Learning Invisible Markers for Hidden Codes in Offline-to-online Photography

Jun Jia[1*], Zhongpai Gao[2*], Dandan Zhu[2], Xiongkuo Min[1], Guangtao Zhai[1†], Xiaokang Yang[2]
[1]Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
[2]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{jiajun0302, gaozhongpai, ddz, minxiongkuo, zhaiguangtao, xkyang}@sjtu.edu.cn

Abstract

QR (quick response) codes are widely used as an offline-to-online channel to convey information (e.g., links) from publicity materials (e.g., display and print) to mobile devices. However, QR codes are not favorable for taking up valuable space of publicity materials. Recent works propose invisible codes/hyperlinks that can convey hidden information from offline to online. However, they require markers to locate invisible codes, which fails the purpose of invisible codes to be visible because of the markers. This paper proposes a novel invisible information hiding architecture for display/print-camera scenarios, consisting of hiding, locating, correcting, and recovery, where invisible markers are learned to make hidden codes truly invisible. We hide information in a sub-image rather than the entire image and include a localization module in the end-to-end framework. To achieve both high visual quality and high recovering robustness, an effective multi-stage training strategy is proposed. The experimental results show that the proposed method outperforms the state-of-the-art information hiding methods in both visual quality and robustness. In addition, the automatic localization of hidden codes significantly reduces the time of manually correcting geometric distortions for photos, which is a revolutionary innovation for information hiding in mobile applications.

1. Introduction

Scanning QR codes with smartphones provides convenience for people to obtain information from offline to online anytime and anywhere. However, with the increasing demand for the quality of experience (QoE), the unaesthetic appearance limits the application of QR codes in many scenarios, such as interactive visual media, and IP protection of user-generated (UGG). To achieve offline-to-online experiences while maintaining good QoE, invisible information hiding becomes a novel alternative [11, 20]. The core requirement of information hiding in display/print-camera scenarios is to make the information invisible to human eyes but detectable by mobile devices.

The general process of invisible information hiding in display/print-camera scenarios includes five steps as Figure 1: (*i*) encoding information in images, (*ii*) displaying/printing, (*iii*) capturing, (*iv*) locating the encoded information and correcting geometric distortion, and (*v*) decoding. The main challenge of the above process is that the decoding procedure needs to recover the hidden information from photos that contains distortions caused by the camera imaging process. These distortions can be divided into three categories according to their sources: (*i*) from environments, e.g., brightness, contrast, and color distortions), (*ii*) from camera sides, e.g., defocus blur, noise, and compression, (*iii*) from photographer sides, e.g., motion blur and geometric distortion. We define the image containing hidden information as the encoded image. Existing methods [7, 8, 11, 20, 25] can recover hidden information under the above-mentioned distortions. However, these methods assume that the coordinates of the encoded image's four vertices in photos are provided such that the geometric distortion can be corrected by performing perspective transformation. Thus, these methods require preprocessing to locate the encoded image's four vertices and rectify the encoded image from photos taken from different perspectives, which is essential for the success of decoding [8].

Existing preprocessing methods commonly used to remove geometric distortions include two categories: manual locating [8, 25] and automatic locating [7, 11, 20]. Manual locating is to find the four vertices of the encoded image manually, as Figure 1. However, manually locating the vertex coordinates is time-consuming and sometimes is difficult when hidden codes are invisible. Traditional automatic locating requires additional markers such that encoded images with distortions are distinguishable from backgrounds. These methods add visible markers, such as bounding boxes [11] and barcodes [7], around the encoded image. However, the markers break the invisibility of invisible codes. In addition to the traditional automatic methods, Tancik *et al.*

*Equal contribution †Corresponding author

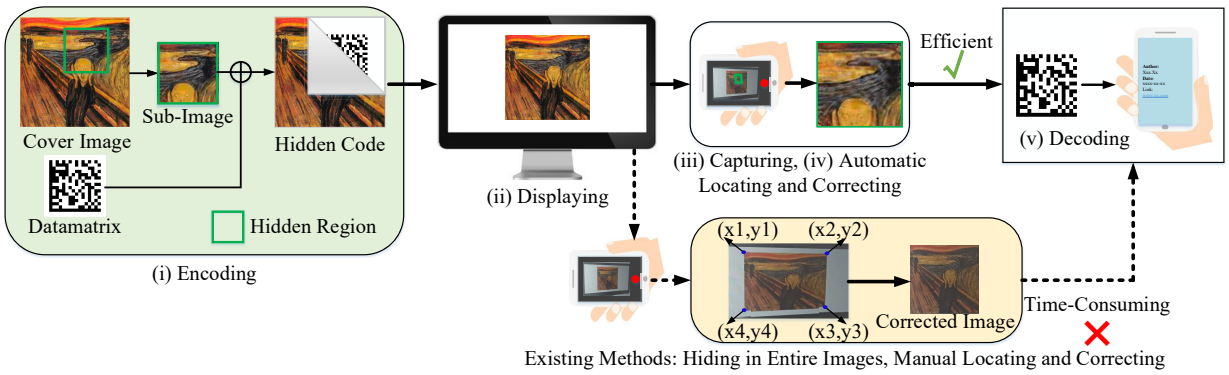

Figure 1. Comparison of the proposed method with the previous methods [7,8,11,20,25]. Compared to the previous methods, the proposed method can automatically locate the accurate position containing the hidden information, which is efficient in mobile applications.

[20] propose StegaStamp, which is an end-to-end trainable framework. The encoder and the decoder of [20] consist of convolution neural network (CNN). During training, StegaStamp uses a series of differentiable distortion operations to process the encoded image and feed the distorted image to its decoder. Thus, the decoder can learn to locate and recover information under different distortions. However, experimental results show that the CNN-based decoder is vulnerable to geometric distortions, thus additional markers (white borders) are still required to locate the encoded image in [20].

This paper proposes a novel invisible information hiding model in display/print-camera scenarios, which can automatically locate the hidden information (data matrix) by learning invisible markers. Inspired by the success of [11, 20], the proposed model is a CNN-based end-to-end framework consisting of an encoder, a distortion network, a localization network, and a decoder. Unlike [7,8,11,20,25], the encoder hides information (data matrix) in a sub-image rather than the entire cover image such that the image area outside the sub-image is considered as backgrounds in photos. The role of the distortion network is similar to [20] and the decoder recovers information from the distorted sub-image. We add a localization network between the distortion network and the decoder for the first time. Different from manual locating or adding artificial markers, the joint training of the encoder and the localization network makes the encoder learn to generate invisible markers around the sub-image while the localization network learns to detect these invisible markers under various distortions, especially geometric distortions. In other words, the optimization goal is to make the markers of the hidden codes invisible to the human eyes, but can be detected by the localization network. Given the detected coordinates of the sub-image, we can remove geometric distortion such that the decoder only needs to recover information from the corrected sub-image. The main contributions of this paper are summarized below:

- We propose a novel invisible information hiding architecture in display/print-camera scenarios, consisting of information hiding, locating, correcting, and information recovery.

- For the first time, we learn invisible markers and develop a localization module in the end-to-end framework for hidden codes. The joint training of the encoder and the localization network generates markers that are invisible to human eyes but detectable by the localization network, which considerably reduces the time of correcting geometric distortions without breaking the visual invisibility.

- To achieve a good trade-off between the detectability of invisible markers, recovery accuracy of hidden codes, and visual invisibility, we propose an effective multi-stage training strategy. A series of loss functions are designed to make the sub-images containing hidden information comfortable to human eyes.

2. Related Work

Barcode With the development of Internet of Things, barcodes have become the most important medium to connect the physical world and the virtual world. However, barcodes are not favorable for their unaesthetic appearances and take up valuable space of publicity materials. To solve these limitations, many aesthetic barcodes are proposed [1–4, 9, 10], which hide barcodes in natural images and preserve visible markers for locating.

Invisible Information Hiding Invisible information hiding includes two major branches: steganography and digital watermarking. Steganography is widely used in the field of information security. According to the requirements of information security, steganography algorithms need high information capacity and security against steganalysis [14]. According to the domain where the information is hidden, steganography can be classified into spatial steganography, e.g., Least Significant Bit (LSB), and steganography in transform domain [12, 16]. Recently,

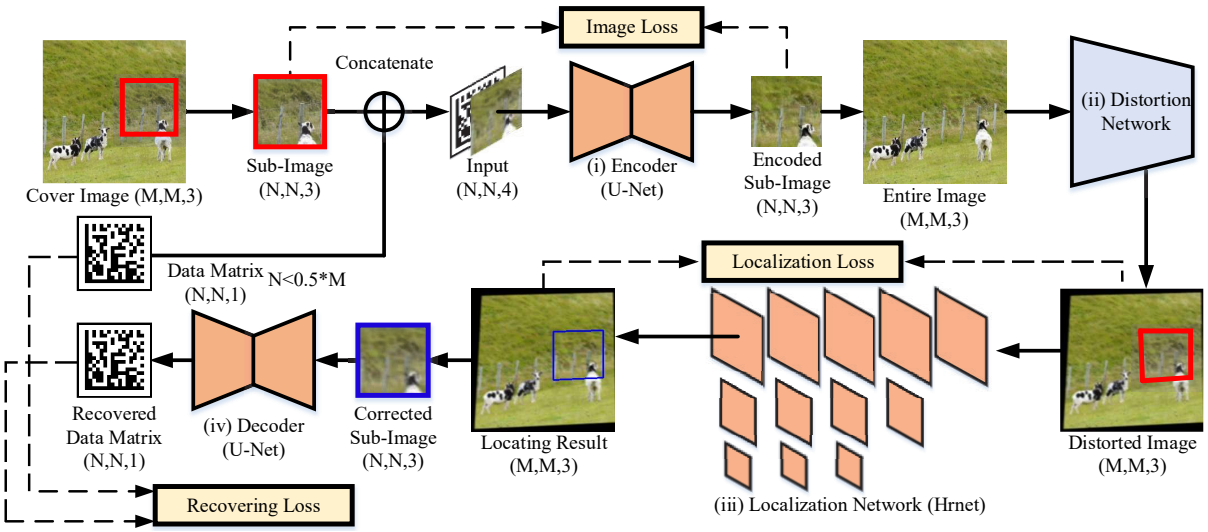

Figure 2. The pipeline of the proposed model. The encoder hides a data matrix in a sub-image of the entire cover image. The distortion network applies differentiable operations to process the entire image containing the encoded sub-image. The localization network locates the accurate position of the distorted sub-image. According to the locating results, the geometric distortion of the sub-image is corrected and the decoder recovers the hidden data matrix from the corrected sub-image.

many methods based on deep neural networks are proposed [19, 22, 25, 26].

Different from steganography, the main application of digital watermarking is the protection of property rights, which requires high accuracy of watermarking recovery rather than high security and information capacity. Digital watermarking also includes spatial watermarking [13, 17], transform domain watermarking [5, 7, 8, 21], and DNN-based watermarking [11, 15, 20, 27, 29]. Recently, some methods are proposed to replace the role of barcodes in offline-to-online messaging [7, 8, 11, 20]. Since these methods do not consider locating hidden codes in the architectures, manually locating hidden codes or adding visible markers is necessary for them to detect the encoded images in practical applications.

3. Methodology

As shown in Figure 2, the proposed model is an end-to-end framework including an encoder, a distortion network, a localization network, and a decoder. The following subsections will describe these modules in detail.

3.1. Encoder

The encoder hides information in a selected sub-image of the cover image, making the hidden data matrix invisible to human eyes. The cover image is a 256×256 RGB image, the sub-image is a 96×96 sub-region of the cover image, and the hidden information is a 96×96 data matrix. We concatenate the data matrix to the last channel of the sub-image to generate a 96×96×4 tensor and send this tensor to the encoder as input. We employ U-Net [18] as the encoder which receives the 96×96×4 tensor and outputs a 96×96 RGB image. After encoding, we replace the original sub-image with the encoded sub-image.

3.2. Distortion Network

The imaging process from digital images to photos may cause quality degradation to the displayed/printed images. Inspired by [11, 20, 29], we insert a distortion network between the encoder and the localization network to simulate the quality degradation caused by the camera imaging process. During training, the distortion network uses differentiable image processing operations to process the entire image after encoding but does not hold any inference parameters. With the help of the distortion network, the localization network and the decoder can learn to resist these distortions.

According to the sources of these distortions, we divide the distortions into three categories: (i) the distortion caused by environmental factors (**brightness, contrast, and color distortions**), (ii) the distortion caused by camera sides (**Gaussian blur, random noise, and JPEG compression**), (iii) the distortion caused by photographer sides (**motion blur and geometric distortion**). The implementation details and settings of these distortions are presented in supplementary materials.

3.3. Localization Network

To locate the encoded sub-image from the entire distorted image, the proposed architecture inserts a localization network between the distortion network and the decoder. The localization network receives the entire distorted image (256×256×3), predicts a heat map for each vertex (64×64×4), and calculates the coordinates of each vertex

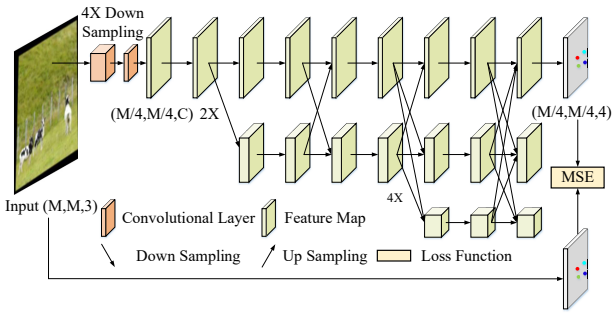

Figure 3. The architecture of the localization network (HRNet [23]).

according to the predicted heat maps H_{pre}. The localization network is a variant version of HRNet [23] which is an architecture with parallel multi-resolution sub-networks and repeated multi-scale fusion. Figure 3 presents a simplified architecture of HRNet. After receiving the distorted image, the localization network uses two convolutional layers of 3×3 kernels and stride size of 2 to process the input image, and then sends the downsampled feature maps to the first high-resolution module of HRNet.

The HRNet used in this paper includes four parallel subnetworks of different scales (Figure 3 presents three parallel sub-networks because of the limited paragraphs). Through multi-scale feature extraction and multi-scale feature fusion, the final feature maps in the first scale (the first row of Figure 3) have enough information to predict a heat map for each vertex. To supervise the regression of the coordinates of the four vertices, we convert the coordinate values to heat maps H_{gt} ($64\times64\times4$) as the ground truth. The ground-truth heat maps H_{gt} are generated by applying 2D Gaussian distribution with a standard deviation of 1 pixel centered on the coordinate value of each vertex. After prediction, we convert the heat maps into coordinate values, calculate a perspective matrix according to the coordinates, and use perspective transformation to correct geometric distortion of the encoded sub-image. The size of the corrected sub-image is $96\times96\times3$.

3.4. Decoder

The decoder receives the encoded sub-image after correction ($96\times96\times3$) and recovers the hidden data matrix ($96\times96\times3$). The structure of the decoder is the same as the encoder [18]. Although geometric distortion has been removed with the help of the localization network, the decoder input still has other distortions compared to the original image, such as noise, blur, and changes in brightness and contrast. These distortions come from the processing of the distortion network, helping the decoder learn to recover the hidden data matrix under these distortions.

In [20], Tancik *et al.* found that it is difficult to make the decoder learn to resist geometric distortion by feeding the geometrically distorted image to the decoder. Furthermore, to make the hidden information decodable under geometric distortion, the encoder of [20] will sacrifice the visual quality of the generated image. Thus, with the help of the localization network, our decoder can focus on recovering information under other distortions except for geometric distortion.

3.5. Training Strategy

Before describing the training strategy, we analyze the optimization objective of each module. The encoder aims to achieve good visual quality for the encoded images, the localization aims to find the target sub-image under distortions, and the decoder is to recover hidden information under distortions except for geometric distortion. However, it is difficult to achieve a good trade-off between these competing objectives if we jointly train the three modules at the beginning. Thus, we propose an efficient multi-stage training strategy to achieve a good trade-off between visual quality and robustness.

3.5.1 The First Stage

In this stage, we only optimize the localization network and the decoder. The loss function to optimize the localization network is defined as L_{loc}:

$$L_{loc}(H_{gt}, H_{hr}) = \frac{1}{w^2}\sum^{w^2}||H_{gt} - H_{hr}||_2, \quad (1)$$

where w is the map width, H_{gt} is the ground-truth heat map and H_{pre} is the heat map predicted by HRNet. The loss function to optimize the decoder is defined as L_{dec}:

$$L_{dec}(dm, dm^{'}) = Cross - Entropy(dm, dm^{'}), \quad (2)$$

where dm is the ground-truth data matrix and $dm^{'}$ is the data matrix recovered by U-Net. The total loss function of this stage is formulated as follows:

$$L_{1-stage} = \lambda_1 * L_{loc} + \lambda_2 * L_{dec}, \quad (3)$$

where λ_1 and λ_2 are set to 30 and 1, respectively.

3.5.2 The Second Stage

After the first stage, our model can accurately locate the encoded sub-images and recover data matrices from the encoded sub-images without geometric distortion. However, the sub-images encoded by the encoder have poor visual quality, because we have not optimized the encoder but the end-to-end training also updates the parameters of the encoder with the supervision of L_{loc} and L_{dec}. Thus, we use L_{pix} and L_{perp} to the second stage as followings:

$$L_{2-stage} = \lambda_1 * L_{loc} + \lambda_2 * L_{dec} + (\lambda_3 * L_{pix} + \lambda_4 * L_{perp}), \quad (4)$$

where L_{pix} is L_2 norm and L_{perp} is LPIPS [28]. λ_1 and λ_2 are set to the same values as the first stage, and λ_3 and λ_4 are set to 1.

3.5.3 The Third Stage

Through the optimization of the above stages, the generated sub-image can achieve good visual quality while the local-

ization network and the decoder can maintain good performance. However, compared to the original sub-image, the generated sub-image has conspicuous markers at its edges and four vertices as Figure 9. From the interpretability perspective, these markers are learned by the end-to-end training of the encoder and the localization. The encoder generates these markers to help the localization network find the encoded sub-image under distortions. Thus, this stage's optimization objective is to decrease the visibility of these markers while keeping the sensitivity of the localization network to these markers. Inspired by [20], we propose a cosine gain scheme to achieve the above objective.

Formally, we define a weight matrix as:

$$M_{cos[i,j]} = \frac{\cos(\frac{s*\pi*d_{x[i]}}{N}+\pi)+1}{2} * \frac{\cos(\frac{s*\pi*d_{y[j]}}{N}+\pi)+1}{2}, \quad (5)$$

where N is the width of the sub-image, s is 4, (i, j) represents the pixel coordinate, and $d_{x/y}$ is the distance to edges along x axis or y axis. When $d_{x/y} \notin [0, \frac{N}{s})$, we set $d_{x/y} = \frac{N}{s}$. We multiply this matrix by L_{pix} and a gain factor f_{cos} to obtain L_{cos}:

$$\begin{aligned} L_{cos} &= f_{cos} * (1 - M_{cos}) * L_{pix} \\ &= \frac{f_{cos}}{N^2} \sum_{[0.N)} (1 - M_{cos[i,j]}) * ||I_{ori[i,j]} - I_{enc[i,j]}||_2 \end{aligned} \quad (6)$$

where f_{cos} is 10 in this paper. The total optimization objective of the third stage is formulated as:

$$\begin{aligned} L_{3-stage} =& \lambda_1 * L_{loc} + \lambda_2 * L_{dec} + \\ & (\lambda_3 * L_{pix} + \lambda_4 * L_{perp} + \lambda_5 * L_{cos}), \end{aligned} \quad (7)$$

where L_{gain} is L_{cos} or L_{gau}, λ_5 is set to 1, and $\lambda_{1,2,3,4}$ are set to the same values as the above stages.

4. Experimental Results

4.1. Dataset

The training set consists of 1,200 images from PASCAL VOC 2012 [6]. The test set consists of 300 images from PASCAL VOC 2012 [6] and 100 images used by RIHOOP [11]. The size of cover images is 256×256×3 and the size of sub-images is 96×96×3. The hidden data matrix is generated by pylibdmtx[1], which is a 16×16 matrix. To align with the sub-images, we resize the data matrices to 96×96 pixels. The total number of bits in the data region is 196, where the length of the encoded data is 96 bits and the other bits represent error correcting codes and padding codes.

4.2. Hyper-parameter Setting

As mentioned in Section 3.5, the training process is divided into three stages. In training, we use 800 epochs, 1,000 epochs, and 300 epochs for the first stage, the second stage, and the third stage of training, respectively. If we train all losses at once, the decoding and locating losses are difficult to converge. The weight parameters of Eq. 7 are set as follows: λ_1=30, λ_2=1, λ_3=1, λ_4=1, and λ_5=1. The settings of the parameters about training strategy are described in Section 3.5 and the settings of the distortion layers are described in Section 3.2. We select Adam as the optimizer and set the initial learning rate to 10^{-4}. In each training stage, the learning rate decays with a cosine annealing schedule until it decays to 4×10^{-5}. The batch size is 32.

[1] https://pypi.org/project/pylibdmtx/

Intensity			Metric		
Defocusing (kernel size)	Motion (kernel size)	Warp (pixel offset)	IoU↑	BER↓	PER↓
3×3	2×2	[-15,+15]	0.9447	0.00%	0.01%
		[-20,+20]	0.9230	1.00%	0.94%
		[-25,+25]	0.9230	4.39%	4.38%
		[-30,+30]	0.8685	11.73%	11.86%
5×5	3×3	[-15,+15]	0.9522	0.10%	0.08%
		[-20,+20]	0.9359	0.10%	0.13%
		[-25,+25]	0.9428	0.00%	0.01%
		[-30,+30]	0.9039	9.08%	9.19%
7×7	4×4	[-15,+15]	0.6354	19.69%	19.54%
		[-20,+20]	0.6418	20.00%	19.87%
		[-25,+25]	0.6269	21.63%	21.76%
		[-30,+30]	0.5512	20.41%	20.68%

Table 1. The IoU, BER, and PER under the combination of defocusing blur, motion blur (degree=15°), and geometric distortion.

4.3. Evaluation Metrics

We use the Intersection over Union (IoU) as the metric to evaluate the performance of the localization network:

$$IoU = \frac{Aera_{pre} \cap Aera_{gt}}{Aera_{pre} \cup Aera_{gt}}, \quad (8)$$

where $Area_{pre}$ is the area of the localized sub-image and $Aera_{gt}$ the area of ground truth. To evaluate the performance of the decoder, we use Bit Error Rate (BER) and Pixel Error Rate (PER) as the metrics:

$$BER = \frac{n_{err}}{len(str)}, \; PER = \frac{p_{err}}{size(datamatrix)}, \quad (9)$$

where n_{err} is the number of error bits, $len(str)$ represents the length of hidden messages, p_{err} is the number of error pixels, and $size(datamatrix)$ represents the resolution of data matrices. In addition, we use PSNR and SSIM [24] to evaluate the visual quality.

4.4. Simulation-based Robustness Test

In this section, we test the robustness of our model to synthetic distortions. In addition to the distortions used in training, we test the generalization ability of our model to unknown distortion by adding distortion categories that are unknown in training. For each distortion category, we set different distortion levels. More details about these settings are presented in supplementary materials. We present the decoding results under different distortion categories in Figure 4. From Figure 4, we can find that our model is vulnerable to JPEG compression and motion blur. In addition,

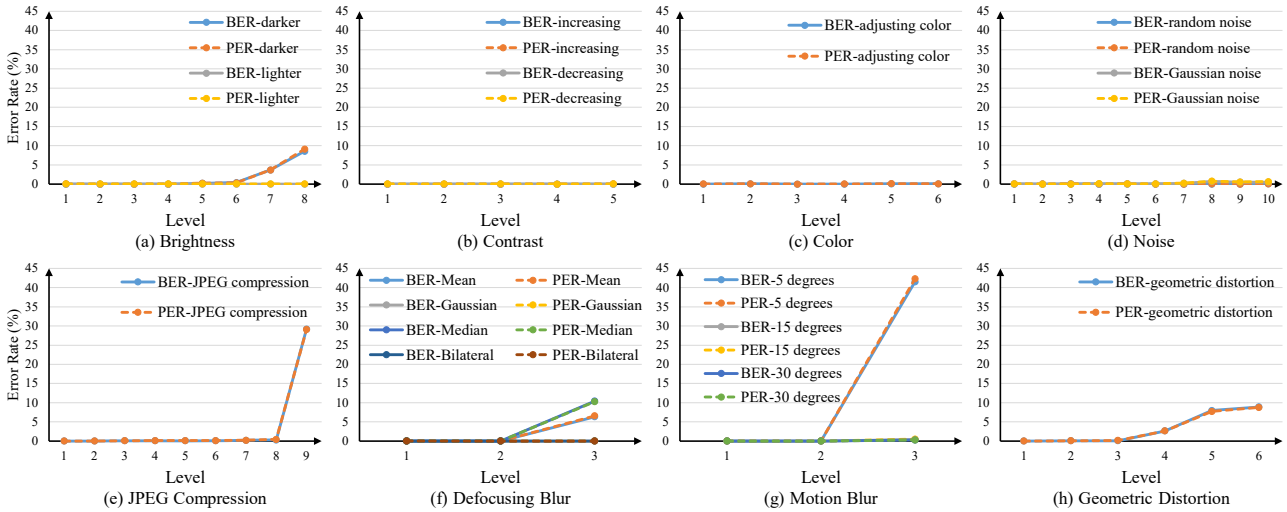

Figure 4. Bit error rate (BER) and pixel error rate (PER) of the decoding results under different kinds of distortions.

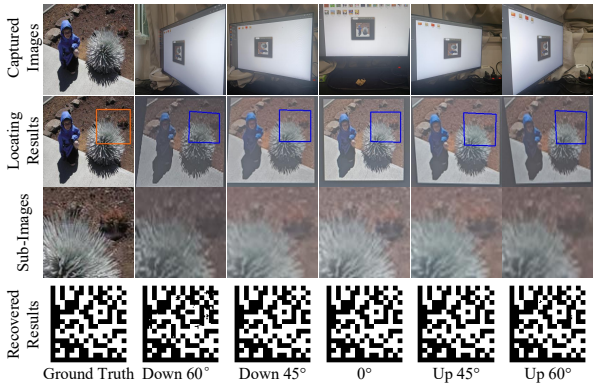

Figure 5. The photographs captured under different vertical shooting angles and the corresponding localization and decoding results.

Figure 4 shows that our model is robust to the unknown distortions in training, such as Gaussian noise, median blur, and bilateral filtering. In this experiment, we find that the errors of locating results will cause decoding failure. For example, when the quality factor of JEPG is 10, the average IoU is only 0.25, so the sub-images sent to the decoder do not contain the full information. We present the correlation between IoU and error rates in Table 1.

4.5. In-the-Wild Robustness

To further validate the practicality of the proposed information hiding model in real scenarios, we capture a large number of photographs under various shooting conditions to test the robustness of our model. In these experiments, we use two smartphones (Redmi Note 9 and iPhone 10) to verify the generalization ability to different cameras. The monitor is Dell S2421HSX and the printer in this experiment is a color consumer printer. We capture this photographs from different shooting angles and different shooting distances.

Horizontal	BER↓	PER↓	**Vertical**	BER↓	PER↓
Left 60°	7.32%	7.39%	**Down 60°**	8.72%	8.70%
Left 45°	0.46%	0.49%	**Down 45°**	1.20%	1.26%
Left 30°	0.64%	0.66%	**Down 30°**	0.54%	0.56%
0°	0.46%	0.45%	**0°**	0.46%	0.45%
Right 30°	0.18%	0.22%	**Up 30°**	0.61%	0.65%
Right 45°	1.71%	1.72%	**Up 45°**	1.28%	1.24%
Right 60°	1.33%	1.28%	**Up 60°**	5.66%	5.70%

Table 2. The decoding results under different shooting angles.

4.5.1 Robustness to Different Shooting Angles

In this experiment, we fix the position of the camera and adjust the monitor's orientation to simulate different shooting angles as Figure 5. We set the distance between the monitor and the smartphone to 40 cm. Because the generated images are displayed at their original resolutions without any scaling, we roughly crop out the regions of encoded images and resize the cropped regions to 256×256 as Figure 5. The first row of Figure 5 presents the original photos and the second row presents the locating results on the resized images. Different from [11] and [20], this process does not need to manually find the accurate vertices of the encoded images or add white borders around the encoded images. The decoding results under different shooting angles are presented in Table 2. When the shooting angle is 60° downward, the error rate is the highest but still less than 10%. The results show that our model is robust enough for shooting from different angles.

4.5.2 Robustness to Different Shooting Distances

In this experiment, we fix the orientation of the monitor and adjust the shooting distances (5 cm, 10 cm, 20 cm, 30 cm, 40 cm, and 50 cm). We present the decoding results in Table 3 and some visualization results in Figure 6. The target sub-images in Figure 6 have different resolutions because of the scaling caused by variant shooting distances. We resize

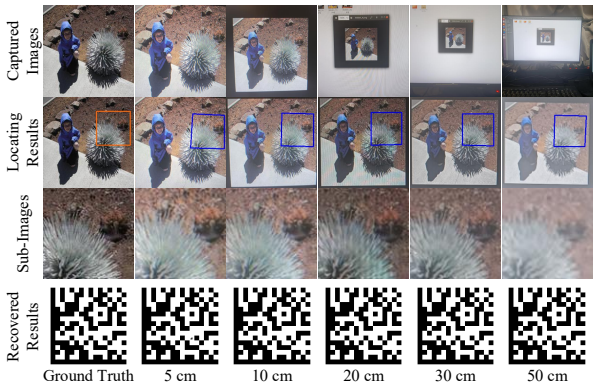

Figure 6. The photographs captured under different shooting distances and the corresponding localization and decoding results.

Shooting Distance	Error Bits ↓	BER ↓	Error Pixels ↓	PER ↓
5 cm	1.05 bits	0.54%	26.25 pixels	0.54%
10 cm	0.40 bits	0.20%	10.40 pixels	0.21%
20 cm	1.40 bits	0.71%	36.65 pixels	0.75%
30 cm	1.10 bits	0.56%	27.50 pixels	0.56%
40 cm	0.70 bits	0.36%	18.00 pixels	0.37%
50 cm	8.00 bits	4.08%	201.4 pixels	4.11%

Table 3. The decoding results under different shooting distances.

these sub-images to 256×256 before inputting them to the localization network. The quantitative results show that our model is robust enough for different shooting distances.

4.5.3 Robustness to Printed Images

In this experiment, we test the robustness of our model in print-camera scenario. We present some examples in Figure 7 where the photos include different lighting conditions and different backgrounds. The average BER and PER of Redmi Note 9 are 1.37% and 1.38%, respectively. The average BER and PER of iPhone 10 are 0.27% and 0.28%, respectively. These results show that our model is robust enough to printed images and generalizes well to different camera models.

Figure 7. The photographs captured after printing. These examples include different lighting conditions and different backgrounds.

4.6. Comparison with State-of-the-art Methods

In this section, we compare our model with state-of-the-art methods: StegaStamp [20] and RIHOOP [11]. Both StegaStamp and RIHOOP are deep learning based information hiding methods that are robust to camera imaging processes. We retrain both StegaStamp [20] and RIHOOP [11] using the training set described in Section 4.1 for fair comparison. We present the comparison results in Table 4. Because the image sizes of [20] and [11] are different from ours, we adjust the ranges of geometric distortion according to the ratio between the resolutions of different models. We present some generated images of these three models in Figure 8. Table 4 shows that our method is much more robust to geometric distortion than [20] and [11]. For visual quality, our method achieves comparable objective scores to [11] and outperforms [20] significantly. Even when error-correcting codes are taken into account, our model is able to hide more bits than [11, 20].

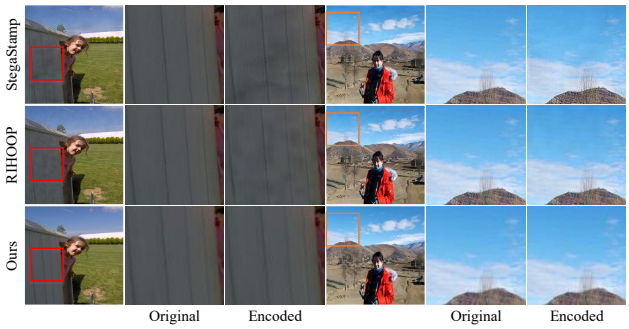

Figure 8. The comparison of the original image and the generated image. [20] and [11] hide the information in the entire images, which cannot control the hidden regions. Thus, the generated images of [20] and [11] have significant quality degradation in low frequency region, e.g. the wall and the sky.

4.7. Ablation Study

4.7.1 The Effect of the Localization Network

In this section, we compare the effect of the localization network with manual locating. When using the localization network, we remove geometric distortion according to the detected coordinates. When using manual locating, we manually find the vertices of the sub-images and remove geometric distortion. The comparison results are presented in Table 5 where *Distortion* 1 represents the combination of brightness, contrast, and color distortions and *Distortion* 2 represents the combination of random noise and JPEG compression. The results show that the localization network may sacrifice a little bit of decoding accuracy but can significantly reduce the preprocessing time compared with manual locating. In mobile applications, the decoding errors caused by locating results can be ignored compared to the significant improvement in running efficiency.

4.7.2 The Effect of the Edge Gain Scheme

In this section, we validate the effect of the proposed edge gain scheme by training two models. The first model does not use the edge gain loss function proposed in Section 3.5.3

Method	Image Size	Capacity	Bits Per Pixel	Bit Error Rate under Different Levels of Geometric Distortion ↓					PSNR ↑	SSIM ↑
				[-20,+20]	[-25,+25]	[-30,+30]	[-35,+35]	[-40,+40]		
StegaStamp [20]	400×400	100	6.25×10^{-4}	14.45%	24.10%	27.85%	32.15%	36.27%	29.82	0.9223
RIHOOP [11]	400×400	100	6.25×10^{-4}	48.20%	51.30%	48.90%	44.10%	50.70%	35.86	0.9712
Ours	256×256	**196**	$\mathbf{2.99\times10^{-3}}$	**2.04%**	**4.01%**	**4.74%**	**7.53%**	**13.77%**	32.95	0.9677

Table 4. The comparison results with state-of-the-art methods.

With Localization	Manual Correction	Distortion 1		Distortion 2		Average Time ↓
		BER ↓	PER ↓	BER ↓	PER ↓	
✓	×	2.10%	2.07%	7.57%	7.48%	**0.014s**
×	✓	0.16%	0.19%	1.22%	1.25%	7s

Table 5. The decoding results with and without the localization network.

(a) The Output of Encoder without the Supervision of Edge Gain Loss Function

(b) The Output of Encoder with the Supervision of Edge Gain Loss Function

Figure 9. The comparison results between the encoder supervised with and without the cosine edge gain loss function.

Model	Edge Gain	IoU↑	BER↓	PER↓	PSNR↑	SSIM↑
Model 1	×	0.8866	11.29%	11.53%	40.64	0.9880
Model 2	✓	0.9318	8.08%	8.32%	41.87	0.9903

Table 6. The Ablation Results of the Proposed Edge Gain.

and the second model uses this loss function. Table 6 shows that the edge gain improves the visual quality of the generated images as expected. In addition, the performance of the localization network and the decoder are also improved. To avoid the randomness of two training processes, we use the edge gain loss to optimize the parameters on the base of the first model. Then, we use the model without the edge gain to generate test images. After that, we decode the images using these two models. The results show that fine-tuning the model with the proposed edge gain can improve the performance of the localization network (IoU↑ increases from **0.8866** to **0.9403**) and the decoder (BER↓ decreases from **11.29%** to **8.59%** and PER↓ decreases from **11.53%** to **8.56%**). The visualized results have been presented in Figure 9, indicating that the edge gain scheme eliminates the effect of visible edges on the HVS.

4.7.3 The Effect of the Selection of Sub-Images

In this subsection, we analyze the effect of the selection of hidden sub-images. For the same image, we hide the same data matrix in two different sub-images: (1) a low-frequency region, such as human face and sky; (2) a high-frequency region, such as animal fur and grass. Compared to hiding information in low-frequency regions, hiding in high-frequency regions increases the visual invisibility of hidden information significantly, but its robustness decreases (BER increases by 9.35%, PER increases by 9.69%, and IoU decreases by 3.75%).

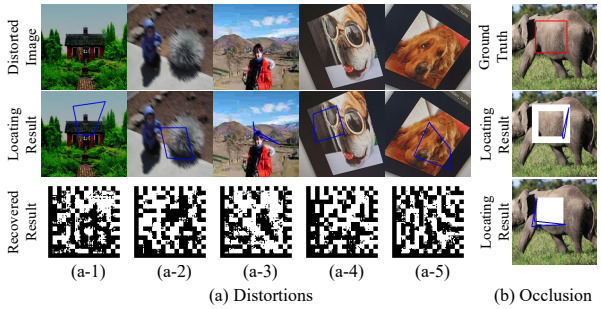

Figure 10. Some failure examples: (a-1) too dark, (a-2) motion blur, (a-3) JPEG compression, (a-4) and (a-5) geometric distortion, (b) occlusion.

4.8. Limitation

This section describes the limitations of our model. In addition to being vulnerable to JPEG compression and motion blur as Section 4.4, the locating errors are too large to recover data matrices when the degree of geometric distortion exceeds the learnable range in training. This phenomenon shows that the learned markers are related to the relative position between the sub-image and the background. We present some failure examples in Figure 10. In addition, our model is vulnerable when the most areas of the encoded regions are occluded.

5. Conclusion

This paper proposes to learn and detect invisible markers for hidden codes in offline-to-online photography (i.e., in display/print-camera scenarios). An effective multi-stage training strategy is developed to achieve high invisibility and detectability. Experimental results show that the our method is robust to general shooting conditions without time-consuming preprocessing to locate and correct encoded images with geometric distortions from the camera imaging process.

6. Acknowledgements

This work was supported by the National Key R&D Program of China under grant No. 2021YFE0206700, National Science Foundation of China (61831015, 61901259, 61901260, 62001289), China Postdoctoral Science Foundation (BX2019208), and the foundation of Key Laboratory of Artificial Intelligence, Ministry of Education, P.R. China.

References

[1] Changsheng Chen, Wenjian Huang, Lin Zhang, and Wai Ho Mow. Robust and Unobtrusive Display-to-Camera Communications via Blue Channel Embedding. *IEEE Transactions on Image Processing*, 28(1):156–169, 2018. 2

[2] Changsheng Chen, Wenjian Huang, Baojian Zhou, Chenchen Liu, and Wai Ho Mow. PiCode: A New Picture-Embedding 2D Barcode. *IEEE Transactions on Image Processing*, 25(8):3444–3458, 2016. 2

[3] Changsheng Chen, Baojian Zhou, and Wai Ho Mow. RA Code: A Robust and Aesthetic Code for Resolution-Constrained Applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3300–3312, 2018. 2

[4] Hung-Kuo Chu, Chia-Sheng Chang, Ruen-Rone Lee, and Niloy J Mitra. Halftone QR Codes. *ACM Transactions on Graphics (TOG)*, 32(6):1–8, 2013. 2

[5] W. C. Chu. DCT-based Image Watermarking Using Subsampling. *IEEE Transactions on Multimedia*, 5(1):34–38, 2003. 3

[6] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 5

[7] Han Fang, Dongdong Chen, Feng Wang, Zehua Ma, Honggu Liu, Wenbo Zhou, Weiming Zhang, and Neng-Hai Yu. TERA: Screen-to-Camera Image Code with Transparency, Efficiency, Robustness and Adaptability. *IEEE Transactions on Multimedia*, pages 1–1, 2021. 1, 2, 3

[8] Han Fang, Weiming Zhang, Hang Zhou, Hao Cui, and Nenghai Yu. Screen-Shooting Resilient Watermarking. *IEEE Transactions on Information Forensics and Security*, 14(6):1403–1418, 2018. 1, 2, 3

[9] Zhongpai Gao, Guangtao Zhai, and Chunjia Hu. The Invisible QR Code. In *Proceedings of the 23rd ACM International Conference on Multimedia*, pages 1047–1050, 2015. 2

[10] Gonzalo J Garateguy, Gonzalo R Arce, Daniel L Lau, and Ofelia P Villarreal. QR Images: Optimized Image Embedding in QR Codes. *IEEE Transactions on Image Processing*, 23(7):2842–2853, 2014. 2

[11] Jun Jia, Zhongpai Gao, Kang Chen, Menghan Hu, Xiongkuo Min, Guangtao Zhai, and Xiaokang Yang. RIHOOP: Robust Invisible Hyperlinks in Offline and Online Photographs. *IEEE Transactions on Cybernetics*, pages 1–13, 2020. 1, 2, 3, 5, 6, 7, 8

[12] Neil F Johnson and Sushil Jajodia. Exploring Steganography: Seeing the Unseen. *Computer*, 31(2):26–34, 1998. 2

[13] I. G. Karybali and K. Berberidis. Efficient Spatial Image Watermarking via New Perceptual Masking and Blind Detection Schemes. *IEEE Transactions on Information Forensics and Security*, 1(2):256–274, 2006. 3

[14] Jia Liu, Yan Ke, Yu Lei, Zhuo Zhang, Jun Li, Peng Luo, Minqing Zhang, and Xiaoyuan Yang. Recent Advances of Image Steganography with Generative Adversarial Networks. *arXiv preprint arXiv:1907.01886*, 2019. 2

[15] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. Distortion Agnostic Deep Watermarking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13548–13557, 2020. 3

[16] Lisa M Marvel, Charles G Boncelet, and Charles T Retter. Spread Spectrum Image Steganography. *IEEE Transactions on Image Processing*, 8(8):1075–1083, 1999. 2

[17] D. P. Mukherjee, S. Maitra, and S. T. Acton. Spatial Domain Digital Watermarking of Multimedia Objects for Buyer Authentication. *IEEE Transactions on Multimedia*, 6(1):1–15, 2004. 3

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, 2015. 3, 4

[19] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. Ssgan: Secure Steganography Based on Generative Adversarial Networks. In *Advances in Multimedia Information Processing – PCM 2017*, pages 534–544, 2018. 3

[20] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible Hyperlinks in Physical Photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2126, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[21] Huawei Tian, Yao Zhao, Rongrong Ni, Lunming Qin, and Xuelong Li. LDFT-Based Watermarking Resilient to Local Desynchronization Attacks. *IEEE Transactions on Cybernetics*, 43(6):2190–2201, 2013. 3

[22] Denis Volkhonskiy, Ivan Nazarov, and Evgeny Burnaev. Steganographic Generative Adversarial Networks. *arXiv preprint arXiv:1703.05502*, 2017. 3

[23] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2020. 4

[24] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5

[25] 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 (CVPR)*, pages 1515–1524, 2019. 1, 2, 3

[26] Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. Steganogan: High Capacity Image Steganography with Gans. *arXiv preprint arXiv:1901.03892*, 2019. 3

[27] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust Invisible Video Watermarking with Attention. *arXiv preprint arXiv:1909.01285*, 2019. 3

[28] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[29] Jiren Zhu, Russell Kaplan, Justin Johnson, and Fei-Fei Li. HiDDeN: Hiding Data with Deep Networks. In *The European Conference on Computer Vision (ECCV)*, 2018. 3