IICNet: A Generic Framework for Reversible Image Conversion

Ka Leong Cheng*, Yueqi Xie*, Qifeng Chen
The Hong Kong University of Science and Technology
{klchengad, yxieay}@connect.ust.hk, cqf@ust.hk

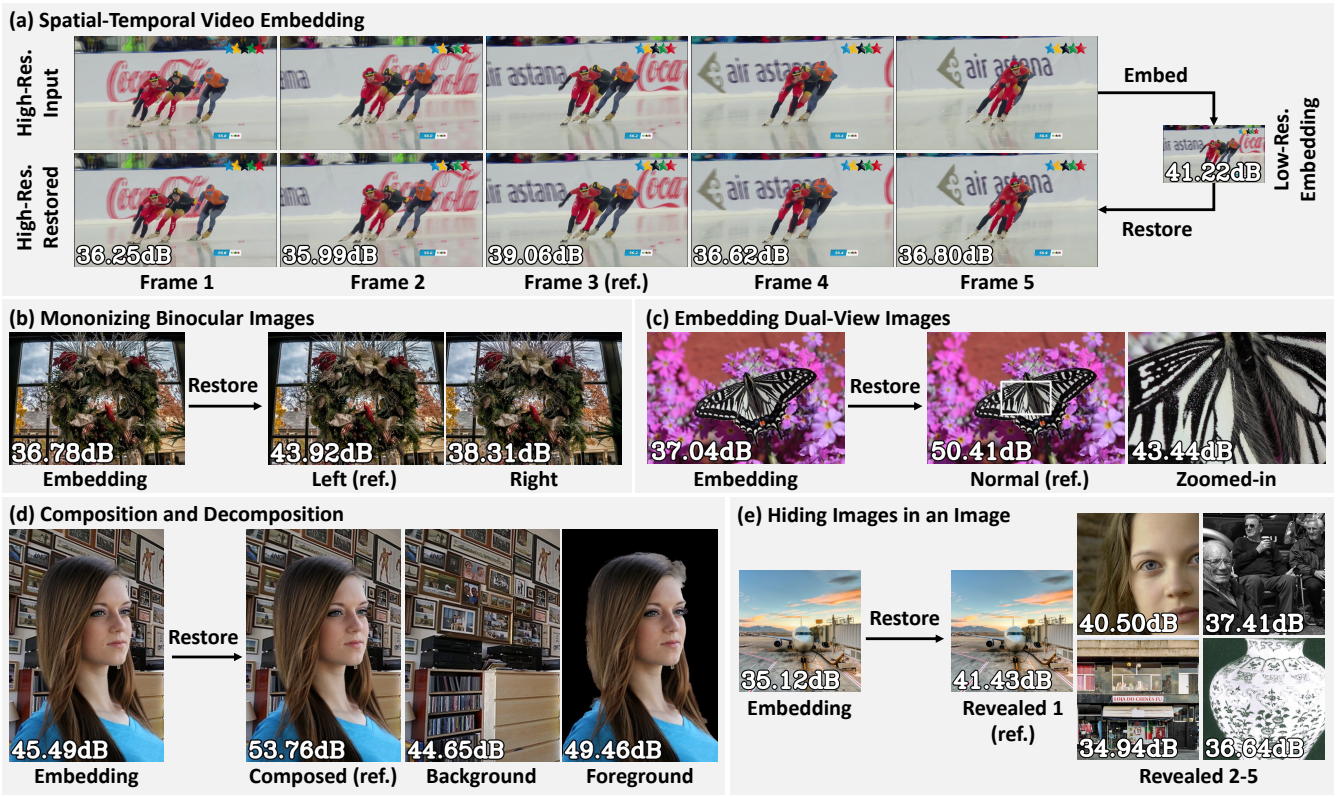

Figure 1: (a) Our IICNet can embed a high-resolution sequency into one low-resolution embedding image, which can be used to restore the original content when necessary. (b-e) Our IICNet is the first approach that can generalize among various reversible image conversion (RIC) tasks. We show the whole process of IICNet in (a) but only the restoration process in (b-e).

Abstract

Reversible image conversion (RIC) aims to build a reversible transformation between specific visual content (e.g., short videos) and an embedding image, where the original content can be restored from the embedding when necessary. This work develops Invertible Image Conversion Net (IICNet) as a generic solution to various RIC tasks due to its strong capacity and task-independent design. Unlike previous encoder-decoder based methods, IICNet maintains a highly invertible structure based on invertible neural networks (INNs) to better preserve the information during conversion. We use a relation module and a channel squeeze layer to improve the INN nonlinearity to extract cross-image relations and the network flexibility, respectively. Experimental results demonstrate that IICNet outperforms the specifically-designed methods on existing RIC tasks and can generalize well to various newly-explored tasks. With our generic IICNet, we no longer need to hand-engineer task-specific embedding networks for rapidly occurring visual content. Our source codes are available at: https://github.com/felixcheng97/IICNet.

*Joint first authors

1. Introduction

Visual media can be classified into different types, including live photos [3], binocular images or videos [14], and dual-view images or videos [1]. Usually, specific devices or platforms are required to view the visual media content. For example, binocular content may only be applicable in 3D devices, so we may need to generate corresponding monocular content to make them compatible with common devices [14]. Instead of simply dropping parts of the original content, a better choice is to build a reversible transformation, where the embedding is compatible with common devices, and the original content can be restored when necessary. Also, the single embedding image can help save the storage cost and transmission bandwidth. As a result, many researchers are motivated to study several reversible image conversion (RIC) tasks [14, 33, 40] to establish a reversible transformation between visual content and an embedding image. Some examples are shown in Figure 1.

RIC tasks are challenging since we often need to embed much richer information implicitly in one single image, which may lead to unavoidable information loss. Previous works [14, 33, 40] usually employ an encoder-decoder based framework, which learns the informative bottleneck representation but has limited ability to capture the lost information [29, 34]. For example, Zhu et al. [40] embed a video preview into a single image and restore the original content with cascaded encoders and decoders, in which they sacrifice the quality of the embedding image to embed more information, but their restored frames are still not highly accurate due to the information loss problem. Hence, one key objective in RIC tasks is to mitigate such information loss. Another concern is that although RIC tasks share the same embedding-restoration procedure for high-quality embedding and restored images, previous methods usually have task-specific designs (e.g., optical flow in [40]), making them challenging to generalize to other types of visual content. Hence, with the rapid growth of media formats plus the increasing interest in the RIC tasks, it is desirable to develop a generic framework for solving all types of RIC tasks.

Considering these aspects, we propose Invertible Image Conversion Net (IICNet) as a generic framework for RIC tasks. To alleviate the information loss problem, we utilize invertible neural networks (INNs) [12, 13] as a strictly invertible embedding module. A channel squeeze layer [35] is used and integrated into INNs for flexible reduction of dimensions, with only very minor deviations introduced to the invertible architecture. Furthermore, we introduce a relation module to strengthen the limited nonlinear representation capability of INNs [12] to better capture cross-image relations, in which independent cross-image convolution layers are used, with residual connections for better maintaining a highly reversible structure.

With the strong embedding capacity and the generic module design, IICNet does not rely on any task-specific technique, making it capable of dealing with different content types. We also allow lower-resolution embedding for higher compression rates.

Figure 1(a) gives a concrete example for illustration. Given a sequence of video frames, our IICNet can embed the spatial-temporal information of the sequence into one lower-resolution image that is visually similar to the downsampled middle reference frame. There are some promising applications. First, we may embed a short video clip or live photo in one image. Second, we can embed a high-resolution high-FPS video into a low-resolution low-FPS video. In this way, we can allow flexible adoptions for different devices and save storage. Other potential applications are shown in Figure 1(b-e), including mononizing binocular images, embedding dual-view images or multi-layer images, and even the general image hiding steganography task.

This paper presents the first generic framework IICNet for different RIC tasks, supported by extensive experiments on five tasks, including two newly-explored tasks: (1) embedding a dual-view image into a single-view one; (2) the reversible conversion between multi-layer images and a single image. Both quantitative and qualitative results show that our method outperforms the existing methods on the studied tasks. Ablation studies are conducted for the network modules and loss functions. More information and demo results are included in the supplementary materials.

2. Related Work

2.1. Reversible Image Conversion

Our work solves the embedding-and-restoration problem, which belongs to the category of reversible image conversion (RIC). Xia et al. [33] first propose to encode the original color information into a synthesized grayscale image, from which the color image can be decoded. Recently, Zhu et al. [40] try to embed a sequence of video frames into one image for single image motion expansion. Hu et al. [14] further attempt to build an invertible transformation between binocular and monocular views. Although these approaches perform well in their tasks using different technical designs, none of them can generalize to solve all the tasks above due to the task-specific designs. Also, these methods are generally based on an encoder-decoder framework with limited ability to handle the information loss problem.

The reversible property is also explored in steganography, where concealing and recovering the hidden information can be viewed as a reversible task. It aims to hide information within different information carriers like images. Recently, several learning-based methods [9, 25, 30, 31, 38, 39] leverage the pair of encoder and decoder to hide different kinds of information in images. Still, some works have a limited hiding capacity with some artifacts. In this work, we mainly focus on RIC tasks related to the image carrier only.

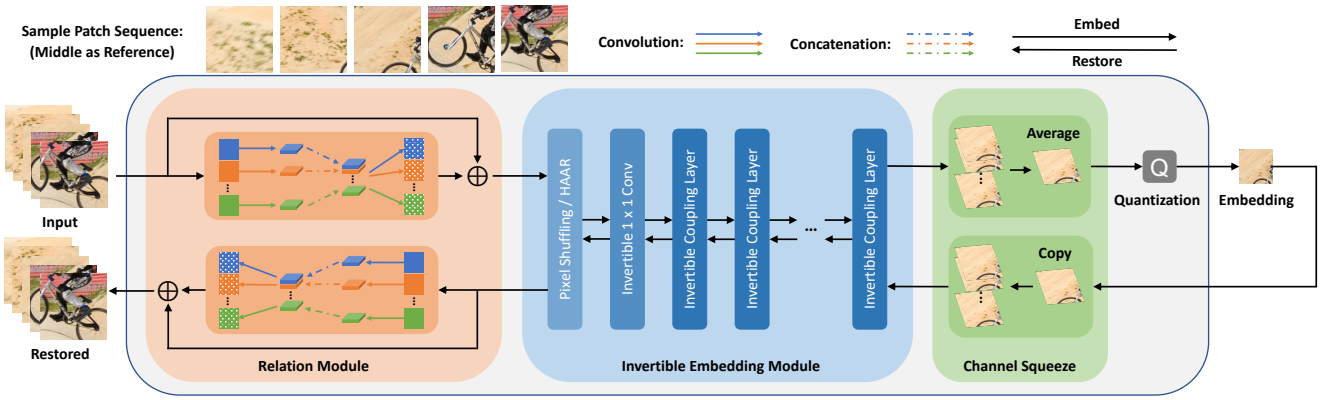

Figure 2: Overview of the proposed network. IICNet sequentially contains a relation module, an invertible embedding module (an optional downscaling module plus several coupling layers), a channel squeeze layer, and a quantization layer.

2.2. Invertible Neural Networks

Invertible neural networks (INNs) [12, 13, 16] guarantee the invertibility property with a careful mathematical design of network architecture and several invertible operations. In general, the forward process of an INN architecture can learn a bijective mapping between a source domain x to a target domain y, with the forward process $f_\theta(x) = y$ and the inverse process $f_\theta^{-1}(y) = x$. A tractable Jacobian is another great characteristic of INNs to compute the posterior probabilities explicitly for the bijective mapping.

Normalizing Flow based methods [17, 22] map a complex distribution x with INNs to a latent distribution z (e.g., Gaussian), usually trained by minimizing the unsupervised negative log-likelihood loss. Different from Normalizing Flow based methods, IRN [34] maps a high-resolution image to a low-resolution image by utilizing additional latent output variables to capture the lost high-frequency information [24] with a cross-entropy loss in the image rescaling task. However, the information loss or residual is usually more complex in other general RIC tasks, making the generalization issue a big challenge for IRN. Recent works also investigate the application of INNs on different tasks, such as conditional image super-resolution [19], image generation [6, 32], point cloud generation [21], segmentation tasks [32], and image signal processing pipeline [36].

3. Method

The proposed IICNet for general reversible image conversion (RIC) tasks aims to encode a series of input images into one reversible image (embedding image), which can have either the same or lower resolution. The embedding image can be decoded back to the original inputs with the network backward passing. The key is to use invertible neural networks (INNs) to model such a bijective mapping. An overview of our generic framework is shown in Figure 2.

3.1. Model Formulation

Formally, the input of IICNet is a series of K input images $\{\mathbf{i}_k\}_{k=1}^K$ with $\mathbf{i}_k \in \mathbb{R}^{C\times H\times W}$, where C, H, and W are the image channel number, height, and width, respectively. IICNet can forwardly encode the input images into an embedding image $\mathbf{e}$, which is visually indistinguishable from the reference image $\mathbf{e}_{ref} \in \mathbb{R}^{C_e\times H_e\times W_e}$. Note that the embedding C_e, H_e, and W_e may be different from C, H, and W. IICNet can then backwardly decode the quantized embedding image $\hat{\mathbf{e}}$ and restore the input images $\{\hat{\mathbf{i}}_k\}_{k=1}^K$. Note that in actual implementations, K input images are stacked along the channel dimension, with input channel size of $N = CK$, denoted as $\mathbf{x}_{1:N} \in \mathbb{R}^{N\times H\times W}$.

Relation module. INNs have strong architecture constraints, limiting the nonlinear representation capacity [12]. Thus, we propose a relation module to add some nonlinear transformation to help capture cross-image relations. To minimize information loss, we add residual connections to greatly preserve the network reversibility.

Details of the relation module are shown as the orange part in Figure 2. K parallel convolutional headers independently transform K images into their feature space. The concatenation of the K image features then goes through K independent convolutional tailers plus residual connections to obtain the corresponding images with relational information extracted. The convolutional blocks used here are based on the Dense Block [15]. We can express the forward process f_{rel}^k for the k^{th} image $\mathbf{x}_{(kC-C+1):kC}$ as follows:

$$\mathbf{r}_{(kC-C+1):kC} = f_{rel}^k(\mathbf{x}_{1:N}) + \mathbf{x}_{(kC-C+1):kC}. \quad (1)$$

We then obtain $\mathbf{r}_{1:N} \in \mathbb{R}^{N\times H\times W}$. For the inverse process, we apply a symmetric relation module.

Invertible downscaling module. If we optionally activate the invertible downscaling module, IICNet can embed the input images into a lower-resolution embedding image.

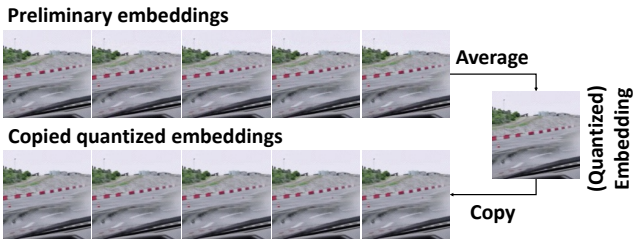

Figure 3: Illustration of the channel squeeze layer.

This module is composed of either a pixel shuffling layer (squeezing operation) [13] or a Haar wavelet transformation layer [27], followed by an invertible 1×1 convolution [16]. This module offers an invertible operation to halve the resolution of the input images, transforming the size of input tensor from (N, H, W) to $(4N, \frac{1}{2}H, \frac{1}{2}W) = (M, H_e, W_e)$. We describe the forward process of this module f_{down} as:

$$\mathbf{u}_{1:M} = f_{down}(\mathbf{r}_{1:N}). \tag{2}$$

If downscaling is disabled, f_{down} is simply an identical function, yielding $\mathbf{u}_{1:M} = \mathbf{r}_{1:N}$.

Coupling layers. Following the design proposed in [5, 12, 13], we structure a deep INN architecture with several basic inveritble building blocks using two complementary affine coupling layers each. Considering the l^{th} block, the corresponding input tensor $\mathbf{u}_{1:M}$ is split into top parts $\mathbf{u}_t^l = \mathbf{u}_{1:\tilde{m}}^l$ and bottom parts $\mathbf{u}_b^l = \mathbf{u}_{(\tilde{m}+1):M}^l$ at position $\tilde{m}$. The two corresponding affine transformations are formulated as follows, with element-wise multiplication $\odot$, exponential function $\exp(\cdot)$, and centered sigmoid function $\sigma_c(\cdot) = 2\sigma(\cdot) - 1$:

$$\mathbf{u}_t^{l+1} = \mathbf{u}_t^l + h_2(\mathbf{u}_b^l), \tag{3}$$

$$\mathbf{u}_b^{l+1} = \mathbf{u}_b^l \odot \exp(\sigma_c(g(\mathbf{u}_t^{l+1}))) + h_1(\mathbf{u}_t^{l+1}). \tag{4}$$

Then $\mathbf{u}_t^{l+1}$ and $\mathbf{u}_b^{l+1}$ are concatenated to get $\mathbf{u}_{1:M}^{l+1}$. We can show that the two transformations are invertible:

$$\mathbf{u}_b^l = (\mathbf{u}_b^{l+1} - h_1(\mathbf{u}_t^{l+1})) \odot \exp(-\sigma_c(g(\mathbf{u}_t^{l+1}))), \tag{5}$$

$$\mathbf{u}_t^l = \mathbf{u}_t^{l+1} - h_2(\mathbf{u}_b^l). \tag{6}$$

Letting f_{inn} be the forward pass of our INN architecture, the output tensor $\mathbf{v}_{1:M}$ can be formulated as follows:

$$\mathbf{v}_{1:M} = f_{inn}(\mathbf{u}_{1:M}). \tag{7}$$

Channel squeeze layer. Similar to [35], we use a channel squeeze layer but without attention to reduce the channel dimension to obtain the embedding image $\mathbf{e}$. The channel squeeze layer forwardly treats its input tensor $\mathbf{v}_{1:M}$ as a stack of preliminary embedding images $\{\mathbf{e}_k\}_{k=1}^{K_e}$, where $K_e = M/C_e$. The embedding image $\mathbf{e}$ is calculated by averaging the preliminary embedding images:

$$\mathbf{e} = f_{cs}(\mathbf{v}_{1:M}) = average(\{\mathbf{e}_k\}_{k=1}^{K_e}). \tag{8}$$

While for the backward pass, the channel squeeze layer copies the quantized embedding image $\hat{\mathbf{e}}$ multiple times as $\{\hat{\mathbf{e}}_\mathbf{k}\}_{k=1}^{K_e}$ and concatenates them along the channel dimension to match the channel size.

Note that our network is jointly trained as a whole with inherent inverse functions of INNs, and the inverse pass takes the copied (same) quantized embedding images $\{\hat{\mathbf{e}}_\mathbf{k}\}_{k=1}^{K_e}$ as input. This implicitly guides the embedding image $\mathbf{e}$ and all the preliminary embedding images $\{\mathbf{e}_k\}_{k=1}^{K_e}$ to look similar to each other. Hence, only minor noise is introduced to the invertibility, and there is no need to pose any explicit constraints on $\{\mathbf{e}_k\}_{k=1}^{K_e}$ during the forward pass. Figure 3 shows some visual patches of the preliminary embedding images during training, where all the preliminary embedding images are similar to each other. Also, we find that such implicit guidance on the preliminary embedding images helps stabilize the overall training process.

During experiments, we try to pose explicit L_2 constraints between $\{\mathbf{e}_\mathbf{k}\}_{k=1}^{K_e}$ and $\mathbf{e}$ or to model the information loss for the channel squeeze layer by simple CNNs. But such designs cause worse performance or unstable training.

Quantization layer. A quantization loss is unavoidable when one saves the embedding image in the common PNG format with only 8 bits per pixel per channel. There are many proposed methods like [7, 10, 26] to address this problem. In this paper, we choose to employ the method in [8] to add uniform noise during training and do integer rounding during testing to obtain the quantized embedding image $\hat{\mathbf{e}}$. The quantized embedding image further needs to be clamped between 0 and 255.

Inverse process. To restore the original input images, we can load the quantized embedding image $\hat{\mathbf{e}}$ and let it sequentially go through the inverse pass of IICNet:

$$\hat{\mathbf{x}}_{1:N} = (f'_{rel} \circ f_{down}^{-1} \circ f_{inn}^{-1} \circ f'_{cs})(\hat{\mathbf{e}}), \tag{9}$$

where $f'_{rel}, f_{down}^{-1}, f_{inn}^{-1}, f'_{cs}$ are the inverse pass functions of the corresponding modules. Then we can obtain the restored images $\{\hat{\mathbf{i}}_k\}_{k=1}^{K}$.

3.2. Loss Functions

As discussed in the channel squeeze layer, we only need to employ loss functions at the two ends: the embedding image and the restored images.

Embedding image. We employ L_2 loss to guide the embedding image $\mathbf{e}$ to be visually like the reference image $\mathbf{e}_{ref}$. In the case of downscaling, we use the Bilinear method to downsample the reference image:

$$\mathcal{L}_{emb} = ||\mathbf{e}_{ref} - \mathbf{e}||_2^2. \tag{10}$$

In our experiments, we find that with L_2 loss only, the embedding image usually contains many high-frequency patterns. Hence, we further apply one-sided Fourier transform (FT) [11] on both the embedding image and the reference image to obtain their frequency domain and add a

Step	Embedding				Restored			
	Zhu et al. [40]		Ours		Zhu et al. [40]		Ours	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
1	25.277	0.5608	**37.908**	**0.9412**	34.356	0.9363	**36.698**	**0.9519**
3	24.561	0.5214	**37.068**	**0.9252**	33.099	0.9227	**36.302**	**0.9490**
5	24.246	0.5056	**36.739**	**0.9190**	32.608	0.9170	**36.074**	**0.9475**

Table 1: Comparison on temporal video embedding test set with embeding range of 9 and time step of 1.

frequency loss $\mathcal{L}_{freq}$ in terms of L_2 distance:

$$\mathcal{L}_{freq} = ||FT(\mathbf{e}_{ref}) - FT(\mathbf{e})||_2^2. \tag{11}$$

Restored images. The restored images $\{\hat{\mathbf{i}}_k\}_{k=1}^K$ should match the original ones $\{\mathbf{i}_k\}_{k=1}^K$, so we have another basic restored loss $\mathcal{L}_{res}$ to minimize the average L_1 distance among each pair of the restored and original image:

$$\mathcal{L}_{res} = \frac{1}{K}\sum_{k=1}^{K} ||\mathbf{i}_k - \hat{\mathbf{i}}_k||_1. \tag{12}$$

Total loss. To sum up, our proposed IICNet is optimized by minimizing the compact loss $\mathcal{L}_{total}$, with corresponding weight factors λ_1, λ_2, λ_3:

$$\mathcal{L}_{total} = \lambda_1\mathcal{L}_{emb} + \lambda_2\mathcal{L}_{freq} + \lambda_3\mathcal{L}_{res}. \tag{13}$$

4. Experiments

We first report experiments conducted on the studied RIC tasks in Section 4.1 and 4.2, followed by the results of two newly-explored tasks in Section 4.3 and 4.4. In Section 4.5, we try the steganography task to hide several images in one image. The main paper reports multiple-and-single RIC tasks that build a conversion between multiple images and a single image. Our supplements present more results of single-and-single RIC tasks like invertible image rescaling and invertible grayscale. Please also check our supplements for detailed experimental settings.

4.1. Spatial-Temporal Video Embedding

The method proposed in [40] aims to embed a sequence of video frames into one embedding image with the same resolution, which can be converted back to the original video sequence. Our proposed IICNet not only performs better but also extends to embed the video frames spatiotemporally into a lower-resolution embedding image.

Dataset and processing. We use the high-quality DAVIS 2017 video dataset [20] in this task. To make our model more robust on different motion levels of video inputs, for each video sample in the train set, we subsample all the possible video subsamples with a time step of 5 between consecutive frames, where we select the middle frame as the reference image.

Zhu et al. [40] Ours

Figure 4: Visual result comparisons on embedding images.

Result comparison. Table 1 only reports the comparison results on the test set with embedding range $N = 9$, since the baseline method [40] only provides the pre-trained $N = 9$ model. We study the performance at different time step levels of 1, 3, 5 to test the capacity of the models in handling small and large motions. Statistics show that our method significantly outperforms the baseline method at all time step levels by large margins. Without dependence on optical flow, our method has less performance drop as the time step grows. We also offer grayscale PSNR and SSIM comparisons in our supplements for reference.

Figure 4 and Figure 5 show the visualization results of the embedding images and the restored frames, respectively. Evident artifacts are found in baseline results, especially for the embedding image. In contrast, our embedding and restored images contain very few artifacts, demonstrating the effectiveness of the employed INN architecture in RIC tasks.

Embedding ranges and resolutions. To investigate the embedding capacity of our method, we conduct experiments using different embedding ranges ($5, 7, 9$ input images) in Table 2. Similarly, we subsample the training videos with a time step of 5 and test at a time step level of 1. Intuitively, more input images indicate more challenges because there is usually more motion information to embed into the embedding image. Table 2 further shows the experimental results of our method to embed the input video sequence spatially and temporally into a lower-resolution embedding image. To the best of our knowledge, no previous work tries

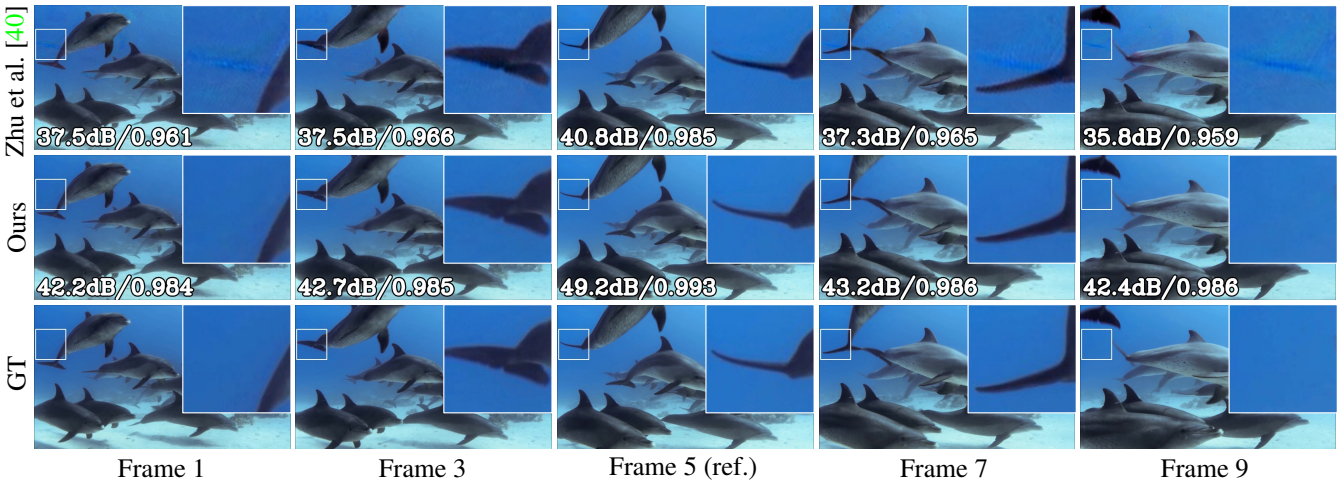

Figure 5: Visual result comparisons on restored frames.

Range (Res.)	Embedding PSNR	Embedding SSIM	Restored PSNR	Restored SSIM
5	38.900	0.9522	41.729	0.9807
7	38.157	0.9437	38.785	0.9660
9	37.908	0.9412	36.698	0.9519
3 (×2)	37.585	0.9584	36.914	0.9540
5 (×2)	36.692	0.9477	33.977	0.9205

Table 2: Results on spatial-temporal video embedding test set with different embedding ranges and resolutions.

to do the spatial-temporal embedding task. We report the results of embedding $N = 3, 5$ into a 2 times lower-resolution image, conducted with a time step of 5 and 1 for training and testing, respectively. We can see that even we compress the input frames up to 20 times smaller; still, the model can have a good preview image and restored frames.

4.2. Mononizing Binocular Images

We also experiment on another studied mononizing binocular images task [14], which aims to convert binocular images or videos into monocular ones with the stereo information implicitly encoded. In this way, monocular devices can cope with stereoscopic data, and the original stereo content can be restored when necessary. We demonstrate that our framework outperforms state-of-the-art methods.

Same as [14], we train on the Flickr1024 dataset [28] with the official train and test splits. Quantitative results are shown in Table 3. We achieve the best performance, especially for the restored images, with an improvement of 6.6dB for the left views and 1.1dB for the right views. Although Mono3D already achieves good performance, we can still see some structural artifacts like the street lamp and the electric tower in the zoomed-in restored patches,

	Mono-view PSNR	Mono-view SSIM	L. Bino-view PSNR	L. Bino-view SSIM	R. Bino-view PSNR	R. Bino-view SSIM
Baluja [9]	26.1	0.81	-	-	27.9	0.88
Xia et al. [33]	28.0	0.89	28.7	0.92	30.7	0.92
Hu et al. [14]	**37.8**	**0.97**	38.3	0.99	37.3	0.98
Ours	37.5	0.95	**44.9**	**0.99**	**38.4**	**0.98**

Table 3: Results on mononizing binocular images test set.

as shown in Figure 6. In contrast, our method can restore nearly artifact-free binocular views. Although we only train our network on images, results show strong temporal consistency when we apply our model to videos in a per-frame manner. Some demos are in the supplementary video.

4.3. Embedding Dual-View Images

Dual-view camera mode is an advanced technology in the field of smartphone cameras, which is first available on HUAWEI P30 Pro [1]. Users can record split-screen images or videos with the primary camera capturing normal-view images or videos on the left and the zoom lens capturing zoomed-view (×4) images or videos on the right.

Similarly, not all devices support dual-view images. Our method can serve as a backward-compatible solution to embed the dual-view images into one normal-view image. We train and test our method using pairs of zoomed-view (×2,×4,×8) and normal-view images generated from the DIV2K dataset [4], with the normal view images as reference. Some setting details are in the supplements.

Quantitative results in Table 4 show that our method achieves great performance to embed dual-view images in terms of both PSNR and SSIM. We also show some visual results in Figure 7, where we can see that both the embedding and restored images are nearly perfect.

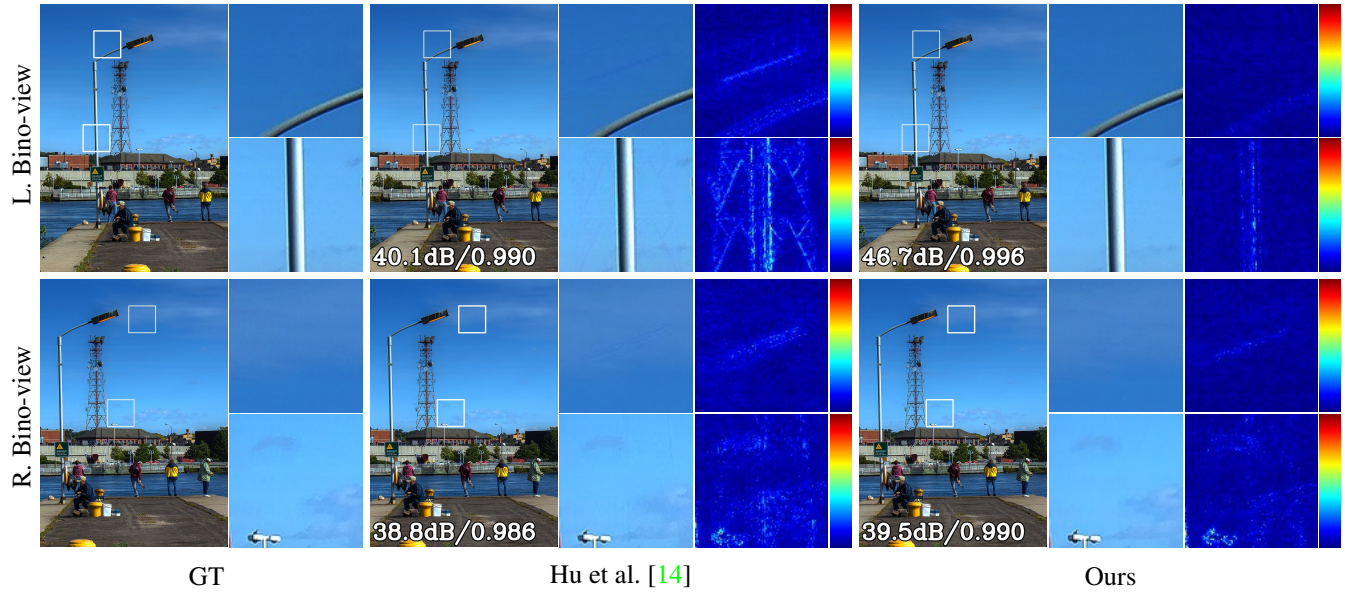

Figure 6: Visual comparison results of the restored Binocular Images. We show the zoomed-in patches with the corresponding error map aside. Note that we amplified the error maps by 10 times for better visualization.

Modes	Embedding	Normal	Zoomed
× 2	38.248	50.171	43.461
× 4	38.438	49.116	43.662
× 8	38.356	48.854	43.578

Table 4: PSNR on embedding dual-view images.

Figure 7: A sample result of embedding dual-view images.

4.4. Composition and Decomposition

Photoshop [2] is a popular image editing software, where users can use multiple layers to perform tasks such as compositing multiple images into one. Usually, the composition process is not reversible, so we cannot recover the sheltered part of the background in the composed image. However, with our method, we can allow the "composed image" to embed all the layer images. In this way, although we only store and transmit one "composed image" as before, users can also get the original layers for further usage.

Since there is no publicly available dataset for composition and decomposition, we instead train and test our method on two matting datasets: the Adobe Deep Matting dataset [37] and the Real Matting dataset [23]. Note that the Real Matting dataset does not have ground truth for the foreground. Detailed settings are available in supplements.

Table 5 shows the quantitative performance of our method on the two datasets. We also include some visual results in Figure 8. We can see that our method performs well and is verified as applicable to the task of composing and decomposing images.

	Embed.	Comp.	Fg.	Bg.
Adobe	45.305	52.709	44.586	44.921
Real	47.350	60.234	-	43.718

Table 5: PSNR on composition and decomposition.

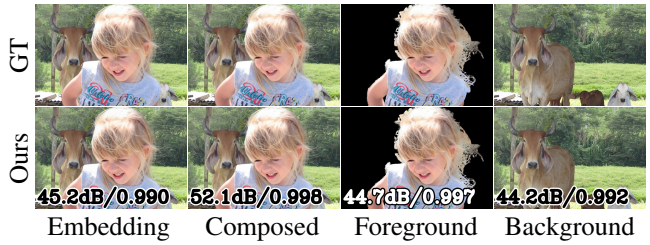

Figure 8: A sample result of composition and decomposition.

Methods	Video Embedding			Mononizing Binocular Images			Hiding Images in an Image		
	Embed.	Restore	#Param.	Embed.	Restore	#Param.	Embed.	Restore	#Param.
AE [33]	37.925	37.242	7.43M	35.387	38.239	4.55M	34.248	31.721	7.43M
INNs [34]	34.029	38.452	6.57M	34.465	38.171	4.49M	29.953	33.843	6.57M
Ours w/o rel.	38.752	41.159	6.57M	36.684	39.667	4.49M	35.533	36.698	6.57M
Ours w/o freq.	32.914	**42.353**	6.81M	31.469	41.161	4.40M	28.780	37.623	6.81M
Ours	**38.900**	41.729	6.81M	**37.540**	**41.649**	4.40M	**35.641**	**37.935**	6.81M

Table 6: Ablation studies on three representative tasks.

#Embed.	Embedding		Restored	
	PSNR	SSIM	PSNR	SSIM
2	38.586	0.9403	48.599	0.9945
3	37.038	0.9166	42.884	0.9852
4	36.184	0.9041	39.883	0.9745
5	35.641	0.8913	37.935	0.9638

Table 7: PSNR on hiding images in an image.

Figure 9: A sample result of Hiding Images in Image

4.5. Hiding Images in an Image

To show the generality of our proposed model, we try the hardest task to hide several unrelated images with our model, which can be viewed as a kind of stenography. We obtain general images from the Flicker 2W dataset [18]. We conduct experiments to embed 2, 3, 4, 5 images into one image, and the numerical results are listed in Table 7. From the results, we can see that our method achieves relatively good performance even when embedding 5 images into one, demonstrating the strong generality of our method. From the visual results shown in Figure 9, despite the variety of colors and structures of the images, we can restore them with no viewable artifacts.

5. Ablation Studies

To ablate our network components and the applied frequency loss, we report some ablation results on three representative tasks in Table 6. For AE, we use the network architecture proposed by Xia et al. [33] to represent general encoder-decoder based methods; for INNs, we adopt the network design and training strategy introduced by Xiao et al. [34] to represent common INN based methods with auxiliary maps. We also present the results of our methods without relation module or frequency loss. For fair comparisons, all the models (unless otherwise specified) are trained with the applied frequency loss as discussed in Section 3, and we adjust the number of invertible blocks or CNN layers of different methods to have a similar number of parameters.

The experiments show that our method outperforms general encoder-decoder style networks and common INNs with auxiliary maps. Intuitively, we know that there exists a trade-off relation between the embedding quality and the restoration quality. From the reported statistics, we can conclude that the frequency loss greatly contributes to the artifacts-free embedding for a significant quality boost with comparable restoration quality. Also, the proposed relation module works well to integrate with INNs to extract cross-image relationships and boost the performance.

6. Conclusion and Discussion

We present a generic framework IICNet for various reversible image conversion (RIC) tasks. IICNet maintains a task-independent and highly invertible architecture based on invertible neural networks (INNs), which can help greatly minimize the information loss during the conversion process. Due to strict invertibility, INNs have limitations in terms of nonlinear representation capacity and dimensional flexibility. The introduced relation module and the applied channel squeeze layer can greatly alleviate such limitations for better cross-image relation extraction and preserve the information-reserving ability of INNs.

IICNet yields state-of-the-art performance on some studied RIC tasks, such as spatial-temporal video embedding and mononizing binocular images. We also introduce and apply our IICNet on some unexplored tasks, which are embedding dual-view images and composition and decomposition. The success on the stenography task further shows the generalization of our IICNet. We hope the generalization and high performance of the proposed framework could help in more practical applications.

References

[1] Huawei p30 and huawei p30 pro's dual-view camera mode is now available. https://consumer.huawei.com/sg/press/news/2019/news-1906132/. 2, 6

[2] Photo, image & design editing software. https://www.adobe.com/products/photoshop.html. 7

[3] Take and edit live photos. https://support.apple.com/en-us/HT207310. 2

[4] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 6

[5] Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W. Pellegrini, Ralf S. Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018. 4

[6] Lynton Ardizzone, Carsten Lüth, J. Kruse, C. Rother, and U. Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019. 3

[7] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *Proceedings of Picture Coding Symposium*, pages 1–5, 2016. 4

[8] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 4

[9] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *Neural Information Processing Systems*, 2017. 2, 6

[10] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4

[11] E. Oran Brigham and R. E. Morrow. The fast fourier transform. *IEEE Spectrum*, 4(12):63–70, 1967. 4

[12] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2015. 2, 3, 4

[13] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2017. 2, 3, 4

[14] Wenbo Hu, Menghan Xia, Chi-Wing Fu, and Tien-Tsin Wong. Mononizing binocular videos. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 39(6):228:1–228:16, 2020. 2, 6, 7

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2261–2269, 2017. 3

[16] Durk P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proceedings of Advances in Neural Information Processing Systems*, volume 31, 2018. 3, 4

[17] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 3

[18] Jiaheng Liu, Guo Lu, Zhihao Hu, and Dong Xu. A unified end-to-end framework for efficient deep image compression. *arXiv preprint arXiv:2002.03370*, 2020. 8

[19] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *Proceedings of the European Conference on Computer Vision*, 2020. 3

[20] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2018. 5

[21] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 3

[22] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the International Conference on Machine Learning*, pages 1530–1538, 2015. 3

[23] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2288–2297, 2020. 7

[24] Claude E. Shannon. Communication in the presence of noise. In *Proceedings of the Institute of Radio Engineers*, volume 37, pages 10–21, 1949. 3

[25] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. *arXiv preprint arXiv:1904.05343*, 2020. 2

[26] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017. 4

[27] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I:511–I:518, 2001. 4

[28] Yingqian Wang, Longguang Wang, Jungang Yang, Wei An, and Yulan Guo. Flickr1024: A large-scale dataset for stereo image super-resolution. In *International Conference on Computer Vision Workshops*, pages 3852–3857, 2019. 6

[29] Yaolong Wang, Mingqing Xiao, Chang Liu, Shuxin Zheng, and Tie-Yan Liu. Modeling lost information in lossy image compression. *arXiv preprint arXiv:2006.11999*, 2020. 2

[30] Zihan Wang, Neng Gao, Xin Wang, Ji Xiang, Daren Zha, and Linghui Li. Hidinggan: High capacity information hiding with generative adversarial network. *Computer Graphics Forum*, 38, 2019. 2

[31] Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1515–1524, 2019. 2

[32] Christina Winkler, Daniel Worrall, E. Hoogeboom, and M. Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019. 3

[33] Menghan Xia, Xueting Liu, and Tien-Tsin Wong. Invertible grayscale. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6):246:1–246:10, 2018. 2, 6, 8

[34] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. Invertible image rescaling. In *Proceedings of the European Conference on Computer Vision*, pages 126–144, 2020. 2, 3, 8

[35] Yueqi Xie, Ka Leong Cheng, and Qifeng Chen. Enhanced invertible encoding for learned image compression. *arXiv preprint arXiv:1308.3432*, 2021. 2, 4

[36] Yazhou Xing, Zian Qian, and Qifeng Chen. Invertible image signal processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[37] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 311–320, 2017. 7

[38] Hyukryul Yang, Hao Ouyang, Vladlen Koltun, and Qifeng Chen. Hiding video in audio via reversible generative models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1100–1109, 2019. 2

[39] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European Conference on Computer Vision*, 2018. 2

[40] Qianshu Zhu, Chu Han, Guoqiang Han, Tien-Tsin Wong, and Shengfeng He. Video snapshot: Single image motion expansion via invertible motion embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2, 5, 6