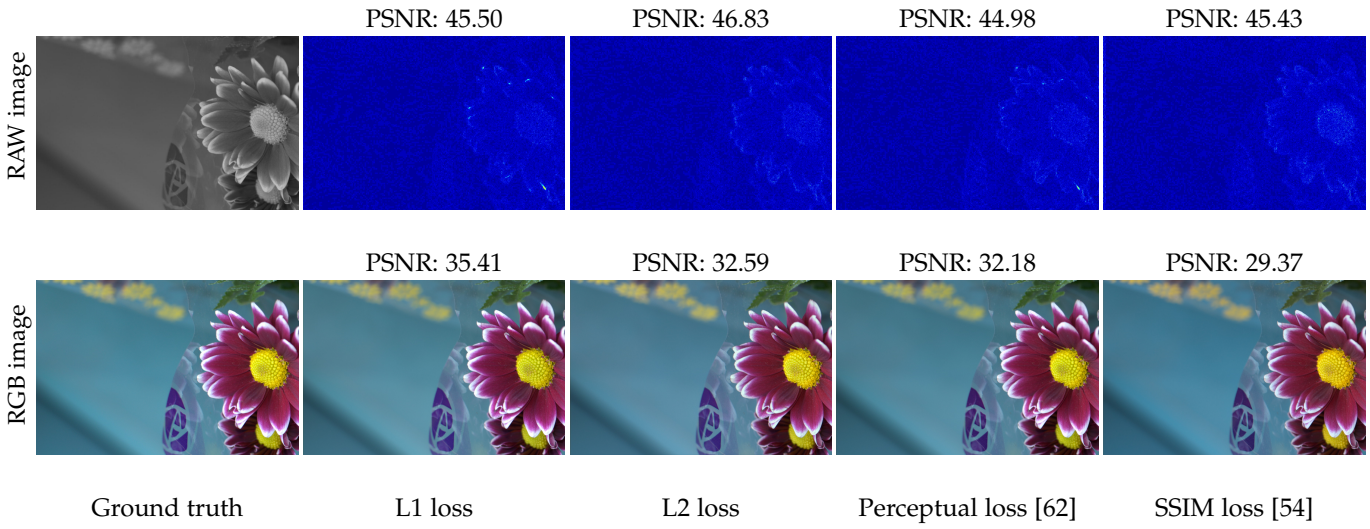

Fig. 10. Comparison of different loss functions, including L1 loss, L2 loss, Perceptual loss [62] and SSIM loss [54]. L1 loss has a steady performance with a delicate balance. L2 loss has a higher PSNR value for RAW data but worse for RGB results. Perceptual loss [62] and SSIM loss [54] are both underperform L1 loss for the image outputs. And they two are not applicable for the RAW data reconstruction in the training models. So we choose L1 loss as our preferred loss function. This figure is best viewed in the electronic version.

TABLE 1
Quantitative evaluation among our model and baselines. Various perceptual metrics show that our proposed ISP model outperforms all the baselines. Our method with JPEG simulation using proposed Fourier quantization outperforms the other two alternative models.

Method	NIKON D700			Canon EOS 5D		
	RGB		RAW	RGB		RAW
	PSNR	SSIM	PSNR	PSNR	SSIM	PSNR
UPI [9]	-	-	30.12	-	-	26.31
CycleISP [61]	-	-	30.19	-	-	34.48
InvGrayscale [57]	24.13	0.8258	33.28	28.22	0.8714	38.00
U-net [13]	36.48	0.9342	41.17	33.44	0.8893	41.14
Ours (w/o JPEG simulation)	37.44	0.9309	44.19	33.45	0.8923	45.73
Ours (JPEG with DSQ [23])	37.44	0.9467	**45.25**	33.15	0.8946	48.22
Ours (JPEG with Fourier)	**37.47**	**0.9473**	45.23	**33.61**	**0.9007**	**48.57**

TABLE 2
Quantitative evaluation among our model and baselines. Various perceptual metrics show that our proposed ISP model outperforms all the baselines. Our method with compression quality-aware simulation outperforms the existing video compression simulation method. Average PSNR is calculated with the average of RGB PSNR and RAW PSNR.

Method	CRF 23				CRF 35				CRF 41			
	RGB		RAW	Average	RGB		RAW	Average	RGB		RAW	Average
	PSNR	SSIM	PSNR	PSNR	PSNR	SSIM	PSNR	PSNR	PSNR	SSIM	PSNR	PSNR
Image flow (w/ JPEG simulation)	35.62	**0.842**	40.76	38.19	31.97	0.761	34.12	33.05	29.98	0.700	25.63	27.81
Image flow (w/ cns)	35.33	0.839	41.22	38.28	32.22	0.756	36.51	34.37	30.01	0.702	32.64	31.33
Image flow (w/ our simulation)	35.44	0.841	41.01	38.23	32.52	0.762	36.69	34.61	**30.33**	**0.703**	32.76	31.55
Video flow (w/o simulation)	35.71	0.834	38.97	37.34	32.53	0.755	34.42	33.48	30.05	0.699	28.51	29.28
Video flow (w/ cns)	**35.81**	0.838	**41.78**	**38.80**	32.51	0.760	**36.94**	34.73	30.18	0.698	32.99	31.59
Video flow (w/ our simulation)	35.74	0.837	41.50	38.62	**32.70**	**0.763**	36.79	**34.75**	30.23	0.701	**33.40**	**31.82**

by Sony RX 100M VI cameras. Each video contains 57-114 frames with the resolution of 2748×1936, and is captured with 24fps in a burst mode. During training, we randomly split the video datasets into a training set and a test set with a ratio of 87.5:12.5 (140 videos for training and 20 videos for testing). We use LibRaw Library to process these video frames to render sRGB video frames.

5.2 Implementation details

We utilize random crop, random rotation, and random flip as data augmentation to train our model. We preprocess the raw data using the white balance parameters provided

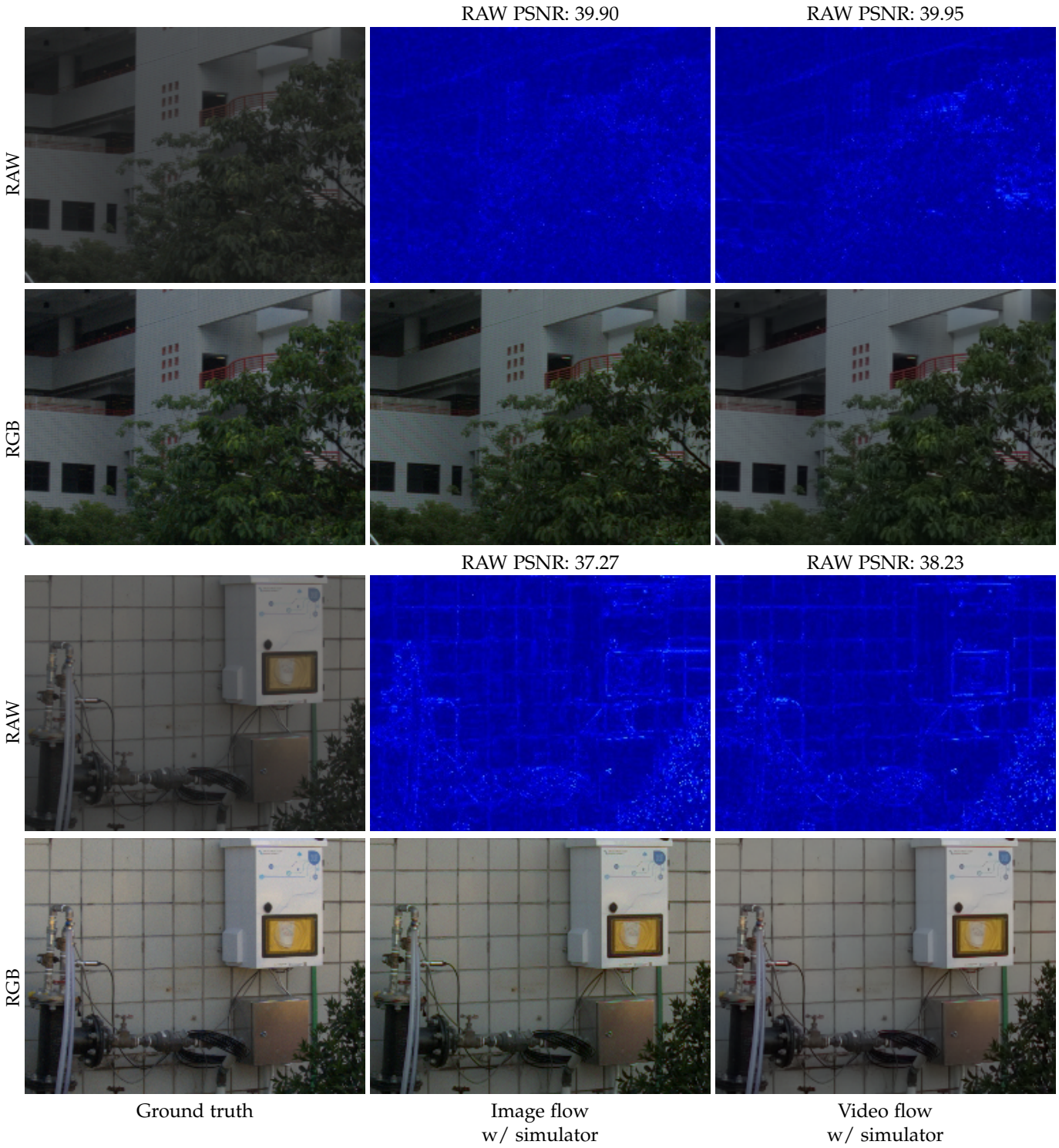

Fig. 11. Comparison between image flow and video flow methods.

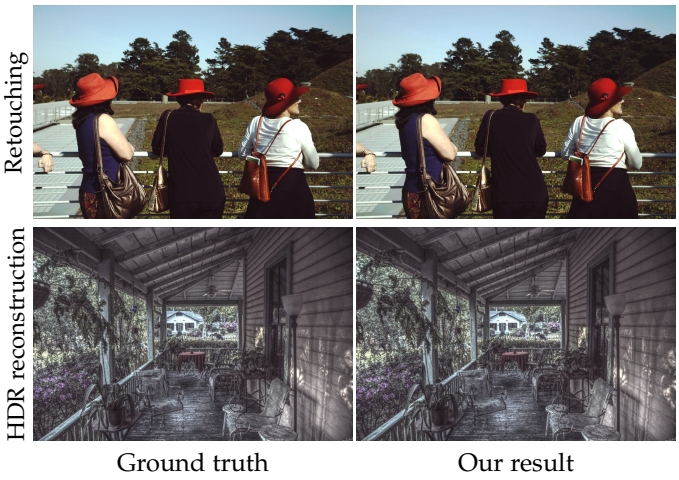

Fig. 12. Our model can enable image retouching [29] and HDR reconstruction [42] applications. Note that we use the pretrained model of [29] for retouching.

by camera metadata since estimating white balance directly from raw images is a research topic in itself [3]. To test the effectiveness of our JPEG simulator, we store the ground-truth RGB image into JPEG format, whose quality is set to 90 (Q=90, most representative JPEG quality in modern digital cameras). We also conduct experiments without preprocessing white balance and with another JPEG quality, whose quantitative results are accessible in the supplement. To test the effectiveness of our video compression simulator, we store the ground-truth RGB video into MP4 format, with the compression codec as H.264/AVC [55] and H.265/HEVC [52].

At test time, we first conduct the forward pass of our network to render RGB images and save them into JPEG images or MP4 videos. Then we load the saved JPEG images or MP4 videos and conduct the inverse step to recover RAW images.

5.3 Evaluation of invertible ISP on images

5.3.1 Baselines and controlled experiments

UPI. Brooks et al. [9] unprocess the sRGB images to synthesize high-quality RAW images for learned RAW denoising. They adopt camera priors to inverse ISP step-by-step, such as digital gain, tone mapping curves, white balance, and color correction matrices [9]. Since the metadata such as color correction matrix, white balance, and digital gain are camera dependent, we modified these parameters in their method to fit our dataset. We use their described method to estimate metadata for our datasets.

CycleISP. We select the state-of-the-art learning-based RAW synthesizing method, CycleISP [61], as another baseline for synthetic RAW direction. Note that their model has access to RGB images at test time, and thus we only need to compare with their synthesized RAW images. We directly utilize their pretrained model since their framework is trained on the MIT-Adobe FiveK dataset, and their proposed color attention unit can be generalized to different cameras.

U-net. U-net is a representative architecture for ISP in recent year publications [13], [63], thus we implement a encoder-decoder baseline using U-net [49]. Both the encoder and decoder are consist of an independent U-net. Same as [13], [63], we pack the Bayer pattern RAW into R-G-G-B channels for encoder input and utilize the depth-to-space operation to restore the RGB resolution. We utilize the same data augmentation strategies as our InvISP. We jointly train the encoder and decoder of our U-net baseline using L1 loss from scratch on all our datasets.

Invertible Grayscale. Invertible Grayscale [57] is a general framework to learn the forward and inverse mapping between two space, such as color-image space and grayscale-image space. The encoder of Invertible Grayscale takes a 3-channel RGB image as input and processes it to a single-channel grayscale image. The decoder recovers the original sRGB image with the same color from the grayscale image. Similar to their settings, we change the input from sRGB image to RAW data after bilinear demosaicing and set the output of the encoder to the 3-channel RGB image. Since the lightness loss function is not suitable for our tasks, we remove it for our experiments.

Ablation studies In the method for RAW data reconstruction, we analyze the parameters of block numbers, JPEG qualities, and loss functions which may affect the performance of our proposed model. In Table 3, we evaluate the performance of both subsets from the dataset [11] respectively under different parameter settings. We set different block numbers to the invertible networks, and use different JPEG qualities as parameters on the preprocessing of the dataset. In addition, we compare the performance adopting L1 loss with experiments using other loss functions, including L2 loss, perceptual loss [62] and SSIM loss [54].

5.3.2 Results

Quantitative results To quantitatively evaluate our method, we use PSNR and SSIM for rendered RGB images, and PSNR for recovered RAW images. The comparison with baselines is reported in Table 1. Compared with the RAW synthesizing method UPI and CycleISP, our model can recover more accurate RAW data, which is proved by more than 13 dB improvement of PSNR. The results are not surprising because lots of information lost in the ISP is quite hard to invert, which results in poor performance for synthetic RAW reconstruction methods. However, our InvISP can jointly optimize RGB rendering and RAW recovering process and thus is better to handle the information lost in quantization, JPEG compression, and saturated value clipping problem in ISP. For the Invertible Grayscale and the U-net baselines, the results indicate that our method contributes a better ISP as well as a stronger model for recovering RAW data. This is because using two separate networks for ISP and inverse ISP will cause the error accumulation problem, which further degrades the RAW reconstruction performance. Our methods take the inherent reversibility of invertible neural networks thus can recover higher-quality RAW images than baselines.

Qualitative results We show qualitative comparisons against baseline methods in Figure 6 and Figure 7. We also show the performance of our method under low-light environment in Figure 8. In Figure 6, the synthetic RAW by CycleISP and UPI differs a lot from ground truth RAW images, especially at over-exposed regions, which indicates that their model performs poorly to handle the information

TABLE 3
Ablation studies in our method for RAW image data reconstruction. The performance reaches a peak when a block number is a certain number. We also find that the increment of JPEG quality causes improved outputs. L1 loss outperforms all other loss functions by comparing PSNR values of RGB images; however, L2 loss has a better performance for RAW data reconstruction.

Block	NIKON D700			Canon EOS 5D		
Number	PSNR RGB	SSIM RGB	PSNR RAW	PSNR RGB	SSIM RGB	PSNR RAW
3	36.16	0.9377	45.06	36.35	0.9239	45.63
5	36.71	0.9407	44.83	36.88	0.9243	45.53
7	36.83	0.9476	**45.17**	**37.10**	**0.9246**	46.03
9	**37.17**	0.9450	45.15	36.55	0.9205	46.00
11	36.87	**0.9482**	44.51	35.84	0.9215	**46.58**

JPEG	NIKON D700			Canon EOS 5D		
Quality	PSNR RGB	SSIM RGB	PSNR RAW	PSNR RGB	SSIM RGB	PSNR RAW
30	34.92	0.9170	41.85	33.85	0.8854	39.92
50	35.12	0.9362	43.02	35.05	0.9089	43.05
70	35.65	0.9483	44.35	36.57	0.9266	43.86
90	36.76	0.9476	45.09	36.01	0.9241	45.86

Loss	NIKON D700			Canon EOS 5D		
Function	PSNR RGB	SSIM RGB	PSNR RAW	PSNR RGB	SSIM RGB	PSNR RAW
L1	**36.93**	0.9483	44.90	**36.54**	0.9256	46.49
L2	35.55	0.9385	**46.01**	36.23	0.9243	**46.62**
Perceptual	35.12	0.9306	44.26	35.39	0.9205	44.85
SSIM	35.44	**0.9486**	44.66	36.02	**0.9297**	44.74

loss of ISP. Our model, however, can recover the RAW information much better than synthetic RAW methods, even at challenging highlight pixels, which raises the potential for prospective photo editing tasks. In Figure 7, Invertible Grayscale fails to pursue a good balance between RGB rendering and RAW reconstruction. Our naive U-net baseline can achieve comparable performance in terms of RGB rendering but not perform well at RAW recovering. Our method reconstructs higher-quality RAW images on edges and over-exposed areas without sacrificing the RGB rendering performance. In Figure 8, our method can recover high-quality (PSNR over 39dB) RAW images and visually similar RGB images with the ground-truth, under the challenging low-light environment. This also opens the opportunities to research on low-light image enhancement with RAW sensor data [13].

5.4 Evaluation of invertible ISP on videos

5.4.1 Baselines

The effectiveness of our novel normalizing flow framework for videos. We compare our proposed video flow with several alternatives.

(1) Image flow without compression simulator. We utilize per-frame normalizing flow as in invertible image ISP, without any compression simulation. We denote this experiment as image flow (w/o simulation).
(2) Image flow with jpeg simulator. Still based on per-frame normalizing flow, we utilize jpeg compression simulator to account for the information loss due to compression, denoted as image flow (w/ JPEG).
(3) Video flow without compression simulator. We adopt our proposed video normalizing flow without any compression simulator to verify its effectiveness. We denote it as video flow (w/o simulation).
(4) Video flow with our video simulator. We use our proposed video normalizing flow with our designed video compression simulator to train the network. This is the full model of our framework, denoted as video flow (w/ our simulation).

Comparison with alternative compression simulators. We compare our proposed video compression simulator with two alternative methods.
(1) Video flow with jpeg compression simulator. We utilize the jpeg simulator for our video flow framework, denoted as video flow (w/ JPEG).
(2) Video flow with CNS simulator [28]. We utilize the video compression simulator proposed in [28] to verify the effectiveness of our proposed compression simulator.
(3) Image flow with video simulator. We use our proposed video compression simulator to make the network robust to video compression codecs. We denote it as image flow (w/ our simulation).

5.4.2 Qualitative and quantitative results

To quantitatively evaluate our method, we use PSNR and SSIM for rendered RGB images, and PSNR for recovered RAW images. The comparison with baselines is reported in Table 1. Compared with the RAW synthesizing method UPI

TABLE 4
The result of our CNS module on H.265/HEVC video compressor comparing to naive CNS. Our method is also robust to H.265/HEVC.

Method	CRF 23			CRF 35			CRF 41		
	RGB		RAW	RGB		RAW	RGB		RAW
	PSNR	SSIM	PSNR	PSNR	SSIM	PSNR	PSNR	SSIM	PSNR
Naive CNS	35.15	0.824	41.71	**32.21**	**0.740**	38.44	30.11	0.698	32.74
Our simulation	**35.21**	**0.826**	**41.81**	**32.21**	**0.740**	**38.51**	**30.13**	**0.704**	**33.27**

and CycleISP, our model can recover more accurate RAW data, which is proved by more than 13 dB improvement of PSNR. The results are not surprising because lots of information lost in the ISP is quite hard to invert, which results in poor performance for synthetic RAW reconstruction methods. However, our InvISP can jointly optimize RGB rendering and RAW recovering process and thus is better to handle the information lost in quantization, JPEG compression, and saturated value clipping problem in ISP. For the Invertible Grayscale and the U-net baselines, the results indicate that our method contributes a better ISP as well as a stronger model for recovering RAW data. This is because using two separate networks for ISP and inverse ISP will cause the error accumulation problem, which further degrades the RAW reconstruction performance. Our methods take the inherent reversibility of invertible neural networks thus can recover higher-quality RAW images than baselines.

5.4.3 Generalization to H.265/HEVC

Since H.265/HEVC is the state-of-the-art video compression method, we also demonstrate the robustness of our CNS module on H.265/HEVC. As illustrated in Table 4, our method can be adapted to H.265/HEVC, and have better reconstruction RAW quality than naive CNS on both CRF 23 and 35. Note that although our CNS module is designed based on H.264/AVC, H.265/HEVC shares the same compression pipeline with H.264/AVC and modified the method used in intra-frame prediction, inter-frame prediction, macro block dividing and DCT [32]. These modified methods only improve the computing efficiency and the performance of each step. Since the key idea of our CNS module is to simulate the key step of video compression pipeline, our method is robust to H.264/HEVC video compressor.

5.4.4 Ablation studies

We perform ablation studies on the number of pixel shift in our CNS module. The results in Table 5 illustrate that shift one pixel have the best RGB and reconstruction RAW quality. Therefore, we empirically set the number of pixel shift to one. Note that the best number of pixel shift may be variant for different RAW video dataset. Since our RAW video dataset is captured in burst mode with 15 fps with very slow camera movement, the motion between two adjacent frames are relatively small. Therefore, only shift one pixel can produce the best result. However, for the dataset with large movement, shift more pixel may produce better results.

6 Applications

6.1 RAW data compression

One important application of our framework is RAW data compression for cameras. Traditionally, users need to explicitly store RAW data for further applications. Using our technique, however, only JPEG images need to be stored, and users can reconstruct the corresponding RAW data from JPEG images. To evaluate the reduced file size, we calculate the compression ratio and the bit per pixel (BPP). The compression ratio C_{ratio} [45] is calculated by

$$C_{ratio} = \frac{\text{Uncompressed size}}{\text{Compressed size}} = \frac{B_{BMP}}{B_{JPEG}}, \tag{11}$$

where B_{BMP} is the file size of RAW data in BMP format and B_{JPEG} is the file size of rendered sRGB image in JPEG format. Note that B_{BMP} is calculated by [8]:

$$B_{BMP} = 54 + \frac{H \times W \times b}{8}, \tag{12}$$

where H, W and b are the height, width and the bit depth of the RAW data. We further compare our compression effectiveness with Adobe lossy DNG. As shown in Table 6, the file size is highly reduced, even compared with lossy DNG.

6.2 Image retouching

Professional photographers choose to retouch images from RAW data for better visual quality. We demonstrate that our recovered RAW can be directly taken as input for high-quality image retouching. We use an automatic deep learning based image retouching method Exposure [29] as an example. We preprocess the recovered RAW and ground truth RAW through demosaicing and white balancing, following the setting of the paper [29]. We directly utilize their pretrained model that is also trained on the MIT-Adobe FiveK dataset. As illustrated in Figure 12, our reconstructed RAW data has an indistinguishable visual quality to the RAW data captured by the camera.

6.3 HDR reconstruction and tone manipulation

Inferring a high dynamic range image from a single low dynamic range input is challenging [17], [18] since the information lost in saturated and under-exposed regions are hard to invert accurately. Our invertible ISP framework fundamentally alleviates these difficulties and thus enables single image HDR reconstruction. Further, the recovered HDR image can be tone mapped to display much more details than the original RGB input. In Figure 12, we use [42] as tone mapper to demonstrate the potential of our method.

TABLE 5
Ablation studies on the number of pixel shift in our simulation. Shifting 1 pixel gives the best result.

Method	CRF 23			CRF 35			CRF 41		
	RGB		RAW	RGB		RAW	RGB		RAW
	PSNR	SSIM	PSNR	PSNR	SSIM	PSNR	PSNR	SSIM	PSNR
Ours (shift 1 pixel)	35.43	0.821	**42.22**	32.21	**0.740**	**38.44**	**30.13**	0.704	**33.27**
Ours (shift 2 pixels)	**35.48**	0.823	42.15	**35.33**	**0.740**	38.41	30.01	0.703	33.24
Ours (shift 3 pixels)	35.50	0.825	42.07	32.34	0.739	37.85	30.12	**0.705**	32.97
Ours (shift 5 pixels)	35.18	0.823	41.99	32.15	0.738	37.73	29.81	0.699	32.81
Ours shift 8 pixels	35.15	**0.826**	41.95	32.14	0.739	37.88	29.74	0.699	32.74

TABLE 6
Comparison of the compression ratio of the file size and bit per pixel (BPP) between our method and lossy DNG. The file size is significantly reduced by our framework.

	Compression ratio ↑		BPP ↓	
Dataset	Lossy DNG	Ours	Lossy DNG	Ours
NIKON D700	1.61	34.98	8.73	0.4655
Canon EOS 5D	1.52	27.37	6.56	0.5237

7 Conclusion

We have proposed an end-to-end invertible image signal processing (InvISP) framework to generate visually pleasing RGB images and recover nearly perfect quality RAW data. We leverage the idea from invertible neural networks to design our invertible structure and integrate a differentiable JPEG simulator to enhance the network stability to JPEG compression. We use LibRaw to simulate ground-truth ISP on the MIT-Adobe FiveK dataset. We evaluate our method through comparisons with other frameworks and RAW data synthesis methods. We also demonstrate that our framework enables RAW data compression, image retouching, and HDR reconstruction tasks. We extend our invertible ISP to videos and show the first approach for RAW video reconstruction from compressed video files. Our proposed video normalizing flow method and video compression simulator can achieve state-of-the-art performance on RGB video rendering and RAW video reconstruction. We hope our method can inspire further research on ISP design and RAW image/video reconstruction.

Limitations While our approach can achieve state-of-the-art performance on raw image and video reconstruction, the computational cost of the our method is larger than the traditional ISP. This is mainly due to the inefficient computation of normalization flow method. Thus, our method is not suitable for real-time applications on low-end computation devices. We leave improving the computational efficiency as future work.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, 2018.

[2] Mahmoud Afifi, Abdelrahman Abdelhamed, Abdullah Abuolaim, Abhijith Punnappurath, and Michael S Brown. Cie xyz net: Unprocessing images for low-level computer vision tasks. *arXiv:2006.12709*, 2020.

[3] Jonathan T Barron. Convolutional color constancy. In *ICCV*, 2015.

[4] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

[5] Paolo Bestagini, Ahmed Allam, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Video codec identification. In *2012 IEEE International Conference on Acoustics, Speech and signal processing (ICASSP)*, pages 2257–2260. IEEE, 2012.

[6] Paolo Bestagini, Simone Milani, Marco Tagliasacchi, and Stefano Tubaro. Codec and gop identification in double compressed videos. *IEEE Transactions on Image Processing*, 25(5):2298–2310, 2016.

[7] R. Boitard, D. Thoreau, R. Cozot, and K. Bouatouch. Impact of temporal coherence-based tone mapping on video compression. In *EUSIPCO*, pages 1–5, 2013.

[8] Paul Bourke. Bmp image format. *BMP Files. July*, 1998.

[9] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *CVPR*, 2019.

[10] Antoni Buades and Joan Duran. Joint denoising and demosaicking of raw video sequences. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2172–2176, 2018.

[11] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR*, 2011.

[12] Chen Chen, Qifeng Chen, Minh N. Do, and Vladlen Koltun. Seeing motion in the dark. In *ICCV*, 2019.

[13] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *CVPR*, 2018.

[14] Marcos V Conde, Steven McDonagh, Matteo Maggioni, Aleš Leonardis, and Eduardo Pérez-Pellitero. Model-based image signal processors via learnable dictionaries. 2022.

[15] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: nonlinear independent components estimation. In *ICLR*, 2015.

[16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[17] Gabriel Eilertsen, Joel Kronander, Gyorgy Denes, Rafał K Mantiuk, and Jonas Unger. Hdr image reconstruction from a single exposure using deep cnns. *ACM transactions on graphics (TOG)*, 36(6):1–15, 2017.

[18] Yuki Endo, Yoshihiro Kanamori, and Jun Mitani. Deep reverse tone mapping. *ACM Transactions on Graphics (TOG)*, 36(6):177–1, 2017.

[19] Gangadharan Esakki, Andreas S Panayides, Venkatesh Jalta, and Marios S Pattichis. Adaptive video encoding for different video codecs. *IEEE Access*, 9:68720–68736, 2021.

[20] Alessandro Foi. Clipped noisy images: Heteroskedastic modeling and practical denoising. *Signal Processing*, 89(12):2609–2629, 2009.

[21] Mohammed Ghanbari. *Standard codecs: Image compression to advanced video coding*. Number 49. Iet, 2003.

[22] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[23] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *ICCV*, 2019.

[24] Gabriele Guarnieri, Stefano Marsi, and Giovanni Ramponi. High dynamic range image display with halo and clipping prevention. *IEEE Transactions on Image Processing*, 20(5):1351–1362, 2010.

[25] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016.

[26] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pajak, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. Flexisp: A flexible camera image processing framework. *ACM Transactions on Graphics (TOG)*, 33(6):1–13, 2014.

[27] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *ICML*, 2019.

[28] Wenbo Hu, Menghan Xia, Chi-Wing Fu, and Tien-Tsin Wong. Mononizing binocular videos. *ACM Transactions on Graphics (TOG)*, 39(6):228:1–228:16, 2020.

[29] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):26, 2018.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[31] Hakki Can Karaimer and Michael S Brown. A software platform for manipulating the camera imaging pipeline. In *ECCV*, 2016.

[32] Syed Ali Khayam. The discrete cosine transform (dct): theory and application. *Michigan State University*, 114:1–31, 2003.

[33] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

[34] Teresa Klatzer, Kerstin Hammernik, Patrick Knobelreiter, and Thomas Pock. Learning joint demosaicing and denoising based on sequential energy minimization. In *ICCP*, 2016.

[35] Chenyang Lei, Xuhua Huang, Mengdi Zhang, Qiong Yan, Wenxiu Sun, and Qifeng Chen. Polarized reflection removal with perfect alignment in the wild. In *CVPR*, 2020.

[36] Zhetong Liang, Jianrui Cai, Zisheng Cao, and Lei Zhang. Cameranet: A two-stage framework for effective camera ISP learning. *IEEE Transactions on Image Processing*, 30:2248–2262, 2021.

[37] Yu-Lun Liu, Wei-Sheng Lai, Yu-Sheng Chen, Yi-Lung Kao, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Single-image HDR reconstruction by learning to reverse the camera pipeline. In *CVPR*, 2020.

[38] Matteo Maggioni, Yibin Huang, Cheng Li, Shuai Xiao, Zhongqian Fu, and Fenglong Song. Efficient multi-stage video denoising with recurrent spatio-temporal fusion. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3465–3474, 2021.

[39] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *CVPR*, 2019.

[40] Rang MH Nguyen and Michael S Brown. Raw image reconstruction using a self-contained srgb–jpeg image with small memory overhead. *International journal of computer vision*, 126(6):637–650, 2018.

[41] Avinash Paliwal, Libing Zeng, and Nima Khademi Kalantari. Multi-stage raw video denoising with adversarial loss and gradient mask. In *2021 IEEE International Conference on Computational Photography (ICCP)*, pages 1–10, 2021.

[42] Sylvain Paris, Samuel W Hasinoff, and Jan Kautz. Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *ACM Transactions on Graphics (TOG)*, 30(4):68, 2011.

[43] William B Pennebaker and Joan L Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[44] Tobias Plotz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *CVPR*, 2017.

[45] Charles Poynton. *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[46] A. Punnappurath and M. S. Brown. Learning raw image reconstruction-aware deep image compressors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):1013–1019, 2020.

[47] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *SIGGRAPH*, 2002.

[48] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[50] Eli Schwartz, Raja Giryes, and Alex M Bronstein. Deepisp: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, 28(2):912–923, 2018.

[51] Boxin Shi, Zhe Wu, Zhipeng Mo, Dinglong Duan, Sai-Kit Yeung, and Ping Tan. A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo. In *CVPR*, 2016.

[52] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[53] Matias Tassano, Julie Delon, and Thomas Veit. Fastdvdnet: Towards real-time deep video denoising without flow estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1351–1360, 2020.

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

[55] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[56] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[57] Menghan Xia, Xueting Liu, and Tien-Tsin Wong. Invertible grayscale. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[58] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. Invertible image rescaling. In *ECCV*, 2020.

[59] Yazhou Xing, Zian Qian, and Qifeng Chen. Invertible image signal processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6287–6296, 2021.

[60] Xiangyu Xu, Yongrui Ma, and Wenxiu Sun. Towards real scene super-resolution with raw images. In *CVPR*, 2019.

[61] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Cycleisp: Real image restoration via improved data synthesis. In *CVPR*, 2020.

[62] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[63] Xuaner Zhang, Qifeng Chen, Ren Ng, and Vladlen Koltun. Zoom to learn, learn to zoom. In *CVPR*, 2019.

Yazhou Xing received the B.E. degree from Wuhan University in 2018. He is currently pursuing the Ph.D. degree with the Hong Kong University of Science and Technology. His research interests include computation photography, low-level computer vision and video processing.

Zian Qian received the B.E. degree from HKUST in 2020. He is currently pursuing the Master of Philosophy degree with the Hong Kong University of Science and Technology. His research interests include computation photography and low-level image and video processing.

Ji Zhou is currently pursuing the B.E. degree in computer engineering with the Hong Kong University of Science and Technology. His research interests include computer vision, big data technology, and embedded systems.

Qifeng Chen is an assistant professor of the Department of Computer Science and Engineering and the Department of Electronic and Computer Engineering at HKUST. He received his Ph.D. in computer science from Stanford University in 2017, and a bachelor's degree in computer science and mathematics from HKUST in 2012. He is named one of 35 Innovators under 35 in China in 2018 by MIT Technology Review. He won the Google Faculty Research Award 2018. He is the HKUST ACM programming faculty coach and won the 2nd place worldwide at the ACM-ICPC World Finals in 2011. He is a member of IEEE.