

# Sticking Information in Plain Sight: Encoding and Detecting Hidden Stickers in the Real World

Christina Shatford<sup>✉</sup> and Szymon Rusinkiewicz<sup>✉</sup>

Princeton University, Princeton NJ 08544, USA  
{shatford, smr}@princeton.edu

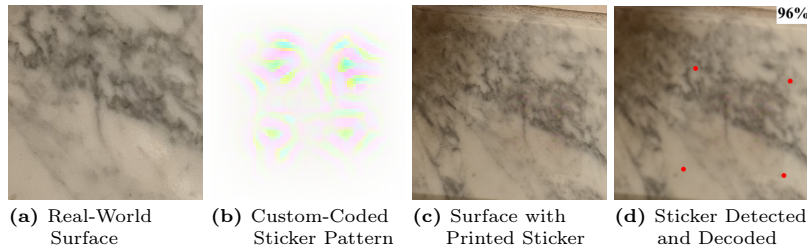
**Abstract.** While there are many techniques (e.g., QR codes) that convey information via visual patterns, some applications would benefit from using imperceptible patterns. We present a method for designing subtle code-conveying patterns that can be printed on transparent sticker paper and applied to real-world surfaces. An image of a scene with an encoded sticker can be captured and the sticker localized and decoded. Thus, our stickers, while mostly unobtrusive to the human eye, are detectable and decodable to our model. We jointly optimize the encoder, localizer, and decoder end-to-end, balancing imperceptibility and accuracy. Our method encodes 100-bit secrets and experimental results show that it is robust to human placement errors, imaging conditions, multiple stickers, occlusions, and wrinkles in stickers. Importantly, this method is easy to deploy and applicable in situations such as object tracking for augmented reality, conveyance of information about tourist sites, and subtle identification of manufactured goods or their components.

**Keywords:** Computer Vision · Fabrication · Perception

## 1 Introduction

Easily incorporating data into the real world is advantageous in a number of scenarios such as consumer-based interactions and industrial manufacturing. Furthermore, incorporating that data in a stylistic or hidden way is also beneficial, especially in situations where the visual appearance of the encoded surface is important. Our work focuses on this type of image-based steganography, or data hiding. We envision applications of this technique ranging from product packaging, to explanatory tags present in museums and at tourist sites, to localization aids for augmented and virtual reality — all situations in which we wish to convey information from the real world to an automated system while not necessarily calling attention to the embedded patterns.

In this work we will present a pipeline for non-permanently modifying real-world surfaces to include hidden data in a way that is unobtrusive to the human eye, but detectable and decodable when fed through our model. First, a user supplies 100 message bits (already augmented with their chosen error correction method) and an image of a real-world surface that they want to encode with their data. The message is then passed with the image to the encoder, which



**Fig. 1:** A real-world example of our method. The sticker pattern (b) was generated by feeding in the image of the real-world surface (a) and a random secret message. The sticker was then printed and applied to the real-world surface to create (c). The image of the surface with the printed sticker was then sent through our localizer to determine the location of the sticker in the image. The predicted location was fed to our decoder and the decoded secret was returned with 96% accuracy—enough to recover the message after BCH error correction with 56 message bits and 40 parity bits.

outputs a pattern. The pattern is then printed on sticker paper and placed over the surface in the environment that corresponds to the surface supplied to the encoder. An image of the scene with the encoded surface can then be captured digitally and sent through our localization and decoding networks, where the sticker is localized and the secret data is retrieved. See Fig. 2 for a visual representation of the pipeline. We accomplish this through jointly optimizing our encoder, localizer, and decoder networks.

A user can select the error correcting scheme they deem most appropriate given their desired applications. Given that our method handles 100 total bits, one possibility is a single BCH codeword with 56 message and 40 parity bits, which can handle up to 5 bit errors. Another good error correction method would be sending 3 BCH codewords of size 31, which would result in 48 message bits overall and up to 3 correctable errors in each codeword.

Existing methods that perform image-based steganography using deep learning [3, 5, 9, 10, 12, 13, 18, 25, 29, 30, 34, 35, 41] rely on being able to directly and precisely change the pixel values of the supplied image when creating the encoded, *container*, image. While those methods do allow for real-world deployment, they do not allow real-world surfaces to be modified to carry encoded information; rather, they must be replaced completely. The novelty and main advantages of our method are that it takes human placement error into account, is a non-permanent modification, and does not require printing of the whole container image (i.e., the background image with the encoded data). Instead, only the secret encoding is printed, which has a small visible footprint. This is beneficial, as printers have a limited color gamut and cannot always accurately recreate background images; placing printed high-opacity images back in the environment they were photographed from makes hiding the printed sections difficult. Thus, our method results in stickers that are easy to apply and simple to remove from a surface if the information they encode needs to be updated for

any reason. In addition, we also present a framework for multi-sticker detection and provide extensive real-world tests under varying imaging conditions.

In this paper, we discuss the encoder, localization, and decoder networks used, and describe a multi-phase training protocol that jointly optimizes for accuracy of decoding, imperceptibility of the encoded sticker, and robustness of localization (Sec. 3). Training includes appropriate augmentations to account for not only image degradations, but also human error in sticker placement. Our results (Sec. 4) demonstrate both the high decoding accuracy and imperceptibility of the patterns. In particular, our work compares favorably to previous work that requires printing an entire image with an embedded pattern and adding a white border, rather than just applying a transparent sticker (Sec. 4.3). A variety of ablation studies (Sec. 4.4) motivate our design choices.

## 2 Related Work

### 2.1 Barcodes

Being able to encode information or data into a scene is desirable in many applications. The variety of schemes for 2D barcodes, and in particular Quick Response (QR) codes, have exploded in popularity due to their high data capacity and ease of use when decoding. However, a drawback of QR codes is their stark black and white appearance. Their appearance can limit their applicability in situations where visual aesthetics are important. Work has been done to improve the appearance of QR codes, with approaches that include creating halftone QR codes [6] and embedding or blending QR codes into images [8, 16, 32, 33, 37]. There has also been work done on creating other barcode schemes that can be embedded into images without having to include all the modules necessary for QR codes [20]. However, even the best of these methods result in patterns that remain easily visible and potentially distracting to human observers.

### 2.2 Steganography

The goal of steganography is to hide data in some medium such that its presence is undetectable, and potentially undecodable, to those who come across it without prior knowledge of its whereabouts. We will focus on instances where steganography is used to hide data in images. In those cases, a *cover* image is encoded with information to become a *container* image.

Many works have implemented image-based steganography utilizing deep learning to both encode and decode hidden messages [3, 5, 9, 10, 12, 13, 18, 25, 29, 30, 34, 35, 41]. In particular, StegaStamp [25] and Jia *et al.* [13] inspired our work. StegaStamp encoded  $400 \times 400$  images with 100 bit secrets and introduced differentiable image perturbations before decoding container images. They fine-tuned an off-the-shelf semantic segmentation network for localization; however, the container images needed an added white border to be properly localized and decoded. The pipeline introduced in [13] allowed for  $96 \times 96$  subsections to be

encoded in larger  $256 \times 256$  images. They also introduced a localization network that took in potentially distorted images and sent predicted encoded subsections to the decoder. Our method is novel in that we do not assume that the container image will be created digitally. Rather, we assume that it will be created in the real world through applying an encoded sticker to a surface. Given our goal, we have to take into account human error in placement and cannot assume that a container image will be created using pixel perfect alignment.

### 2.3 Watermarking

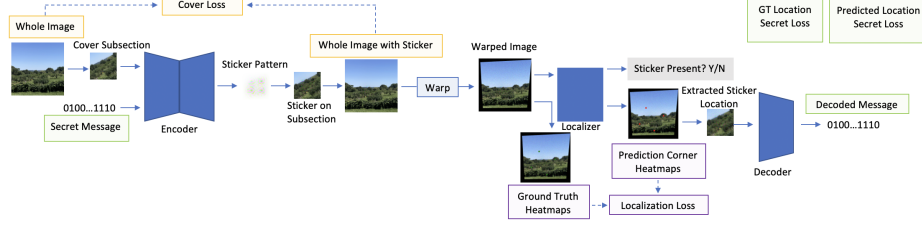
Digital watermarking is another form of embedding hidden data in digital images. It is distinct from steganography in that it is typically used to link image ownership, so robustness to image perturbations and distortions is of more importance than remaining perceptually invisible to the human eye [27]. In addition, the capacity of the message is typically of less importance than in steganography. Digital watermarking can be done using a variety of methods; those that utilize deep learning [1, 14, 17, 19, 21, 38, 41] are closest to our work. Digital watermarking also assumes that the cover image can be directly modified.

Imprinto [7] is an example of non-digital watermarking; an infrared inkjet watermarking technique is used to print invisible patterns on paper. While it allows patterns to be added with handheld printers, if a user no longer wants a printed pattern on a surface, it can be difficult to remove. In contrast, with our method, a user can easily remove a sticker if they no longer want it on a surface. We also do not rely on infrared technology to make our patterns imperceptible.

### 2.4 Real-World Adversarial Attacks

In contrast to the underlying goal of steganography and watermarking to hide information so that it can later be decoded, some real-world adversarial attacks aim to add visual information to prevent accurate classification. Work has shown that adding stickers or patches to a camera lens, scene, or face can alter how the scene or face is classified [4, 15, 23, 28]. There has also been work on generating 3D adversarial objects [2]. While the goals of those works are distinctly different from ours, they also take robust real-world deployment into account. In [4], adding a patch—trained to optimize the probability of a specific target classification—to a scene could make the object in the scene become misclassified as that target. The authors show a decrease in the performance of their patches in the real world versus simulation, something they hypothesize is the result of not optimizing for the printability of the colors of the patches. In contrast, [23] defined a non-printability score to make their optimized adversarial objects more robust for real-world deployment. In our work, we handle color management through printer calibration and defining a color model that we use during training to mimic placing a sticker on an image.





**Fig. 2:** Overview of the encoding/decoding pipeline during stage three of training. Labels of the same color correspond to the same loss component.

### 3 Joint Training of Encoder, Localizer, and Decoder

#### 3.1 Data Generation

For background images, we use the ADE20K dataset [39, 40], which contains 27,574 images. We randomly split the dataset into 23,437 training images and 4,137 testing images. All training and testing background images are resized to  $128 \times 128$  pixels. For each image, we randomly assign a location that a sticker should be placed during training or testing. For a given sticker location, a random height and width, center coordinate, and rotation are generated. The height and width are constrained to be even integers between 32 and 64 pixels and the rotation is set to be between -30 and 30 degrees. We ensured that every sticker is fully within the bounds of its background image. We randomly generate 100-bit secret messages during training and testing. Note that we randomly generate the location of a sticker for a specific background image once, and then use that location-image pair during training. In contrast, we randomly generate new 100-bit binary messages using `torch.randint()` for every location-image pair in every epoch.

#### 3.2 Network Architecture

We use the same general architecture for our encoder and decoder as [25], with some small changes. Thus, we use a UNet [22] for our encoder and a series of convolutions and linear layers for our decoder. For localization, inspired by the localization network in Jia *et al.* [13], we use a variant of HRNet [24, 26, 31]. See our supplementary material for architecture details.

#### 3.3 Training Strategy

In order to balance optimizing for localization accuracy, decoding accuracy, and imperceptibility, we train in stages. We begin by solely optimizing for decoding accuracy at the ground truth sticker location. Thus, we start with having every training example include a sticker, and then change to randomly including stickers after we start optimizing for localization. In order to make the decoder more

robust, instead of sending perfectly cropped encoded images to the decoder, an affine transformation is performed, where we randomly shift the background image by  $\pm 0.05 \cdot 128$  pixels in each of the horizontal and vertical directions. We also randomly rotate the background image by  $\pm 5$  degrees and randomly scale the image between  $(0.98, 1.02)$ . This is done so that when an image is cropped using the ground truth coordinates to get the sticker, the sticker is likely not perfectly in frame. We find that this leads to stickers with patterns that hold most of their visual information close to the center, with softer edges that gradually fade to fully transparent.

The possible augmentations during training are a small perspective warp, random changes to brightness, contrast, and saturation, all between  $\pm 50\%$ ; Gaussian blur with sigma between 0.1 and 1; and/or random Gaussian noise. Each type of augmentation independently has a 75% chance of being added to an image. We found that when we removed any of those augmentations, the patterns that develop have similar or lower average perceptual distance scores, as measured by version 0.1.4 of LPIPS [36]. However, either the average decoding accuracy or detection rate is lower, when testing with the missing augmentation.

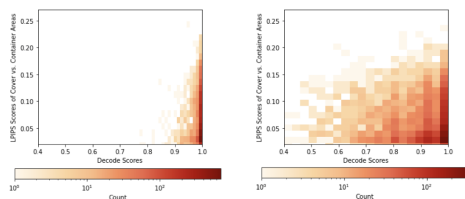
The first stage of training ends after 120 epochs, with the average decoding accuracy of ground truth sticker locations being approximately 95%. We then start the second stage of training by updating the loss function to include a cover loss that measures sticker imperceptibility. The weight of the cover loss starts small, and increases in steps as training continues. After about 300 more epochs, the decoding of ground truth sticker locations reaches average accuracies around 80%, and we enter stage three by beginning to train the localization portion of the network from scratch. We update the loss function to include measurements for localization accuracy. After a few epochs, we add in loss associated with decoding at the predicted sticker location.

The loss function we use is:

$$L = \lambda_1 L_{gtLocDecode} + \lambda_2 L_{heat} + \lambda_3 L_{predLocDecode} + \lambda_4 L_{cover}. \quad (1)$$

We adjust the weights of the components of the loss throughout training. During stage one,  $\lambda_1$  is 1, and  $\lambda_2$ ,  $\lambda_3$ , and  $\lambda_4$  are all 0. In stage two,  $\lambda_1$  is 1,  $\lambda_2$  and  $\lambda_3$  are 0, and  $\lambda_4$  increases from 0.01 to 40. In the third stage, the weights are adjusted such that the losses are roughly balanced with each other, ending with  $\lambda_1$  set to 0.2,  $\lambda_2$  set to 100000,  $\lambda_3$  set to 2, and  $\lambda_4$  set to 80. In addition, we double the weight of the loss associated with the top 4 samples in each batch that have the largest LPIPS scores. See our supplementary material for additional hyperparameter analysis for each  $\lambda$ , as well as results from experiments where training started from stage two or stage three.

We use binary cross-entropy (BCE) with logits loss for measuring loss associated with the decoding of stickers in ground truth locations ( $L_{gtLocDecode}$ ). We calculate the localization loss ( $L_{heat}$ ) as the Mean Squared Error (MSE) between the ground truth and predicted four corner heatmaps. We also use BCE with logits loss for measuring the accuracy of decoding in the predicted sticker location ( $L_{predLocDecode}$ ); we only calculate it when a sticker is predicted to be



**Fig. 3:** LPIPS scores (lower is better) vs. Decoding accuracies (higher is better) on the test set, with no augmentations added (left) and with augmentations added (right).

present in a background image. The cover loss ( $L_{cover}$ ), which measures the imperceptibility of a sticker in an image, is calculated by using Learned Perceptual Image Patch Similarity (LPIPS) [36].

See Fig. 2 for a visual representation of the training pipeline. Note that it depicts training at stage three. In stages one and two, the ground truth heatmaps are used to extract the sticker locations that are fed to the decoder. We found that it was necessary to have two losses associated with decoding accuracy, as when  $L_{predLocDecode}$  was not included, it led to a decrease of approximately 6.4 percentage points in decoding accuracy, with augmentations.

## 4 Results

### 4.1 Simulation

Our model achieves an average detection rate of 99.95% and 97.24% on the test set, for non-augmented and augmented images respectively. It also achieves an average decode accuracy of 99.14% and 89.16% when sending the predicted sticker locations to the decoder, for non-augmented and augmented images respectively. The possible augmentations are the same as those outlined in Sec. 3.3. We measure detection rate as the number of true positives divided by the sum of true positives, false negatives, and false positives, where true positives are stickers that are localized with an IoU greater than 0.5. The average decoding accuracy is calculated by averaging the individual decoding accuracies of images that were detected as having stickers. The average LPIPS score achieved is approximately 0.0632, as measured between  $64 \times 64$  cover and container subsections. See Fig. 3 for histograms of LPIPS scores vs. decoding accuracies. They suggest that the stickers are able to blend well with most backgrounds in the dataset, while maintaining a high decode accuracy, except under more extreme augmentations.

See our supplementary material for examples of sticker patterns placed on images in simulation. While the patterns that develop are not completely invisible when placed on backgrounds, they are able to blend in fairly well, especially in areas that are not low frequency or uniformly light in color. While these results reflect  $128 \times 128$  background images, in simulation, stickers only have to be within the range of  $32 \times 32$  to  $64 \times 64$  to be found within an image with resolution

higher than  $128 \times 128$ . In such cases, we use a sliding window approach for sticker localization. We have tested this on images up to  $400 \times 400$  and have found that the stickers were still consistently and accurately localized and decoded.

## 4.2 Real-World

For real-world experiments, all stickers are printed using a Canon Pixma G620 printer on MACO matte clear sticker paper. An iPhone 15 Pro is used to capture all images. For printer calibration, Calibrite ColorChecker Studio was used to generate a printer profile.

See Fig. 1, Fig. 4, and our supplementary material for examples of stickers placed on real-world surfaces. In these examples, the stickers are placed approximately over the cover subsections. The stickers are photographed in the real world, the resulting images are fed through our localizer, and the localized container subsections are sent through the decoder.

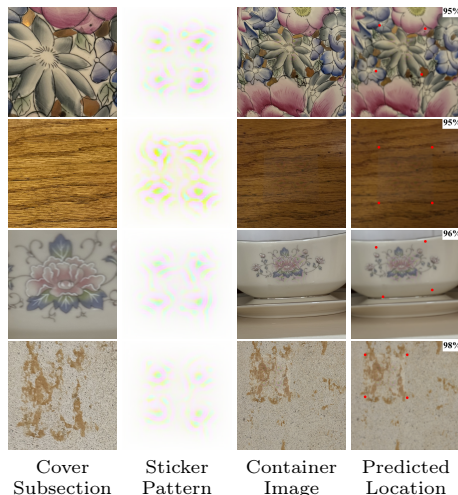
For our real-world test set, we use 34 different surfaces, taking between 1 to 3 pictures of each container surface with varying imaging conditions, for a total of 84 images. The average decoding accuracy is 93.10%, and the median is 94%. In practice, we find that decoding accuracy can vary depending on the characteristics of the background surface. Some backgrounds can lead to worse decoding accuracies when they contain high complexity components, highly saturated colors, or colors or patterns that are a close match to some subset of the sticker. We do not calculate LPIPS scores based on this test set, since imaging conditions and misalignment would lead to artificially higher scores.

In all real-world examples provided in our paper and supplementary material, the stickers are approximately  $2 \times 2$  inches ( $5 \times 5$  cm). However, sticker patterns can be generated for real-world surfaces that are both smaller and much larger than that size. In practice, the printed stickers do not have to match the physical dimensions of the cover image.

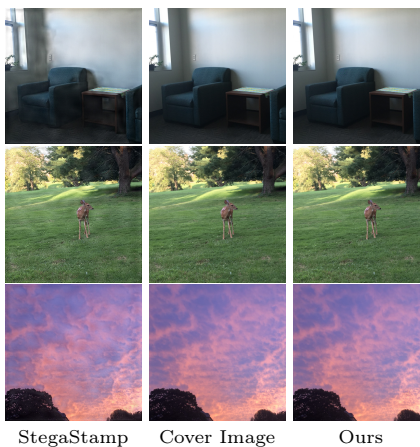
## 4.3 Comparison with State-of-the-Art Method

**Simulation** When encoding using StegaStamp [25], a residual image is generated by the encoder, which is then concatenated with the cover image to create the container image. See Fig. 5 for examples of images encoded using StegaStamp versus our method. Note that StegaStamp uses a complete  $400 \times 400$  pixel image for the cover image, whereas our method uses at most a  $64 \times 64$  subsection of the image. However, that container subsection can still be decoded by sending the  $400 \times 400$  container image through our localizer/decoder using a sliding window approach.

Since a StegaStamp container image is created by directly manipulating the pixels of a cover image, we explore how misalignment influences decoding accuracy. We create misalignment by generating a residual for a cover section, and then concatenating the residual with a modified cover section, which is moved a certain number of pixels diagonally when compared to the original cover section. The distance we shift is the Euclidean distance each pixel moves, which



**Fig. 4:** Examples of our method in the real world.



**Fig. 5:** Examples of images encoded using StegaStamp versus our method.

requires bilinear interpolation, as we do not assume the horizontal and vertical shift components are integers.

For our cover images, we use 2,500 randomly selected images from MIR-FLICKR [11], which is StegaStamp’s training dataset. For testing StegaStamp, we resize images to be  $430 \times 430$  and then take the central  $400 \times 400$  subsections as cover images. For our method, we resize images to be  $430 \times 430$  and take the central  $64 \times 64$  subsections. We use a constant message to generate a residual/sticker for a cover subsection and place it varying distance away from location of the cover image to create container images. We feed the container images directly to their respective decoders, without augmentations or localization.

The average decoding accuracy per number of pixels shifted is reported in Fig. 6a. For each number of pixels shifted, we perform the shift in each of the four diagonal directions and then average the results. We find that StegaStamp is able to maintain its decoding accuracy when the amount of placement jitter is low, between 1-3 pixel. Decoding accuracy sharply drops when the misalignment is greater than or equal to 4 pixels. In contrast, the amount of misalignment does not drastically influence the average decoding accuracy of our method.

We also compare how visible the encoded secret is with an increasing amount of placement jitter. The setup is the same as described above, except we measure average LPIPS scores instead of decoding accuracies. In addition, we resized our stickers and placement areas to be  $400 \times 400$  before blending them and calculating LPIPS scores. See Fig. 6b for the results.

Overall, we find StegaStamp’s LPIPS scores increase substantially after only 1-2 pixels of misplacement. In contrast, our method’s LPIPS scores increase by a much smaller amount and are not substantially different, even after 20 pixels

of misalignment. Note that these LPIPS scores come from comparing  $400 \times 400$  sections, instead of  $64 \times 64$  sections, resulting in a different scale for these scores than the ones seen in Sec. 4.1.

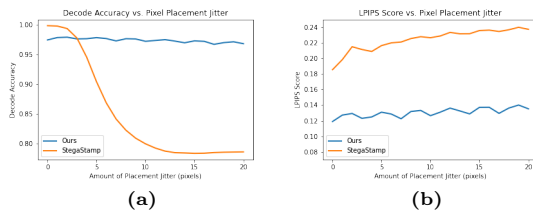
**Real-World** We analyzed a small subset of real-world StegaStamps to verify if the trends found in simulation hold in the real world. Using a sample of 13 cover images from our real-world dataset, we created 12 container images for each background, with 0-11 pixels of placement jitter. We printed the container images, photographed them under good lighting, manually cropped around the StegaStamp, and then sent them through the decoder. In doing this, the average accuracy curve is similar to the one shown in Fig. 6a. Numerically, the average decoding accuracies were 99.85%, 95.23%, 78.77%, and 73.00% under 0, 3, 6, and 9 pixels of misalignment. Note, visually, these StegaStamps were very noticeable due to the white border they require, and the printed patterns had notable differences from the real-world backgrounds.

In summary, although our method takes inspiration from StegaStamp, the problem our method addresses is fundamentally different and more general than the one that StegaStamp addresses. In particular, we develop a pipeline that allows imperceptible stickers to be added to existing surfaces, which requires extra consideration in terms of deployment as it needs to be robust while simple and easy for people to use the system. In addition, unlike StegaStamp, we use a portion of a background image as the cover image, meaning we must ensure that the sticker pattern is optimized so that it is both locally and globally imperceptible (i.e., the border between the sticker region and the non-sticker region should also be as imperceptible as possible). Thus, when a sticker pattern is generated, it does not have to be placed in the exact “correct” location, as there is assumed to be some amount of placement error. This provides the user with flexibility, as they can place a sticker on a portion of a surface, or a whole surface, depending on their needs. Furthermore, StegaStamp requires a white border around container images for localization, resulting in them standing out by necessity, whereas we aim for the container images to blend in as imperceptibly as possible. Nevertheless, even when only accounting for container sections, our method performs better than StegaStamp in terms of imperceptibility, as measured by LPIPS, and it is able to maintain better imperceptibility and accuracy metrics under increasing amounts of placement jitter. Even when no placement jitter is present, our method still does better based on LPIPS and non-perceptual metrics, as presented in Tab. 1.

#### 4.4 Ablation Studies

We experimented with how adding or removing components of the training loss influenced outcomes. See Tab. 2 for numerical results and Fig. 7 for example stickers generated by each version of our model discussed here and in Sec. 4.5.

**Effect of Adding Total Variational Loss** When we introduce total variational loss (TV) during training, we find that the sticker patterns that de-



**Fig. 6:** Average (a) decode accuracy and (b) LPIPS score (lower is better) when misalignment is added when creating container images using StegaStamp and our method.

**Table 1:** For each method, we give the sizes of the full container images and the encoded sections within each image. We use the image and encoding sizes that best match the ratios used by StegaStamp and Jia *et al.* Since we do not have access to the code for Jia *et al.*, their results come from their paper. Also, since it was not stated in their paper, we assume that their calculations use entire container images and not just encoded subsections, as that is the most likely case, given how they were presented.

| Method            | Image Size       | Encoding Size    | PSNR $\uparrow$ | SSIM $\uparrow$ |
|-------------------|------------------|------------------|-----------------|-----------------|
| Ours              | $400 \times 400$ | $400 \times 400$ | <b>32.27</b>    | <b>0.9792</b>   |
| StegaStamp        | $400 \times 400$ | $400 \times 400$ | 28.60           | 0.8857          |
| Ours              | $170 \times 170$ | $64 \times 64$   | <b>40.62</b>    | <b>0.9922</b>   |
| Jia <i>et al.</i> | $256 \times 256$ | $96 \times 96$   | 32.95           | 0.9677          |

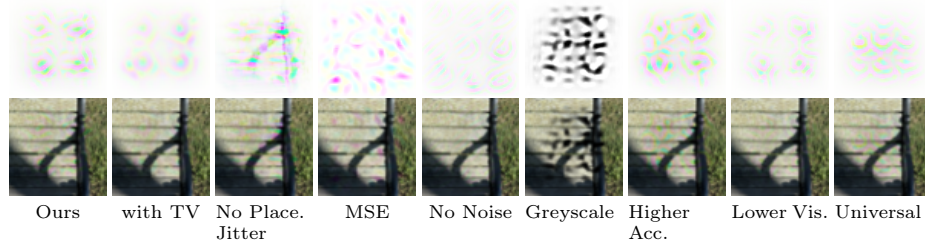
velop have softer boundaries between colors, and overall have more muted and slightly blurrier appearances when compared to sticker patterns from our standard model. In simulation, these stickers have a lower average LPIPS score (0.0428), but also a lower average decoding accuracy, both with (84.20%) and without (96.53%) the augmentations described in Sec. 3.3.

**Effect of Removing Placement Jitter** We also studied the effects of removing simulated human placement error during training. Here, the cover subsection that a sticker is generated for is always the exact subsection where the sticker is placed. We find that the patterns that develop are able to use local features from their cover subsections. The resulting LPIPS score (0.0557) is slightly lower than our standard model’s score. However, the presence of these highly localized patterns makes the stickers more apparent when misaligned with the cover subsections. Using the same test described in Sec. 4.3, we measure how misalignment influences accuracy. We find that decoding accuracy suffers greatly when stickers are misaligned with their cover sections, with decoding averages of 98.74%, 94.51%, and 93.89% after 0, 5, and 10 pixel shifts.

**Effect of MSE instead of LPIPS for Cover Loss** When using simple Mean Squared Error (MSE) for the cover loss instead of the more complicated LPIPS,

**Table 2:** Average decode accuracy, LPIPS score, and detection rate for each version of our model. Decode accuracy and detection rate are both calculated using the test set with augmentations added. LPIPS is calculated by comparing  $64 \times 64$  cover and container subsections. See Fig. 7 for example sticker patterns.

| Method              | Decode Acc. $\uparrow$ | LPIPS $\downarrow$ | Detect. Rate $\uparrow$ |
|---------------------|------------------------|--------------------|-------------------------|
| Ours                | 89.16%                 | 0.0632             | 97.24%                  |
| With TV             | 84.20%                 | 0.0428             | 94.04%                  |
| No Placement Jitter | 86.85%                 | 0.0557             | 96.69%                  |
| MSE                 | 89.11%                 | 0.1111             | 95.76%                  |
| No Noise            | 90.22%                 | 0.0125             | 46.45%                  |
| Greyscale           | 93.79%                 | 0.2132             | 99.81%                  |
| Higher Accuracy     | 94.80%                 | 0.1099             | 98.10%                  |
| Lower Visibility.   | 81.93%                 | 0.0290             | 89.73%                  |
| Universal           | 87.89%                 | 0.0667             | 94.36%                  |



**Fig. 7:** Examples of stickers and their corresponding simulated container images. Each sticker is generated using a different version of our model and a random 100-bit secret. See Tab. 2 for quantitative results.

we observe that the average decoding accuracy and detection rate are similar to our model. However, this model’s LPIPS score (0.1111) is about 1.76 times higher than the LPIPS score of our model. This higher LPIPS score is unsurprising, given the model is not optimized to minimize it. However, the patterns that develop do appear more noticeable because they contain more concentrated areas of color.

**Effect of Removing Noise** When removing noise as a possible augmentation during training, the sticker patterns that develop have 5 times lower LPIPS scores compared to our standard model. However, when noise is added, the detection rate is approximately 50 percentage points lower. In practice, we find that if good lighting can be guaranteed, these stickers work very well in the real world due to their imperceptibility.



## 4.5 Variants

**Greyscale** Given that our encoder outputs color patterns, we look into the outcome of constraining the patterns to be greyscale. The encoder output is changed to be a  $1 \times 64 \times 64$  tensor which is concatenated with itself to create a  $3 \times 64 \times 64$  tensor. After training, we see average decoding accuracies of 98.33% and 93.79%, without and with augmentations added, and an average LPIPS score of 0.2132 between cover and container subsections.

**Higher Accuracy Vs. Lower Visibility** Instead of balancing decoding accuracy and LPIPS during stage three of training, we looked into two variants, where we either placed emphasis on decoding accuracy or LPIPS score. We start training these variants from a checkpoint of our model during stage three, at a point where the localization loss has stabilized, but the cover and secret losses have not. The numerical results suggest that how cover and secret losses are balanced during training leads to decoding accuracies and LPIPS scores that mirror that balance, supporting that there exists a Pareto frontier of models that can be tuned for a desired application.

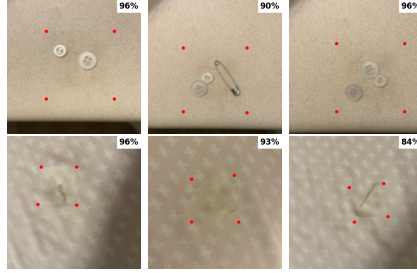
**Universal** Here, we test what happens when stickers are optimized to be universal rather than specialized per location. We do this by generating a sticker for a cover image, and then placing the sticker in a random different image during training. The goal is for a sticker to be universally detectable and decodable, regardless of the background it is placed in.

This model was able to achieve average decoding accuracies of 97.83% and 87.89% on the test set, before and after augmentations were included. It also measured an average LPIPS score of 0.0667 between  $64 \times 64$  cover and container subsections. Additionally, using the same background images and test as described in Sec. 4.3, we measure how misalignment influences the accuracy of these universal stickers. As expected, we find that decoding accuracy does not appear to be influenced by placement jitter. It had decoding averages of 94.57%, 95.49%, and 94.86%, and average LPIPS scores of 0.1481, 0.1586, and 0.1506 after 0, 5, and 10 pixel shifts. The LPIPS scores are calculated using  $400 \times 400$  cover and container subsections.

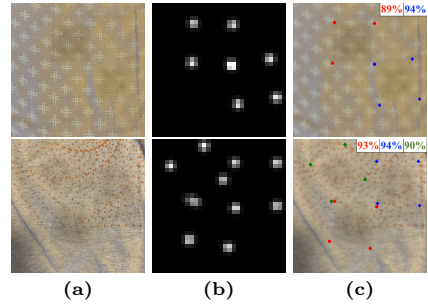
## 4.6 Robustness in the Real World

Even though our method is not optimized to handle occlusion or wrinkles, we found that our localizer and decoder are able to handle some amounts of both. We tested this by constructing additional real-world datasets.

Our occlusion dataset consists of 5 sticker-background pairs, with varying amounts of occlusion added in either the real world or digitally. Out of 32 container images, a sticker was detected in 30, with an average decoding accuracy of 85.17% among those instances.



**Fig. 8:** Real-world examples of occluded stickers (top) and wrinkled stickers (bottom) being localized and decoded.



**Fig. 9:** (a) Container images. (b) Heatmaps returned by our localizer. (c) Results of predicting sticker locations using our auxiliary system and decoding the predicted sticker locations.

Our wrinkles dataset consists of 5 sticker-background pairs with wrinkles added when applying stickers. Out of 49 container images, a sticker was detected in 48, with an average decoding accuracy of 84.60% among those instances.

We found that where occlusion or wrinkles cover a sticker, and the amount a sticker is occluded or wrinkled, heavily impacts decoding accuracy. Additionally, even when decoding accuracy declines, localization is typically still a good approximation. Thus, we conclude that mild occlusions and wrinkles in the stickers can still lead to adequate localization and decoding, while more severe instances can still result in adequate localization. See Fig. 8 and our supplementary material for examples of localization and decoding under these conditions.

#### 4.7 Multi-Sticker Detection

Since our localizer is fully convolutional, we can detect multiple stickers in an image by modifying how we handle the localizer’s output. We developed an auxiliary system, built using HRNet along with blob detection, to process the raw output from the localizer and return all locations of detected stickers. In simulation, we get detection rates of 99.7% and 92.4% when 0 to 4 stickers are randomly placed in an image, before and after augmentations are added. In those cases, the average decoding accuracies are 98.91% and 89.17%, for all predicted sticker locations, before and after augmentations. Note that since we require a small minimum buffer between stickers, there were very few instances where 4 stickers are present. In Fig. 9, we present examples of inputs and outputs when detecting multiple stickers in an image. See our supplementary material for more examples of multi-sticker detection.

## 5 Limitations

While the custom sticker patterns are optimized to have a low perceptual impact on the surfaces they are placed, they are not completely invisible to the human eye. In general, the stickers appear more noticeable in the real world than in simulation due to the physical properties of the sticker paper. While we model real-world color blending in simulation, it does not account for the qualities of the sticker paper itself, and some surfaces' finishes do not blend well with the sticker paper. Also, some backgrounds, as described in Sec. 4.2 can lead to poorly decoded stickers. See our supplementary material for examples of poor sticker-background pairs due to background texture or background pattern. We leave it up to the individual user of our method to pick the sticker paper medium and surfaces that best suit their purposes.

While there are many positive applications of our work, as with most steganographic methods, there are also possibilities to apply it for nefarious purposes. In particular, a potential misuse case could be a user putting encoded stickers on surfaces where the decoded messages or links would not be welcome. However, QR codes already pose this risk and they are meant to be highly visible and are decodable by most smartphone cameras. In comparison, our sticker patterns are not meant to be visible, and thus only decodable by those who know to look for them in a given location.

## 6 Conclusion

The ease and versatility of applying data to the real world through encoded stickers makes our method worthwhile. Depending on the situation, the stickers can be mildly to moderately imperceptible to the human eye. In addition, since our method can detect multiple stickers in an image, additional stickers can be placed to encode longer messages. Furthermore, the stickers are robust to human placement error, making them suitable for real-world deployment in a wide range of scenarios.

## Acknowledgments

The authors would like to thank Ryan P. Adams at Princeton University for his early collaboration and ideas on this work.

## References

1. Ahmadi, M., Norouzi, A., Karimi, N., Samavi, S., Emami, A.: Redmark: Framework for residual diffusion watermarking based on deep networks. *Expert Systems with Applications* **146**, 113157 (12 2020). <https://doi.org/10.1016/j.eswa.2019.113157>
2. Athalye, A., Engstrom, L., Ilyas, A., Kwok, K.: Synthesizing robust adversarial examples. In: *Proceedings of the 35th International Conference on Machine Learning*. vol. 80, pp. 284–293 (2018)
3. Baluja, S.: Hiding images in plain sight: Deep steganography. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*. vol. 30. Curran Associates, Inc. (2017), [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/838e8afb1ca34354ac209f53d90c3a43-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/838e8afb1ca34354ac209f53d90c3a43-Paper.pdf), accessed: 2026-06-25
4. Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch (2018)
5. Bui, T., Agarwal, S., Yu, N., Collomosse, J.: Rosteals: Robust steganography using autoencoder latent space. In: *Proc. CVPR WMF* (2023)
6. Chu, H.K., Chang, C.S., Lee, R.R., Mitra, N.J.: Halftone qr codes. *ACM Trans. Graph.* **32**(6) (nov 2013), <https://doi.org/10.1145/2508363.2508408>
7. Feick, M., Tang, X., Garcia-Martin, R., Luchianov, A., Huang, R.W.X., Xiao, C., Siu, A., Dogan, M.D.: Imprinto: Enhancing infrared inkjet watermarking for human and machine perception. In: *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, New York, NY, USA (2025)
8. Garateguy, G.J., Arce, G.R., Lau, D.L., Villarreal, O.P.: Qr images: Optimized image embedding in qr codes. *IEEE Transactions on Image Processing* **23**(7), 2842–2853 (2014). <https://doi.org/10.1109/TIP.2014.2321501>
9. Guan, Z., Jing, J., Deng, X., Xu, M., Jiang, L., Zhang, Z., Li, Y.: Deepmih: Deep invertible network for multiple image hiding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp. 1–1 (2022). <https://doi.org/10.1109/TPAMI.2022.3141725>
10. Hayes, J., Danezis, G.: Generating steganographic images via adversarial training. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*. vol. 30. Curran Associates, Inc. (2017), [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/fe2d010308a6b3799a3d9c728ee74244-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/fe2d010308a6b3799a3d9c728ee74244-Paper.pdf), accessed: 2026-06-25
11. Huiskes, M.J., Lew, M.S.: The mir flickr retrieval evaluation. In: *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*. p. 39–43. MIR '08, Association for Computing Machinery, New York, NY, USA (2008), <https://doi.org/10.1145/1460096.1460104>
12. Jia, J., Gao, Z., Chen, K., Hu, M., Min, X., Zhai, G., Yang, X.: Rihoop: Robust invisible hyperlinks in offline and online photographs. *IEEE Transactions on Cybernetics* **52**(7), 7094–7106 (2022). <https://doi.org/10.1109/TCYB.2020.3037208>
13. Jia, J., Gao, Z., Zhu, D., Min, X., Zhai, G., Yang, X.: Learning invisible markers for hidden codes in offline-to-online photography. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 2263–2272 (2022). <https://doi.org/10.1109/CVPR52688.2022.00231>
14. Lee, J.E., Seo, Y.H., Kim, D.W.: Convolutional neural network-based digital image watermarking adaptive to the resolution of image and watermark. *Applied Sciences* **10**(19) (2020), <https://www.mdpi.com/2076-3417/10/19/6854>, accessed: 2026-06-25

15. Li, J., Schmidt, F., Kolter, Z.: Adversarial camera stickers: A physical camera-based attack on deep learning systems. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 3896–3904. PMLR (09–15 Jun 2019), <https://proceedings.mlr.press/v97/li19j.html>, accessed: 2026-06-25
16. Lin, S.S., Hu, M.C., Lee, C.H., Lee, T.Y.: Efficient qr code beautification with high quality visual content. *IEEE Transactions on Multimedia* **17**(9), 1515–1524 (2015). <https://doi.org/10.1109/TMM.2015.2437711>
17. Liu, G., Si, Y., Qian, Z., Zhang, X., Li, S., Peng, W.: Wrap: Watermarking approach robust against film-coating upon printed photographs. In: Proceedings of the 31st ACM International Conference on Multimedia. p. 7274–7282. MM '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3581783.3612143>
18. Lu, S.P., Wang, R., Zhong, T., Rosin, P.L.: Large-capacity image steganography based on invertible neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10811–10820 (2021). <https://doi.org/10.1109/CVPR46437.2021.01067>
19. Luo, X., Zhan, R., Chang, H., Yang, F., Milanfar, P.: Distortion agnostic deep watermarking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
20. Ma, Z., Yang, X., Fang, H., Zhang, W., Yu, N.: Oacode: Overall aesthetic 2d barcode on screen. *IEEE Transactions on Multimedia* **25**, 8687–8698 (2023). <https://doi.org/10.1109/tmm.2023.3239755>, <http://dx.doi.org/10.1109/TMM.2023.3239755>
21. Qin, C., Li, X., Zhang, Z., Li, F., Zhang, X., Feng, G.: Print-camera resistant image watermarking with deep noise simulation and constrained learning. *IEEE Transactions on Multimedia* **26**, 2164–2177 (2024). <https://doi.org/10.1109/TMM.2023.3293272>
22. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing (2015)
23. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 1528–1540. CCS '16, Association for Computing Machinery, New York, NY, USA (2016), <https://doi.org/10.1145/2976749.2978392>
24. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: CVPR (2019)
25. Tancik, M., Mildenhall, B., Ng, R.: Stegastamp: Invisible hyperlinks in physical photographs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2117–2126 (2020)
26. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence* **43**(10), 3349–3364 (2020). <https://doi.org/10.1109/TPAMI.2020.2983686>
27. Wang, Z., Byrnes, O., Wang, H., Sun, R., Ma, C., Chen, H., Wu, Q., Xue, M.: Data hiding with deep learning: A survey unifying digital watermarking and steganography. *IEEE Transactions on Computational Social Systems* **PP**, 1–15 (12 2023). <https://doi.org/10.1109/TCSS.2023.3268950>

28. Wei, X., Guo, Y., Yu, J.: Adversarial sticker: A stealthy attack method in the physical world. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022)
29. Wengrowski, E., Dana, K.: Light field messaging with deep photographic steganography. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019)
30. Wu, P., Yang, Y., Li, X.: Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet* **10**, 54 (06 2018). <https://doi.org/10.3390/fi10060054>
31. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: *European Conference on Computer Vision (ECCV)* (2018)
32. Xu, M., Li, Q., Niu, J., Su, H., Liu, X., Xu, W., Lv, P., Zhou, B., Yang, Y.: Art-up: A novel method for generating scanning-robust aesthetic qr codes. *ACM Trans. Multimedia Comput. Commun. Appl.* **17**(1) (apr 2021). <https://doi.org/10.1145/3418214>, <https://doi.org/10.1145/3418214>
33. Xu, M., Su, H., Li, Y., Li, X., Liao, J., Niu, J., Lv, P., Zhou, B.: Stylized aesthetic qr code. *IEEE Transactions on Multimedia* **21**(8), 1960–1970 (2019). <https://doi.org/10.1109/TMM.2019.2891420>
34. Ye, H., Zhang, S., Jiang, S., Liao, J., Gu, S., Zheng, D., Wang, C., Li, C.: Robust message embedding via attention flow-based steganography. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 12840–12849 (June 2025)
35. Zhang, K.A., Cuesta-Infante, A., Xu, L., Veeramachaneni, K.: Steganogan: High capacity image steganography with gans. *ArXiv abs/1901.03892* (2019), <https://api.semanticscholar.org/CorpusID:59413797>, accessed: 2026-06-25
36. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 586–595 (2018). <https://doi.org/10.1109/CVPR.2018.00068>
37. Zhang, Y., Deng, S., Liu, Z., Wang, Y.: Aesthetic qr codes based on two-stage image blending. In: He, X., Luo, S., Tao, D., Xu, C., Yang, J., Hasan, M.A. (eds.) *MultiMedia Modeling*. pp. 183–194. Springer International Publishing, Cham (2015)
38. Zhong, X., Huang, P.C., Mastorakis, S., Shih, F.Y.: An automated and robust image watermarking scheme based on deep neural networks. *IEEE Transactions on Multimedia* **23**, 1951–1961 (2021). <https://doi.org/10.1109/TMM.2020.3006415>
39. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017)
40. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision* **127**(3), 302–321 (2019)
41. Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L.: Hidden: Hiding data with deep networks. In: *European Conference on Computer Vision* (2018), <https://api.semanticscholar.org/CorpusID:50784854>, accessed: 2026-06-25