Robust Message Embedding via Attention Flow-Based Steganography

Supplementary Material

A. Details on Model Implementation

A.1. Invertible QR Code Transition

We directly adopt the invertible neural network (INN) architecture of ISN [8] to implement the QR Code transition procedure. Instead of the 16 invertible blocks used in the original paper, we only use 2 of them to lower the model complexity, since we empirically find that this transition does not need that large volume of parameters.

As mentioned in Sec. 3.3, we employ a constraint on the transformed QR Code to ensure that it can be identified to restore the secret message. We adopt the same strategy as ArtCoder [12], which is to simulate the most used *Google ZXing* [10] rules that only read the center pixel of each module in a QR Code. According to Xu et al. [15], the pixels closer to the module center should have a higher probability to be sampled. As a result, this sampling procedure can be modeled by performing a Gaussian convolution operation upon each code module. Specifically, given a QR Code with $n \times n$ modules of size $m \times m$, a Gaussian convolution kernel sized $m \times m$ is used to convolute each module with a stride of m and derive an $n \times n$ sample result. The feature map is then binarized with a threshold, which is empirically set as 0.02 (given the pixel values range in $[0, 1]$) in this paper, since we find this threshold value can guarantee the identifiability of transformed QR Code. This means those pixels with values greater than 0.02 are regarded as white modules and the rest are black modules. During training, we obtain an error map ξ which indicates the wrongly transformed code modules and backward the gradient for model optimization. This process is demonstrated in Fig. S1 and the optimization target can be formulated as the following loss function:

$$\begin{aligned} \mathcal{L}_t &= \left\| qc(I_q^*) \cdot \xi - qc(I_q) \cdot \xi \right\|_1, \\ \xi &= \left\| bin_k(qc(I_q^*)) - qc(I_q) \right\|_1, \end{aligned} \tag{S1}$$

in which I_q and I_q^* represent the original QR Code and transformed result, respectively, $qc(\cdot)$ indicates the Gaussian kernel convolution and $bin_k(\cdot)$ is the binarize operation with the threshold as k. Since the aforementioned calculation is differentiable, it can be optimized jointly with other network modules during training.

In our implementation, we resize the QR Code to a size of $5n \times 5n$ and use a 5×5 kernel to perform the convolution operation. The value of n depends on the version of QR Code and it is 37 for version 5 that we adopt in this paper. For each subsequent version, the value of n is 4 greater than the previous version, e.g., it is 41 for version 6 and so forth.

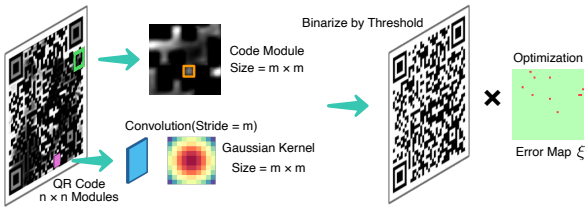

Figure S1. Demonstration of the QR Code scanning simulation.

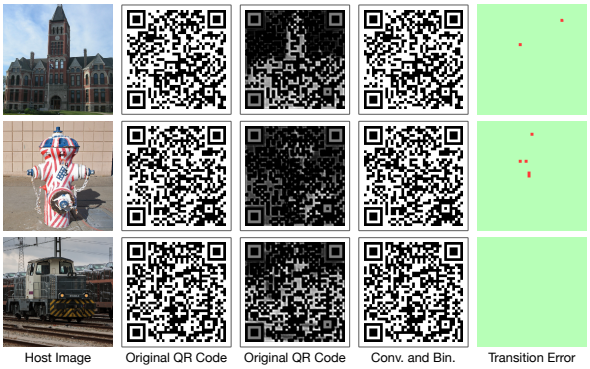

Figure S2. Some transition results. Here Conv. and Bin. indicate convolution and binarization, respectively.

Fig. S2 shows some transition results, we also provide the QR Code and its error map after the aforementioned convolution and binarization operations. Although the transition can sometimes lead to some wrongly transformed code modules, it does not affect the identifiability.

A.2. AttnFlow Model

We use some tokenizers to convert images to tokenized representations and some detokenizers to transform them back in the AttnFlow model. In our implementation, all the tokenizers used are based on the vision transformer (ViT) [2] architecture. We make some slight changes on the ViT-Base model that contains 12 transformer blocks with the token dimensionality and the multi-layer perceptron (MLP) size being 768 and 3072, respectively. Specifically, we reduce the model complexity in our implementation by lowering the block number to 2 and the MLP size to 2048. The patch size used is 16×16. For the detokenizer, we simply use an MLP for dimension projection followed by a reshaping operation and two convolutional layers with GELU [3] as activation function to convert the tokens back to image.

For the self-attnetion and cross-attnetion blocks used in AACB, they have the same token dimensionality and MLP size as the tokenizers. We choose to use 4 AACBs in our full model since we find this block number makes a good balance between model performance and training cost.

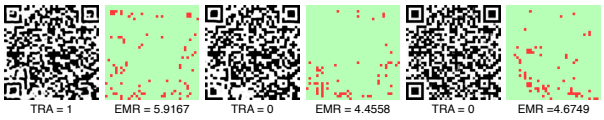

Figure S3. An example of decoded QR Code with higher EMR but lower TRA.

B. Experiment Details

B.1. Metrics Calculation

LPIPS We use LPIPS [16] as part of the optimization target during model training and one of the metrics for stego image quality evaluation. We calculated it with a VGG [11] model pre-trained on ImageNet [1].

TRA We calculate the text recovery accuracy (TRA) with the following scheme:

$$TRA(\hat{I}_{qr}) = \begin{cases} 1.0 & \text{if } \hat{I}_{qr} \text{ is identifiable} \\ 0.0 & \text{otherwise} \end{cases}, \tag{S2}$$

where $\hat{I}_{qr}$ indicates the decoded QR Code. The final TRA is average value upon the whole testing dataset.

EMR We calculate the error module rate (EMR) by measuring the wrongly decoded QR Code modules. Given a decoded QR Code, we binarize and compare it with the ground truth to derive the error rate.

It is worth noticing that, although low EMR can generally guarantee a high TRA, these two metrics are not necessarily positively correlated. An example is shown in Fig. S3, the second and third decoded QR Codes have lower EMR than the first one but they are not identifiable. This can be caused by two reasons, the first is that the finder and alignment patterns are damaged, making the code cannot be detected, corresponding to the second case in Fig. S3 (the finder pattern in the lower left is damaged). The second is that, a high error rate is caused in a small area, making the error correction (ECC) scheme of QR Code fail to restore the error, corresponding to the third case in Fig. S3.

B.2. Training Details

Our model is implemented with PyTorch [9] and is trained on 4 NVIDIA GeForce 3090 GPUs. We set the batch size as 8 for each GPU and the mode is trained for 50K iterations. The initial learning rate is 0.0001, which decays by 10% after each epoch until it reaches 0.00001. The model is optimized with the AdamW optimizer [7] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The hyper-parameters in the loss function (Equation 10) are set as $\alpha = 5.0$, $\beta = 0.2$, $\gamma = 3.5$, $\delta = 16$ and $\epsilon = 3.0$. The trainnable parameters α_i introduced in Equation 3 are initialized as 0.01 in the beginning. The training host images are randomly cropped from the original images as 224×224 patches. The overall training process takes for about 13 hours.

Table S1. Comparison of distortion parameters used by StegaStamp and RMSteg. Here *Bri.* is brightness, *Sat.* is saturation, *Noi.* is Gaussian noise level and *Tra.* is transition.

Method	Bri.	Hue	Sat.	Contrast	jpeg	Noi.	Blur	Tra.
StegaStamp	0.3	0.1	1.0	$[0.5, 1.5]$	25	0.02	7	0.10
Ours	0.3	0.1	1.0	$[0.5, 1.5]$	**60**	**0.07**	7	**0.02**

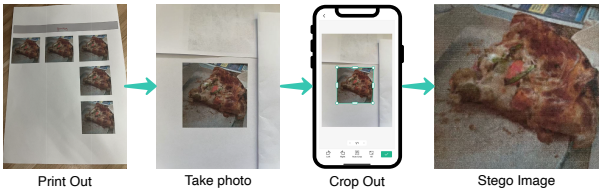

Figure S4. How we derive the stego image in the printing test.

B.3. Distortion Simulation

We adopt the same types of distortions as StegaStamp [13] with some changes in the hyper-parameters. We provide a comparison of the settings of StegaStamp and our in Tab. S1. We largely increase the Gaussian noise level to fit for our task. We lower the distortion level of JPEG compression since we find that there is no need to use a very low JPEG compression quality for training to achieve sufficient robustness when a high-level Gaussian noise is employed. For the parameter of transition, since we manually crop the stego image out from the photo, rather than using object detection as StegaStamp, we do not need such a high parameter setting to promise enough robustness. As for how the distortion simulation is implemented and how do these parameters work, we strongly suggest referring to the description in the original paper [13].

B.4. Printing and Photography

In this paper, we choose to use the case of printing and photography to measure the method robustness under extreme real-world distortions. During experiment, we first print the stego images out and take photos for them. For one stego image, we print it for multiple times on the same paper to prevent the fluctuation of printer's printing quality. Then, we manually crop the stego images out from the photos using CamScanner [4] and use them for decoding test. A demonstration of the above workflow is shown in Fig. S4.

The printer we used for experiment is an HP OfficeJet Pro 8710 inkjet printer. We choose the printing quality of 'Normal' (among 'Draft', 'Normal' and 'High'). The printed image size on paper is $5.4cm \times 5.4cm$. We take photos with an iPhone 13 Pro indoors under regular illumination. We take 5 photos for each image and choose the best value for final metrics calculation. For the experiments under different shooting angles, we maintain a vertical distance of 11 *cm* between the lens and the paper, which is our default shooting distance.

Table S2. The α_i values when using different numbers of AACBs.

Block Number	α_1	α_2	α_3	α_4
1	0.1354	–	–	–
2	0.0062	0.1425	–	–
3	0.0070	0.0915	0.0087	–
4	0.0022	0.1350	0.0745	0.0070

C. Further Experiment and Discussion

C.1. Trainnable Coefficients in AACB

We set the coefficient α_i of the cross-attention item in the AACB transformation function (Equation 3) as a trainnable parameter to let the network learn by itself. We initialize this parameter as 0.01 at the beginning and optimize it during training. Here we provide the final converged values of α_i in the models with different numbers of AACBs. The result is shown in Tab. S2.

C.2. Anti-Distortion Ability

We evaluate the anti-distortion ability of our method in the paper. Here we further consider more real-world image distortions.

Tampering During image transmission, tampering is one of the most common and severest distortions. We randomly tamper (in our implementation, we use some black squares and mask them on the stego images) a certain ratio of the areas in the stego image and calculate the decoding accuracy to measure the robustness against this kind of distortion. The results are shown in Tab. S3. It can be observed that, our method has significantly better robustness against tampering than previous methods. We speculate that, after introducing the transformer architecture into normalizing flow, secret message can be embedded into host images in a manner similar to redundant coding due to the inner-channel feature interaction brought by attention mechanism. This allows for the correct recovery of information even when some areas of the image are tampered. Take the two cases shown in Fig. S5 as example, our method can achieve a high decoding accuracy under both cases. However, the remaining four CNN-based methods have high error rates in the tampered regions. This can to some extent prove that, CNN-based method tends to hide secret message in corresponding areas. Therefore, when a certain area is damaged, the corresponding area of the secret message will also fail to decode correctly. More results are provided in Appendix C.5.

Light Field Messaging Wengrowski et al. [14] consider the message embedding robustness against on-screen shooting, e.g., taking photo of the stego image displayed on a PC screen. Since the quantitative study upon this topic requires a large amount of experiments, such as the influence of different displayers and camera lenses, which are not the main contribution of this paper, here we just provide some qualitative results. We use an iPhone 13 Pro as the shooting camera and the displayer used to show the stego images is a BenQ EW2770QZ with 2560×1440 resolution. As shown in Fig. S6, we choose different shooting distances for a comprehensive demonstration. It can be observed that, with the shooting distance grows, the distortion of Moiré pattern becomes more and more obvious. However, compared with other methods, our method can keep a high decoding accuracy against this kind of distortion. We believe this can again prove the superiority of transformer-based scheme in handling robust steganography task. More results are provided in Appendix C.5.

Figure S5. Decoding results under image tampering. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

C.3. Ablation Study

Since the quantitative results have been provided in Sec. 4.4, here we mainly focus on the ablation experiment implementation details and qualitative comparison results.

IQRT We validate the effectiveness of invertible QR Code transition by removing it from the pipeline, which means we directly tokenize the QR Code image and feed the tokens to the ITF module and then to the AttnFlow model. As shown in Fig. S7, the stego images generated without IQRT have more artifacts. As discussed in the paper, since we only apply one constraint on the transformed QR Code to guarantee its identifiability, the learned transition strategy will tend to help achieve a better steganography quality. However, as shown in Tab. 4, IQRT can lead to a slightly worse decoding accuracy. This is due to the information loss that sometimes happens during the transition process. Two examples can be found in Fig. S2. Overall, we believe this module can effectively improve the stego image quality.

ITF Module The invertible token fusion module learns

Table S3. Decoding accuracy under tampering rate r. The best and second-best results are marked in red and blue colors.

Method	$r = 5\%$		$r = 10\%$		$r = 15\%$		$r = 20\%$		$r = 25\%$		$r = 30\%$		$r = 35\%$	
	TRA↑	EMR↓	TRA↑	EMR↓	TRA↑	EMR↓	TRA↑	EMR↓	TRA↑	EMR↓	TRA↑	EMR↓	TRA↑	EMR↓
ISN†	0.571	2.878	0.349	5.008	0.110	7.137	0.005	9.259	0.000	11.39	0.000	14.24	0.000	16.37
HiNet†	0.575	2.698	0.354	4.725	0.105	6.723	0.007	8.736	0.000	10.78	0.000	13.48	0.000	15.53
StegaStamp	0.915	3.221	0.673	5.080	0.245	6.902	0.029	8.716	0.001	10.54	0.000	12.89	0.000	14.68
StegaStamp†	0.918	3.038	0.732	4.801	0.336	6.539	0.050	8.277	0.001	9.994	0.000	12.30	0.000	14.05
Ours	0.996	0.119	0.995	0.340	0.966	0.755	0.818	1.324	0.533	2.059	0.233	3.192	0.113	4.195

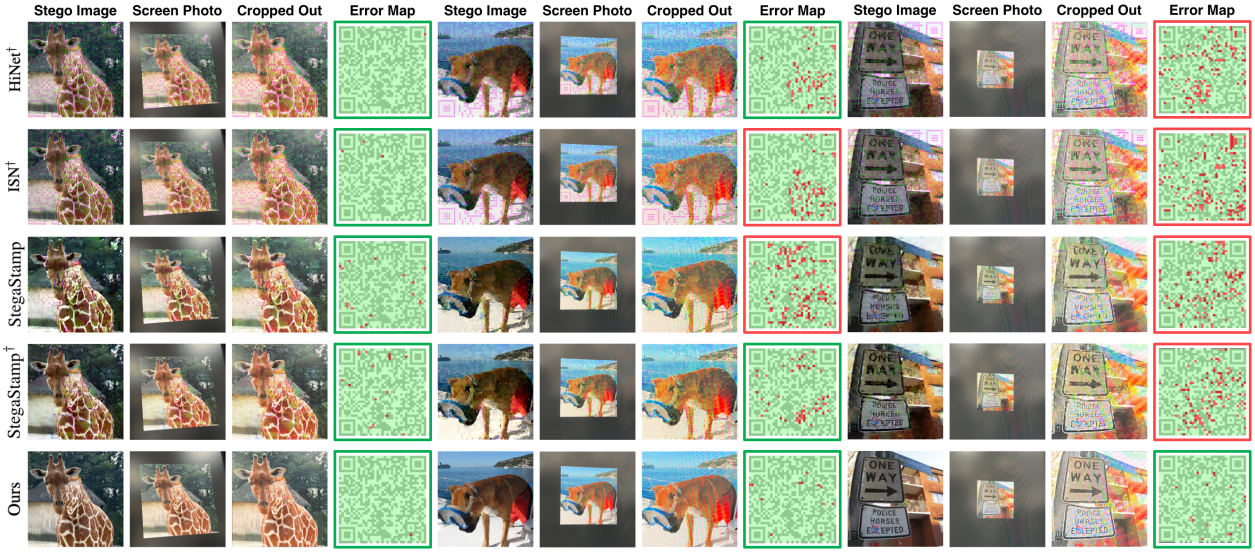

Figure S6. Stego images and decoding results under the distortion of light field messaging. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

a transform matrix for QR Code image tokens. Compared with the invertible 1×1 convolution (IConv) proposed by GLOW [6], our ITF learns a patch-wise (or, token-wise) transformation instead of a channel-wise one. We believe it can rearrange the tokens just like IConv that can re-permute the channels in order to compensate for the insufficient distribution transformation ability of normalizing flow due to the affine-formed functions that have to be adopted for the invertibility of the model. The experiment result proves the competence of ITF module and as introduced in the paper, we empirically find that this module can help derive a better distribution of the artifacts in the stego image. Some results are also shown in Fig. S7, we believe this is to some extent due to the token rearranging brought by ITF.

Cross Attention in AACB We introduce an extra cross-attention item in Equation 3. We make this design because we hope more feature interactions can happen between host image tokens and QR Code tokens. As a result, we incorporate the cross-attention mechanism and add it to the affine transformation function. As shown in Fig. S7, this module can improve the visual quality of the stego image. The results in Tab. 4 also shows that, the model with cross-attention has an around 15% improvement in LPIPS.

Tokenized Representation We incorporate tokenized representation (TR) in ITF module and AttnFlow model. To validate the effectiveness of TR, we remove the ITF module and replace the AttnFlow with CNN-based normalizing flow models (ISN [8] and HiNet [5]). As the results shown in Fig. S7, the stego images generated are similar to that of original ISN and HiNet, containing obvious QR Code-like artifacts. This proves that CNN-based normalizing flow struggles to achieve a high-quality feature learning in the robust steganography task. In contrast, our transformer-based scheme extends the model ability and can help generate stego images with high visual similarity.

C.4. Limitation Analysis and Future Work

Although our RMSteg can achieve the state-of-the-art performance in robust message embedding, it still has some limitations currently. First, as mentioned in Sec. 4.4, our method can help distribute the steganography residual in heterogeneous regions to avoid perceptible artifacts. However, when facing host images with many homogeneous regions, our method can fail to preserve a good visual quality. It can be observed from Fig. S8 that, although the artifacts is mainly concentrated in heterogeneous areas, the ratio of this kind of regions is too small to preserve the overall visual quality. Secondly, although the embedding capacity of RMSteg far exceeds previous methods, it still cannot conceal large-scale secret information, e.g., multiple images.

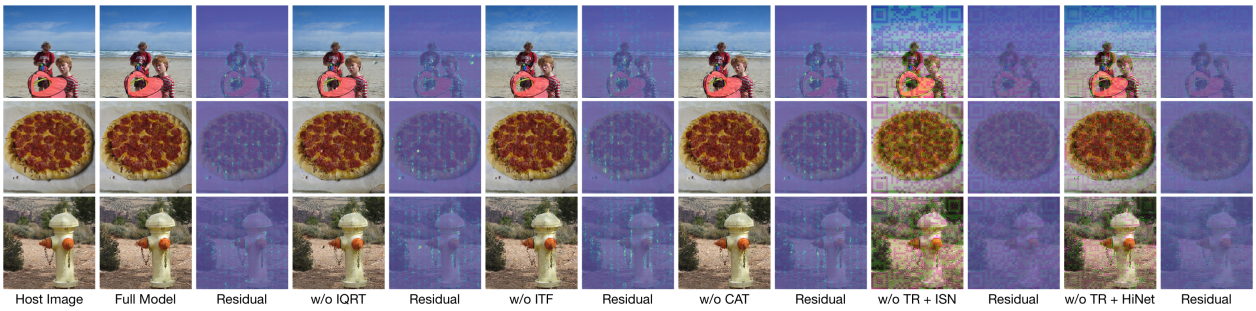

Figure S7. Qualitative results of the ablation study. Zoom in for better observation.

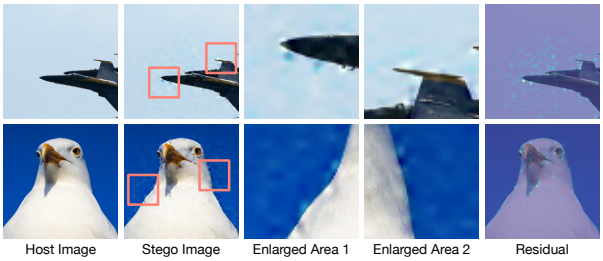

Figure S8. Generated stego images when facing host images with many homogeneous areas.

In summary, we are going to focus on the two aforementioned limitations, i.e., better steganography quality and higher embedding capacity, in the future. We will explore more schemes to extend the performance of transformer-based steganography method. In addition, as mentioned in Appendix C.2, we will consider more kind of real-world image distortions to improve the method's applicability.

C.5. Additional Results

In this section, we provide more qualitative results of the experiments mentioned in the paper and appendix.

IQRT Results More QR Code transition results (corresponding to Fig. 3) are provided in Fig. S9 - Fig. S10.

Steganography Results More stego images generated by different methods (corresponding to Fig. 5) are provided in Fig. S11 - Fig. S13.

Print-Proof Robustness More decoding results under different shooting situations (corresponding to Fig. 6) are provided in Fig. S14 - Fig. S16.

Anti-Tampering More decoding results under image tampering (corresponding to Fig. S5) are provided in Fig. S18.

Anti-Light Field Messaging More decoding results under light field messaging (corresponding to Fig. S6) are provided in Fig. S17.

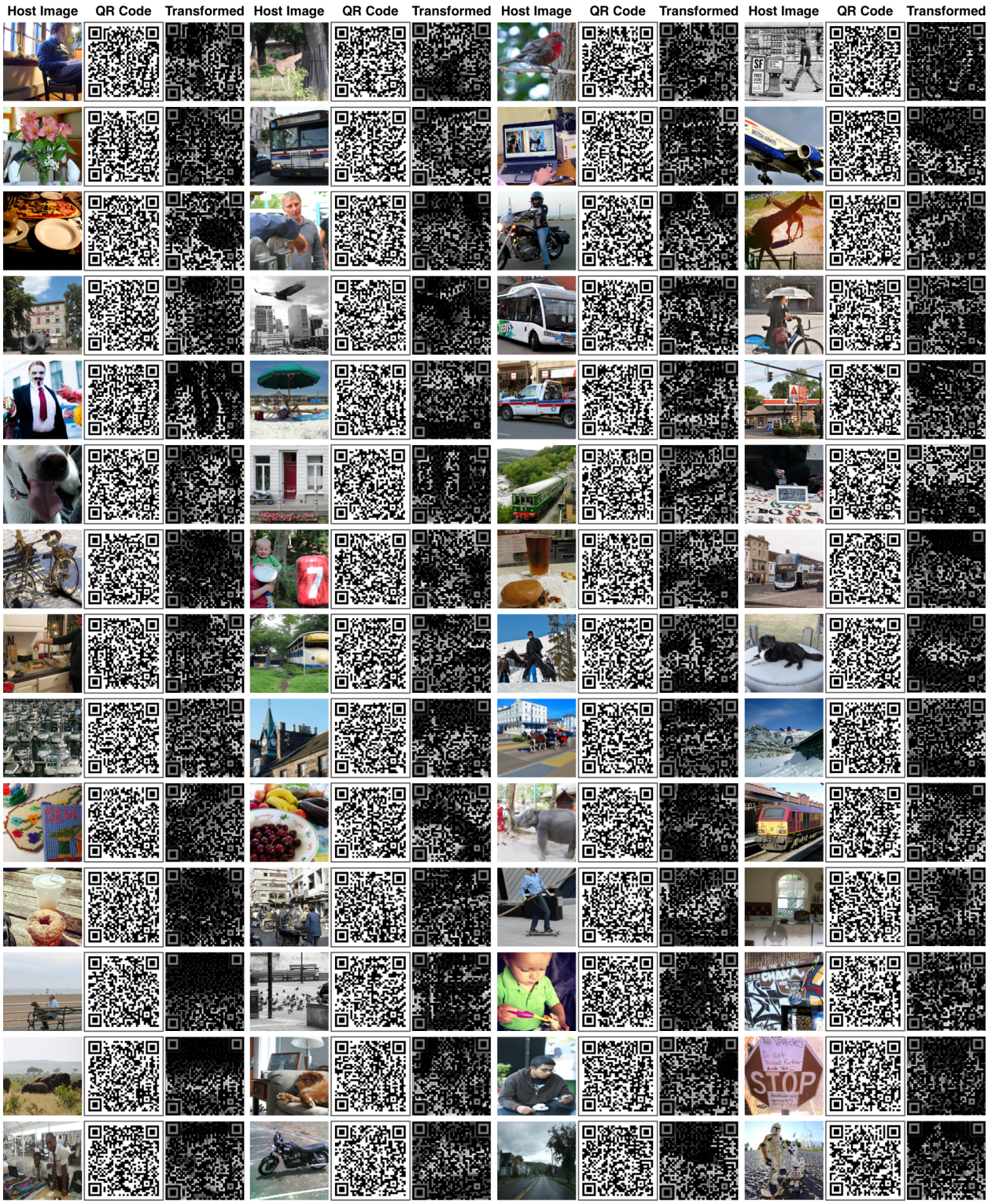

Figure S9. QR Code transition results, the transformed QR Codes are still identifiable.

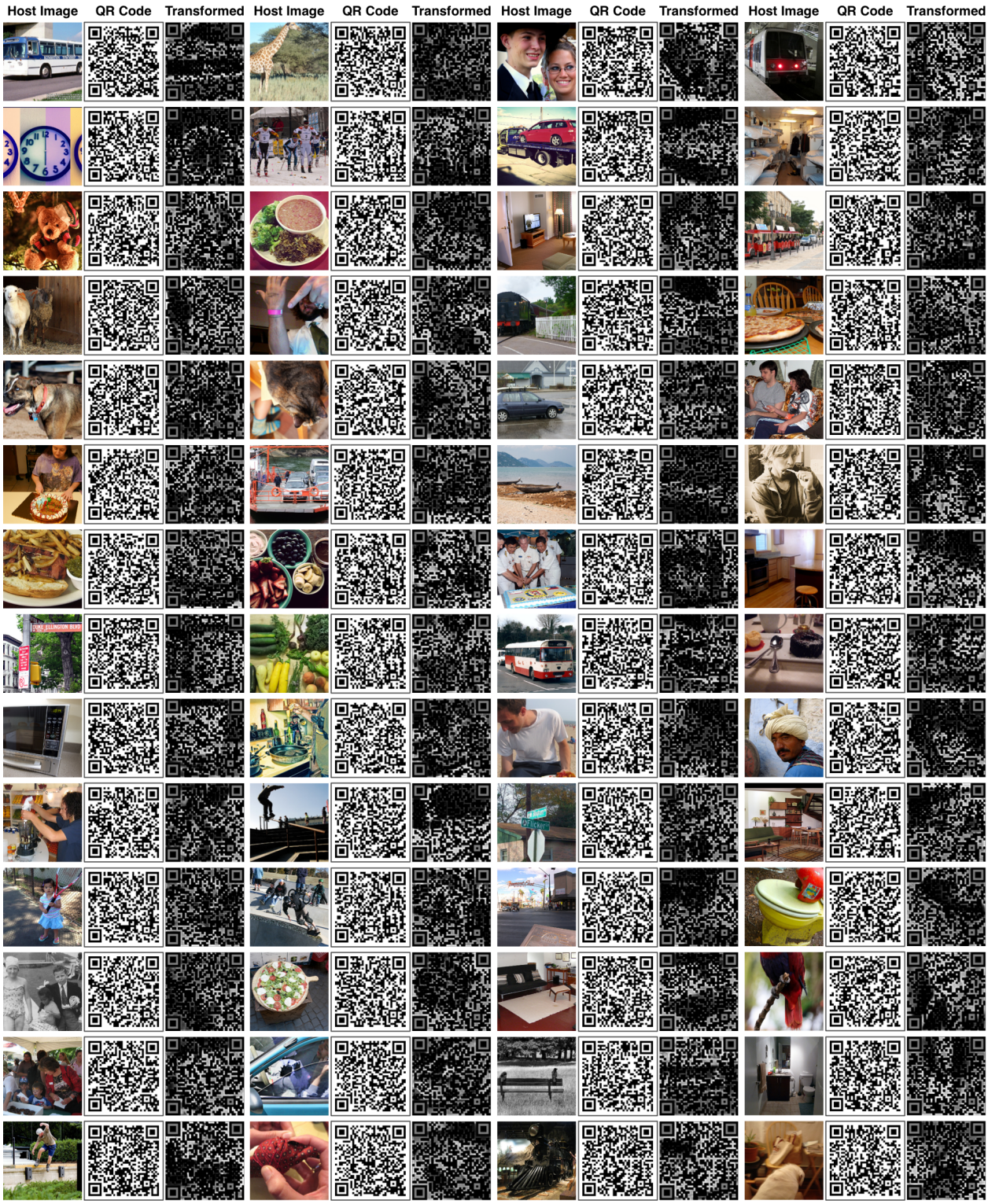

Figure S10. QR Code transition results, the transformed QR Codes are still identifiable.

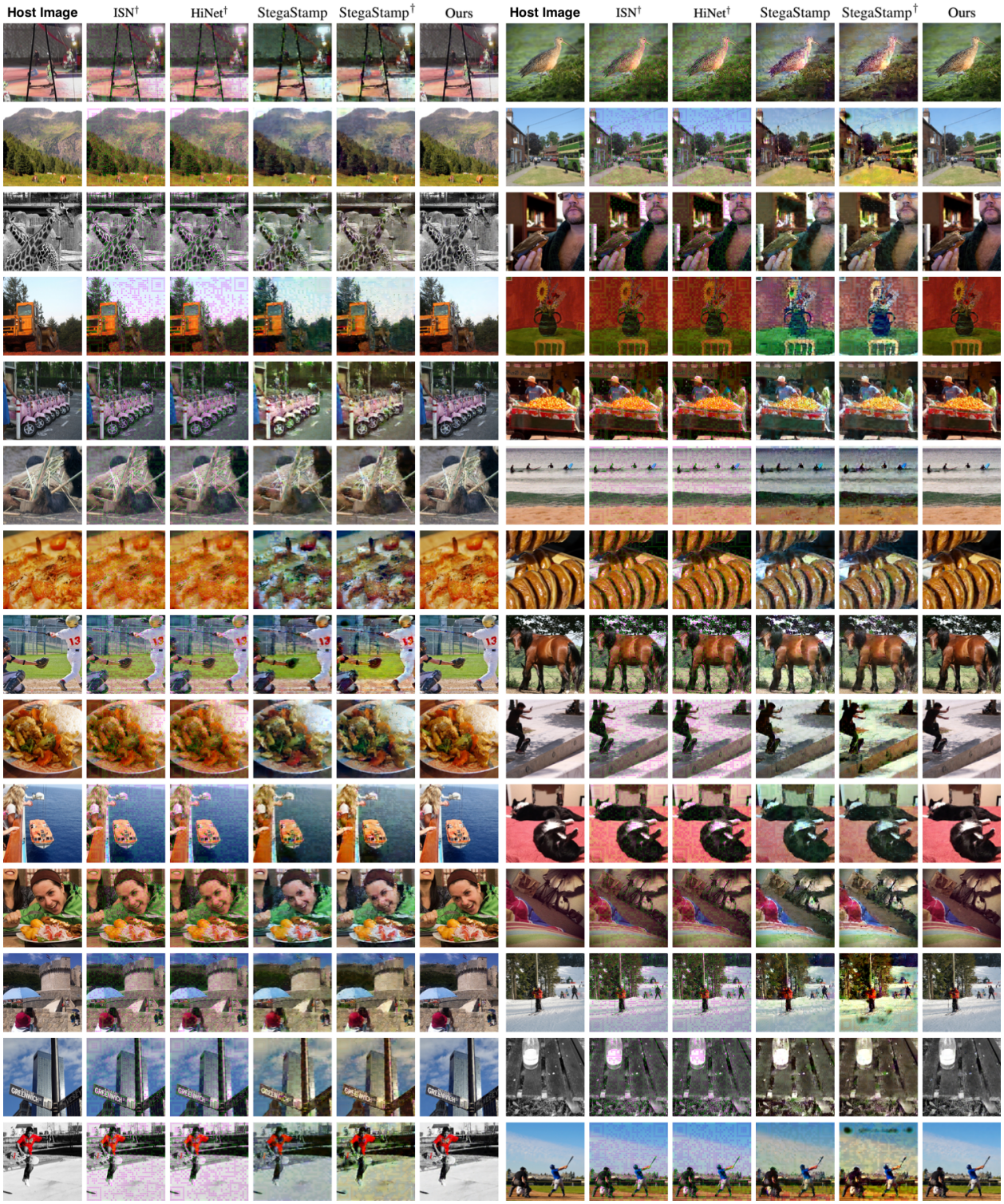

Figure S11. Stego images generated by different methods. Zoom in for better observation.

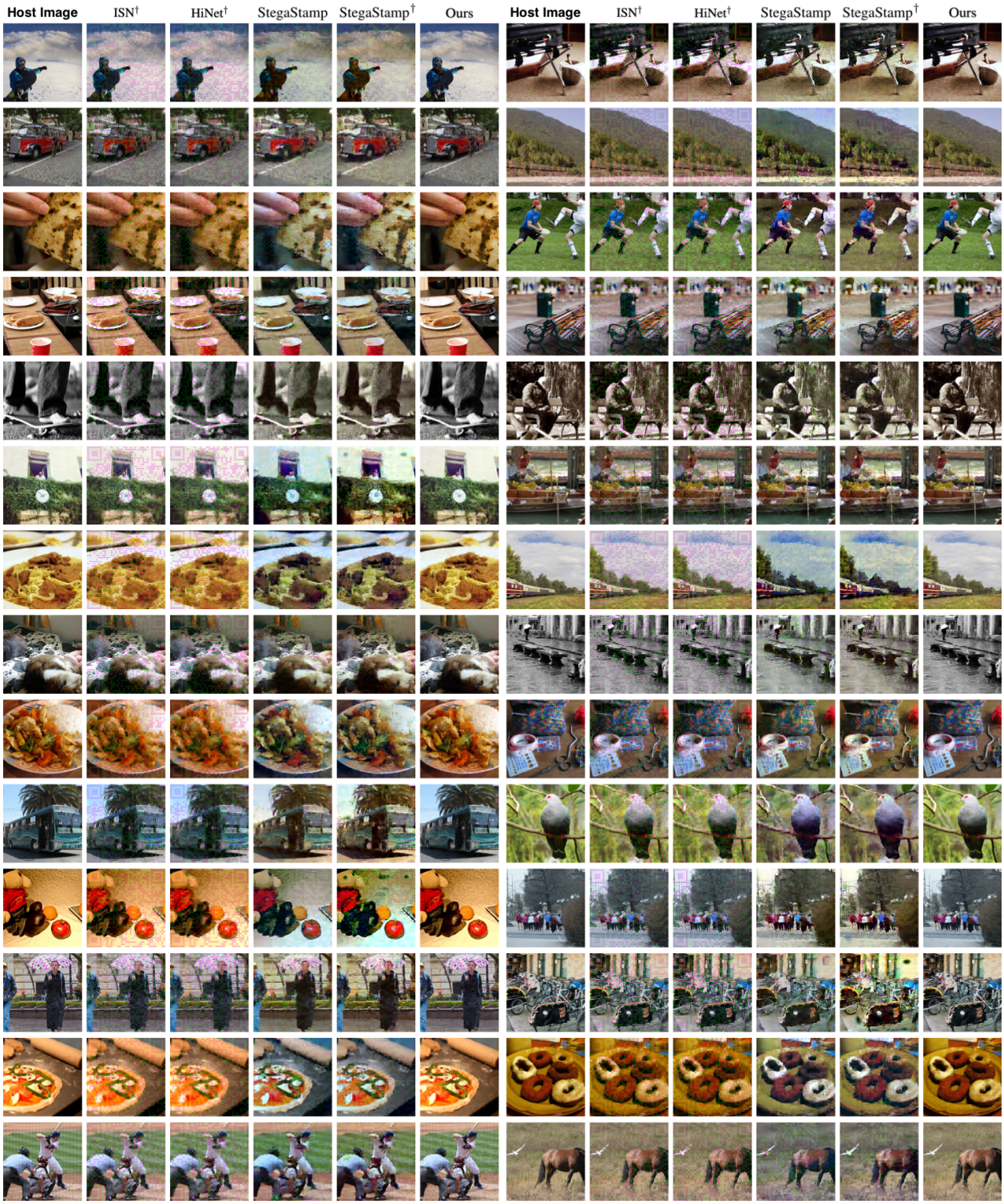

Figure S12. Stego images generated by different methods. Zoom in for better observation.

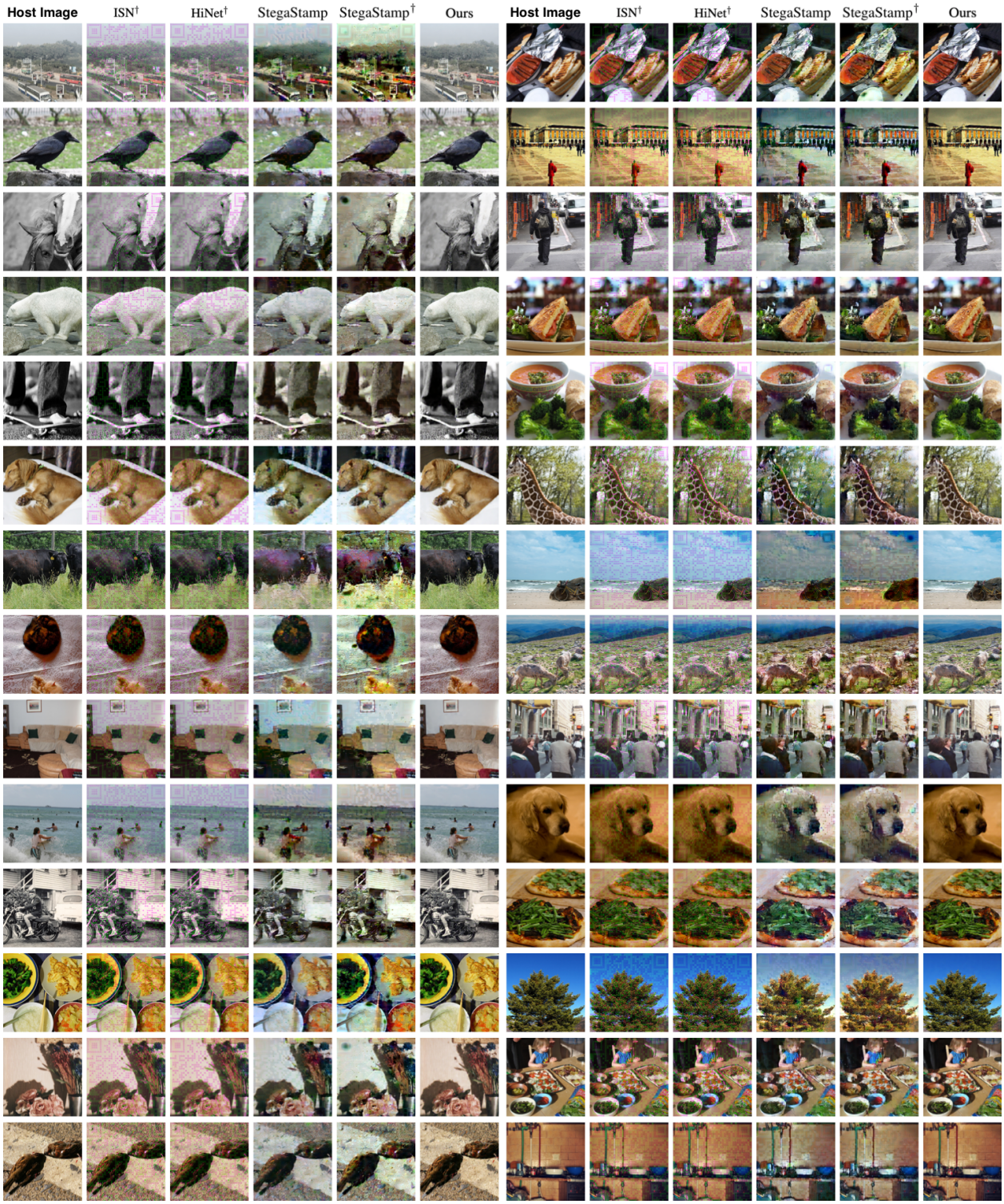

Figure S13. Stego images generated by different methods. Zoom in for better observation.

Figure S14. Decoding results under different shooting distances and angles. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

Figure S15. Decoding results under different shooting distances and angles. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

Figure S16. Decoding results under different shooting distances and angles. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

Figure S17. Stego images and decoding results under the distortion of light field messaging. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

Figure S18. Decoding results under image tampering. QR Codes with green borders can be successfully identified while those with red borders cannot. Zoom in for better observation.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[4] INTSIG. CamScanner: Text and Image Scanning and Recognition, PDF to Word, Document Format Conversion, Online Editor. https://www.camscanner.com.

[5] Junpeng Jing, Xin Deng, Mai Xu, Jianyi Wang, and Zhenyu Guan. Hinet: Deep image hiding by invertible network. In *Proc. IEEE/CVF Intl. Conf. Comput. Vis.*, pages 4733–4742, 2021.

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

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[8] Shao-Ping Lu, Rong Wang, Tao Zhong, and Paul L Rosin. Large-capacity image steganography based on invertible neural networks. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 10816–10825, 2021.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst. (NIPS)*, 32, 2019.

[10] O. S. ZXing. https://github.com/zxing/zxing, 2013. Accessed on Mar. 2018.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd Intl. Conf. Learn. Representations (ICLR)*, 2015.

[12] Hao Su, Jianwei Niu, Xuefeng Liu, Qingfeng Li, Ji Wan, Mingliang Xu, and Tao Ren. Artcoder: An end-to-end method for generating scanning-robust stylized qr codes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2277–2286, 2021.

[13] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 2117–2126, 2020.

[14] Eric Wengrowski and Kristin Dana. Light field messaging with deep photographic steganography. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 1515–1524, 2019.

[15] Mingliang Xu, Qingfeng Li, Jianwei Niu, Hao Su, Xiting Liu, Weiwei Xu, Pei Lv, Bing Zhou, and Yi Yang. Art-up: A novel method for generating scanning-robust aesthetic qr codes. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(1):1–23, 2021.

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 586–595, 2018.