Guided Image Inpainting: Replacing an Image Region by Pulling Content from Another Image

Yinan Zhao[*], Brian Price[+], Scott Cohen[+], Danna Gurari[*]
[*] University of Texas at Austin, [+] Adobe Research

Abstract

Deep generative models have shown success in automatically synthesizing missing image regions using surrounding context. However, users cannot directly decide what content to synthesize with such approaches.We propose an end-to-end network for image inpainting that uses a different image to guide the synthesis of new content to fill the hole. A key challenge addressed by our approach is synthesizing new content in regions where the guidance image and the context of the original image are inconsistent. We conduct four studies that demonstrate our method yields more realistic image inpainting results over seven baselines.

1. Introduction

People often wish to replace undesired content from their photos with a plausibly realistic alternative. For example, a visitor spends a long time trying to capture a picture of the Leaning Tower of Pisa but a passerby occludes the tower in the captured photo. It is often desirable to remove the passerby from the photo. In another case, a digital artist may wish to show Denver with a beach by replacing part of the photo with a beach from San Diego. These tasks involve removal of undesired regions followed by automated image inpainting (also called image completion and hole filling), the task of filling in the lost part of an image. Automated image inpainting is an established computer vision problem with many valuable applications [16, 37, 6, 9, 38, 27, 45].

Numerous image inpainting solutions have been proposed. One line of work involves cutting and pasting a semantically similar patch and then blending it with the image [16, 37]. Unfortunately, such methods typically generate unrealistic results when the pasted content differs from the context of the image; e.g., as exemplified in Figure 1 (see yellow bounding boxes on images). Another set of methods automatically synthesize missing image regions using surrounding context [11, 10, 24, 6, 9, 38, 28, 40]. With such approaches, users may receive realistic looking results, however a user cannot control what content is synthesized in the hole. An open challenge is how to produce a realistic looking image inpainting result while enabling the user to control what content fills the hole.

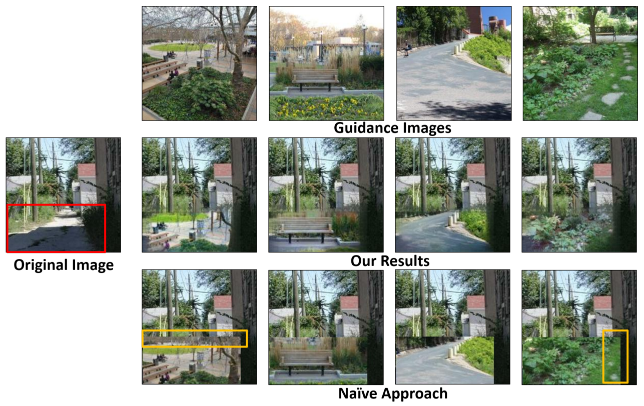

Figure 1. We propose a method for automatically filling a hole in an image. It takes an original image with a hole (e.g., the red bounding box of the road in the original image) paired with a guidance image and outputs an image inpainting result. As shown, different guidance images (row 1) lead to diverse image inpainting results (row 2). We also show results from a naive approach which cuts and pastes the guidance image content into the hole as is (row 3). As observed, a key challenge is synthesizing new content for the inconsistent regions in the transition between the original image and the guidance image (see yellow bounding boxes).

We propose an image inpainting method to overcome the limitations discussed above. Our key idea is to use another image to "guide" the synthesis process within a deep learning framework. This empowers a user to control the result, as illustrated in Figure 1. Given an original image and a region to remove, we show different "guidance images" (top row) and results from our method (middle row). The key difficulty is successfully evolving the content of a guidance image to match the context of the original image, as illustrated by the results of a naive cut-and-paste method (bottom row). The transition between the original image and pasted content often has inconsistent regions (shown in yellow bounding boxes on images). Our main contribution is an end-to-end neural architecture that, given a guidance image, leverages the guidance image to fill the hole while identifying inconsistent regions and then synthesizing new content to match those regions with the image context.

We explore three questions when evaluating our proposed inpainting approach: 1) how well does our approach recover true pixel values in the hole?, 2) how often does our approach synthesize images that people think are real? and 3) how often do people perceive our results as more realistic than alternative methods? Our experiments demonstrate our approach is effective in capturing true pixel values with an image restoration task and show humans perceive that our approach synthesizes more realistic hole-fillings than seven alternative approaches.

2. Related Work

Image Inpainting: Existing inpainting methods fall into two groups.

One group relies only on image context to fill holes. Classical approaches use diffusion to propagate isophotes from the surrounding context to the hole [3, 27] or apply texture synthesis to extend texture from surrounding regions to the holes [11, 10, 2, 24, 6, 9, 38]. However, such methods cannot capture the semantics of the image. More recently, deep neural networks were introduced to regress from surrounding context to missing pixel values [28, 41, 18, 40, 42] or model the distribution of possible outputs by random sampling [44]. However, these approaches do not give users the control on what should be synthesized.

The other group relies on other images to fill the hole. Scene completion involves finding a semantically similar patch from a database of millions of images and then cutting, pasting, and blending it with the original image [16]. Internet-based inpainting improves the semantic similarity in nearest neighbor search by using a web engine [37, 45]. While such approaches often can transfer structure and high-frequency details from the guidance image to the hole, results often have inconsistent regions where the pasted/blended content fails to match the context. Unlike these methods, our approach can synthesize new content to resolve such inconsistencies.

Image Harmonization: Image harmonization methods adjust low-level appearances of foreground and background regions to make them compatible when generating realistic composite images [34, 35, 30, 31, 29, 36]. For example, blending transfers content from the guidance image to the source image and interpolates the error in the transition region to address color inconsistencies [29]. Image melding uses patch-based synthesis to resolve texture inconsistencies when transferring content from the guidance image to the source image [7]. Unlike prior work, our approach synthesizes new consistent content in the transition region using an end-to-end deep neural network. Experiments demonstrate the advantage of our system over relying on existing top-performing harmonization methods [29, 36, 7].

Style Transfer: Image style transfer combines the high-level content of one image with the low-level style of another [13, 21, 14, 12]. While our approach similarly combines information from two images, our approach differs from style transfer because it inserts high-level features of the guidance patch directly into the high-level feature map of the incomplete image to synthesize a consistent image. Rather than try to transfer low-level style from one image to another, our aim is to retain the high-level content of the guidance image when synthesizing content.

3. Methods

Our inpainting approach is embedded in a larger system. For completeness, we provide a system overview in Section 3.1. We then describe the key novel component of our system in detail in Sections 3.2 and 3.3.

3.1. System Overview

Our system consists of three steps. Given an image, the first step is identifying a region of the image to remove. Next, a guidance image is chosen, either manually from a user or automatically from an intelligent image retrieval system. Our implementations for these two steps are described in the experimental section (Section 4.2). The final step is creating an image inpainting result, given an *incomplete image* (i.e., original image with the user-defined region removed) and *guidance image*. This final step is where the key novelty of our work lies, and we describe our implementation for this step in the remainder of this section.

3.2. Image Inpainting Framework

An overview of our deep learning framework is shown in Figure 2. It takes as input an *incomplete image* and *guidance image*, and applies two modules sequentially to generate an image inpainting result. The first module, which we call "Localization Network", identifies a patch in the guidance image to use to replace the hole in the incomplete image. The second module, which we call "Synthesis Network", synthesizes new content to fit within the image context, informed by the guidance image patch. We describe here the design of this deep learning framework and discuss in the next section how we train it (Section 3.3).

3.2.1 Localization Network

This network is a spatial transformer [20] designed to identify an image patch from the guidance image to use to fill the hole in the incomplete image. It consists of two modules. The first module is a CNN, derived from the VGG architecture [33], that takes as input an incomplete image, guidance image, and binary mask indicating the hole[1] and outputs the affine transformation parameters that will be used to transform the guidance image. The CNN contains

[1]Each input image is of size 224×224.

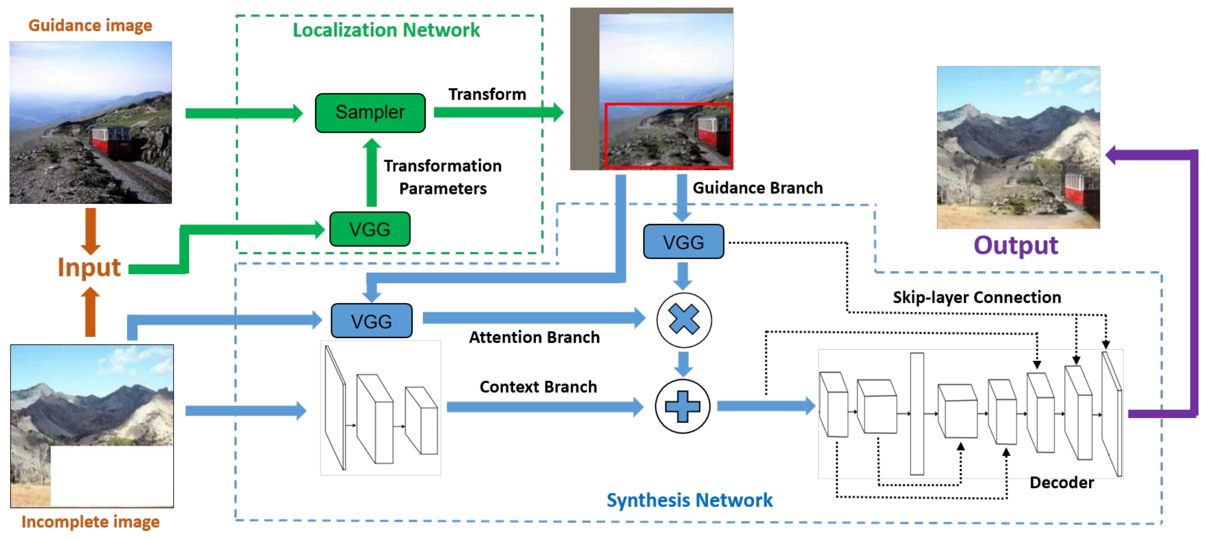

Figure 2. Our proposed inpainting framework includes a 1) localization network which locates the best fitting patch in the guidance image and aligns it with the hole and 2) synthesis network which encodes both the incomplete image and the aligned guidance image to synthesize a reasonable patch to fill the hole. (Best viewed in color.)

13 convolution layers for computing a spatial feature representation of $7 \times 7 \times 512$ dimensionality, followed by three fully connected layers to predict the six parameters of affine transformation. The second module is a differentiable image sampler that transforms the guidance image based on the predicted transformation parameters. It samples the input using a sampling grid to transform the guidance image. Once completed, the patch at the corresponding location in the transformed guidance image is used as the source for filling the hole. In other words, the guidance patch that informs the image synthesis process is aligned with the hole in the incomplete image (e.g., see red box in Figure 2).

3.2.2 Synthesis Network

We design a synthesis network that automatically decides which high-level features from the guidance patch to use, locates inconsistencies between the guidance patch and incomplete image, and synthesizes new content to clean-up the inconsistencies. Figure 2 provides an overview of the network architecture. As shown, we use an encoder-decoder CNN. The encoder combines information from three branches that encode the (1) incomplete image (context branch), (2) aligned guidance image (guidance branch), and (3) inconsistent regions between the guidance patch and context of the incomplete image (attention branch). The decoder converts the encoded feature representation into a hole filling. Skip-layer connections [32] are used to preserve details that may be lost due to the bottleneck (i.e., small number of neurons) in the encoder-decoder pipeline. We describe below the technical details of this network.

Attention Branch: This branch computes a spatial attention map that indicates to what extent each pixel in the aligned guidance image is consistent with the surrounding context of the incomplete image. It takes as input a concatenation of the incomplete image, aligned guidance image, and binary mask indicating the hole. Its architecture is a fully convolutional network, following the *VGG* architecture until *conv3_3*, with an additional convolutional layer and then finally a sigmoid layer that outputs 1-channel feature map. This branch controls to what extent to preserve the original content of the guidance patch when synthesizing new content. Note that even in areas where the network is directed to use the original content, it still often must adjust the patch appearance to match the surrounding context.

Context Branch: This branch compresses the incomplete image concatenated with the binary mask indicating the hole into a high level feature representation. Its architecture is a fully convolutional network, following the *VGG* architecture until *conv3_3*, and so includes six convolutional layers. It outputs a $56 \times 56 \times 256$ dimensional feature.

Guidance Branch: This branch compresses the aligned guidance image into a high level feature representation. We choose a pretrained *VGG* on *ImageNet* until *conv3_3* to encode the high-level semantic features for the visual recognition task. The output feature dimension matches that of the context branch.

Decoder: This branch takes as input the result from multiplying the guidance branch with the attention branch and then adding that result to the context branch. Consequently, the input highlights which regions from the guidance patch to preserve versus synthesize when fusing it with the context of the incomplete image. The decoder architecture uses the *conv3_3* to *fc7* layer from the *VGG* architecture to extract a 4096 feature dimension. This compact representation then goes through a series of bilinear upsampling followed by convolution to generate an image of the original image size. Finally, a *tanh* layer is used to constrain the output image that shows a realistic hole filling to a normalized range.

Skip-layer Connections: Our motivation for this component is to retain details that otherwise would be lost because of the bottleneck (i.e., small number of neurons) in the encoder-decoder pipeline. We introduce a skip-layer connection [32] that concatenates the *conv1_2* and *conv2_2* feature maps in the guidance branch with the corresponding symmetric feature maps in the decoder and skip-layer connections within the decoder. These connections at different layers provide different levels of abstraction.

3.3. Image Inpainting Training

A key challenge for our proposed guided image inpainting problem is there is no clear way to generate the ground truth needed to train a solution. Inspired by prior work that shows models trained on synthetic data can generalize well to real images [4, 39], we propose a process to create synthetic training data for our problem. The idea of our approach is to use a corrupted patch of the original image as the guidance image in order to have the ground truth needed for training. If a method worked perfectly, then given the incomplete original image and corrupted patch of the original image, it would perfectly reconstruct the original image. We will show in our experiments that our system generalizes well in real scenarios when the guidance image is a real, different image from the original image (Sections 4.2 and 4.3), despite training with corrupted patches of the original images as the guidance images. We describe below our process for creating synthetic training data and the loss functions for training the localization and synthesis networks.

3.3.1 Training Data Generation

We create each training example by randomly cropping out a patch on the original image, corrupting the patch (as described below), adding irrelevant content to its boundary (as described below), and then pasting the enlarged corrupted patch back into the original image for use as the guidance image. This four-step process is illustrated in Figure 3. Note that the hole in the original image needing to be filled matches the size of the enlarged patch. This process ensures that we both have the ground truth inpainting result and we have inconsistent regions between the guidance patch and incomplete image (i.e., irrelevant content) that the synthesis network will need to modify to match the ground truth. Also note that the location of the corrupted patch in the guidance image is not fixed. This ensures the localization network will need to be able to find the location of the guidance patch.

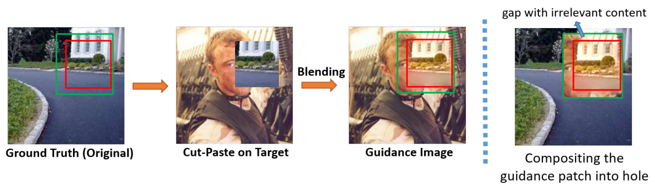
gap with irrelevant content
Ground Truth (Original)
Cut-Paste on Target
Blending
Guidance Image
Compositing the guidance patch into hole

Figure 3. Shown is an example illustrating our process for generating synthetic training data. After identifying the hole to remove from the original image (green bounding box) and the patch of the original image to corrupt (red bounding box), our process creates an image containing both corrupted original content (inside red bounding box) and content irrelevant to the original image (gap between red and green bounding boxes).

To corrupt the patch that is cropped from the original image, we first insert it into a random location in another random image (which we call the target image), as shown in step 2 in Figure 3. Then, we apply Poisson Blending [29] to corrupt its appearance, as shown in step 3 in Figure 3 (see red rectangle).

To add irrelevant content to the corrupted patch, we add content from the target image, as illustrated by the region between the red and green rectangles in Figure 3. We use different gap widths between the patch (red rectangle) and hole size (green rectangle) so that the synthesis network cannot inadvertently learn biases for where to transfer existing content in the guidance patch and where to synthesize new content. Specficially, the corrupted patch and hole sizes are randomly sampled in the ranges $[80 \times 80, 96 \times 96]$ and $[96 \times 96, 128 \times 128]$ respectively.

Given a large synthetic dataset created as described above, the expectation is that our method trained in a supervised manner will learn to find the guidance patch using the localization network and synthesize the content of the original patch of the image using the synthesis network.

3.3.2 Image Synthesis Loss

We describe here the loss function we use to train our synthesis network. The choice of loss function is important because the problem is underconstrained. Recent work [21, 8, 25] has shown perceptual loss computed on feature representations of a visual perception network can be used to generate high-quality images. Generative Adversarial Networks (GANs) [15] are shown to be successful in image generation as an adaptive loss [28, 19, 25]. Consequently, we combine a perceptual loss computed on a visual perception network ϕ together with an adversarial loss to train the synthesis network:

$$l_g(G) = \frac{1}{N} \sum_{(\tilde{I}, I_{guide})} l^{\phi}_{perc}(I, I_G) + \lambda_{adv} l_{adv}(I_G)$$

$$I_G = G(\tilde{I}, I_{guide})$$

The number of training pairs is N. We use I to denote the ground truth and $\tilde{I}$ for the corresponding incomplete image with the user defined region removed. I_{guide} is a guidance image. The perceptual loss guides the image synthesis to match the ground truth, while the adversarial loss encourages the network to favor image generation that resides on the natural image manifold.

Perceptual Loss: We compute the perceptual loss on multiple layers of a visual perception network ϕ. Our intention is that matching both high-level and low-level representations guides the synthesis network to learn both global structure and fine-grained details [5]. Let $\phi_j(x)$ be the feature map of the jth layer of a pretrained network ϕ given input image x. $\phi_j(x)$ has shape of $C_j \times H_j \times W_j$. Suppose the completed image by our network is $\hat{I} = G(\tilde{I}, I_{guide})$. The perceptual loss is defined as follows:

$$l^{\phi}_{perc}(I, \hat{I}) = \sum_j \frac{\lambda_j}{C_j \times H_j \times W_j} ||\phi_j(I) - \phi_j(\hat{I})||_2^2$$

The hyperparameters λ_j balance the contribution of each layer j to the perceptual loss. For layers ϕ_j we use *conv1_2*, *conv2_2*, *conv3_3*, *conv4_3*, *conv5_3* in *VGG-16*. λ_j are updated to normalize the expected contribution of each layer in a certain number of iterations.

Adversarial Loss: Following [15] we train a discriminator network D to distinguish synthesized images from natural images. We alternately optimize D and a generator G to solve the following min-max problem:

$$\min_G \max_D {}_{I \sim p_{data}(I)}[\log D(I)] +_{\hat{I} \sim p_G(\hat{I})} [\log(1 - D(\hat{I}))]$$

With this formulation, the generator G is trained to fool the differentiable discriminator D, creating solutions highly similar to natural images.

The discriminator D, following [18], has two critics: a global one and a local one. The global critics takes as input the whole completed image to judge global consistency of a scene while the local critics only takes a patch centered around the completed region to encourage local consistency with the surrounding area. When training the generator, the adversarial loss $l_{adv}(\hat{I})$ is defined based on the real/fake probability of the discriminator D as:

$$l_{adv}(\hat{I}) = -\log D(\hat{I})$$

3.3.3 Localization Loss

We now describe the loss function we use to train our localization network. Although the spatial transformer has a differentiable image sampler, the function of image synthesis with respect to the transformation parameters does not have a good shape that is easy to be optimized. To make the network trainable, we add a loss directly on the predicted transformation parameters (we call it localization loss).

Let M be the number of points sampled in the guidance image, $x_i(I)$ be the vector representing the homogeneous coordinate of the i-th sampled point, $T(I)$ be the ground truth affine transformation matrix for image I (known when creating the dataset), and $\bar{T}(I)$ be the estimated transformation matrix. We define the localization loss l_{loc} as the average of Euclidean distances between estimated point positions after transformation and their corresponding ground truth locations:

$$l_{loc}(\bar{T}) = \frac{1}{MN} \sum_I \sum_{i=1}^{M} ||\bar{T}(I)x_i(I) - T(I)x_i(I)||_2^2$$

We encourage the estimated transformation matrix to be close to the ground truth transformation by minimizing the localization loss.

3.3.4 Implementation

We create the synthetic dataset from ADE20K [43]. There are $20,210$ images in the training set of ADE20K and 2000 in the validation set. For each image in the training set, we generate 50 pairs of incomplete image and guidance image, resulting in $1,010,500$ pairs for training. The localization network and synthesis network are trained separately with localization loss and synthesis loss. Training them jointly is undesirable since this leads the synthesis network to blur the image generation to accommodate localization errors.

For training CNNs, we use Tensorflow [1] framework. We choose Adam [23] as the optimization method. We fix the parameters of Adam as $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The initial learning rate in Adam is set to $l_r = 0.0002$.

We use a weighted sum of perceptual loss and adversarial loss to train the generator. The weight of adversarial loss is $\lambda_{adv} = 2$. $M = 8$ points are sampled in each image to compute the localization loss. We train the synthesis network for $1,918,000$ iterations with batch size equal to 2. It takes roughly two weeks to train on NVIDIA GeForce GTX 1080 GPU. The localization network is trained for $841,000$ iterations, which takes an extra 5 days.

4. Evaluation

We now describe our studies to evaluate the power of our localization and synthesis network in localizing fitting patches and synthesizing realistic images. We address the following research questions:

- How well does our model capture true pixel values in the missing region?
- How often does our method synthesize realistic images that humans cannot distinguish from natural images?
- How often is our synthesis preferred to baselines?

We conduct four experiments to address these questions, evaluating synthesis and localization separately. In Section 4.1, we measure how well our synthesis network restores true pixel values of the original patch given a corrupted one as guidance. Section 4.2 demonstrates the effectiveness of our synthesis network to synthesize images that people mistakenly perceive as real. In Section 4.3, we conduct pairwise comparisons between our synthesis and baselines to measure how often our results are preferred to alternative methods. In Section 4.4, we perform pairwise comparisons between our localization and Local Context Matching [16] to investigate which localization method results in more realistic synthesis.

4.1. Image Restoration

Baselines: We compare our synthesis network to seven related baselines across five types of methods. We use the code provided by the authors.

- *Texture synthesis*: Content-Aware-Fill (CAF) using PatchMatch [2].
- *Deep Generative Models*: We evaluate three methods for this type of approach. We use Context Encoder (CE) [28], where we retrain it with *l2* and adversarial loss on ADE20K [43]. We also use High-Resolution Image Inpainting (HR) [40] and Globally and Locally Consistent Image Completion (GLCIC) [18].
- *Harmonization*: Deep Harmonization (DH) [36].
- *Blending*: Poisson Blending (PB) [29].
- *Melding*: Image Melding (IM) [7].

CAF, CE, HR and GLCIC hallucinate the missing region from the context without guidance. DH, PB, IM and our method use a corrupted patch from the original image as guidance. The latter four approaches have access to additional guidance images that are not available to the former four approaches. We compare to both classes of algorithm for a comprehensive evaluation.

Dataset: We evaluate on our synthetic test set, consisting of 2000 images with random holes derived from ADE20K validation set. The guidance patch is corrupted from the original patch, with a gap filled with unrelated content near the boundary. To evaluate the synthesis network exclusively, we assume ground truth localization, aligning the guidance patch with the hole perfectly.

Evaluation Method: We evaluate the restoration by measuring pixelwise distance between the prediction and ground truth using three metrics: L1 Loss, L2 Loss and PSNR (Peak Signal to Noise Ratio).

Results: Results are shown in Table 1. As observed, our synthesis network consistently offers significant gains over baselines for all three metrics. Our synthesis method decreases the error by **5.34%** and **1.93%** in terms of Mean L1 Loss and Mean L2 Loss compared to the best baseline, achieving **6.89%** and **1.11%** respectively. We also outperform the best baseline by **4.13dB** in PSNR, achieving **20.68dB**. Interestingly, our results show that methods that hallucinate results without guidance (CE, HR, GLCIC) perform similarly to those that have guidance (PB, IM). This highlights both the effectiveness of the methods and the difficulty in properly leveraging the information in a corrupted patch. The results reveal our method is able to effectively utilize this corrupted information. We attribute the success in this task to the nature of our approach: it can transfer the structure of the guidance image with color and appearance adjusted and synthesize appropriate new content when necessary (fill the gap with irrelevant content in this case).

Method	Mean L1	Mean L2	PSNR
CAF[2]	15.43%	5.09%	14.38dB
CE[28]	12.91%	3.21%	15.91dB
HR[40]	13.05%	3.29%	15.83dB
GLCIC[18]	13.28%	3.47%	15.56dB
PB[29]	13.63%	3.28%	15.41dB
IM[7]	12.23%	3.04%	16.55dB
DH[36]	18.87%	6.02%	12.73dB
Ours	**6.89%**	**1.11%**	**20.68dB**

Table 1. Results of our method and seven baselines with respect to three metrics for the image restoration task. Smaller Mean L1 values, smaller Mean L2, and larger PSNR values indicate better performance. (% in the table is included to facilitate reading).

4.2. Absolute Realism

In this section, we show people a random image from a mixed dataset of real and synthesized images and ask them to guess whether the shown image is real or fake. We balance the number of real and fake images to avoid a strong real/fake prior. We investigate using a guidance image that is a different real image from the original image to address the common use case scenario of replacing content in an image with content from another image.

To recruit people for our human perception task, we employ crowd workers on Amazon Mechanical Turk (AMT). We show each image for four seconds to give people sufficient time to determine its realism. Our choice of this time period is motivated in part by the work of Joubert et al. [22], which showed humans can understand scenes and complete simple tasks such as categorizing man-made and natural scenes with high accuracy (both around 96%) in around 390ms.

Implementation: We implement the components in Section 3.1 as follows.

We use a rectangular hole to remove content from the original image, since bounding boxes are an efficient, commonly-used approach to annotate images. We simulate a user who wants to remove an object or part of a scene by using a ground truth semantic segmentation to select the region to remove and fitting a bounding box around it.

Method	NI	CE[28]	HR[40]	CAF[2]	PB[29]	DH[36]	GLCIC[18]	IM[7]	Ours
Retrieval (a)	97.7%	10.0%	14.0%	31.0%	18.0%	23.0%	14.0%	23.0%	**33.0%**
Retrieval (b)	97.7%	22.0%	15.0%	16.0%	20.0%	22.0%	12.0%	27.0%	**36.0%**

Table 2. User study results showing the perceived absolute realism of synthesized images. The numbers in the table are the percentage of images deemed to be real. NI refers to natural images. Retrieval (a) uses the full original image to retrieve the guidance image while Retrieval (b) uses the incomplete image with its hole filled by CAF [2] to retrive the guidance image from a large database.

Method	HR[40]	PB[29]	DH[36]	CAF[2]	CE[28]	IM[7]	GLCIC[18]
Retrieval (a)	76%	76%	71%	70%	70%	67%	66%
Retrieval (b)	71%	73%	72%	70%	73%	67%	70%

Table 3. Shown are results from our relative realism study. Each cell lists the percentage of pairs in which synthesized images by our approach are rated more realistic than the corresponding baseline in human perception experiments. Chance is at 50% result.

For a guidance image, we are motivated to efficiently find an image that is semantically similar to the original image. We follow the work of Heo et al. [17] to retrieve nearest neighbors in a large dataset. We examine two retrieval scenarios: using (a) the full original image or (b) the incomplete image with its hole filled by *Content-Aware-Fill* [2]. We will report results for Retrieval (a) and (b) separately. In Retrieval (a), we assume access to the full image because the hole often comes from replacing part of the original image. In Retrieval (b) we remove the original content in the hole to prevent it from biasing the retrieval.

Baselines: We use the seven baselines used in Section 4.1.

Dataset: We use images in ADE20K validation set as original images and retrieve guidance images in MSCOCO [26]. To evaluate synthesis separately, we use Local Context Matching (LCM) [16] as localization for all the synthesis methods. LCM [16] minimizes pixelwise SSD error in the local context. We randomly sample 100 images to go through our synthesis network and all the baselines. Inpainting results from all the methods are mixed together with 300 real images from ADE20K validation set to balance the number of real and fake images.

Evaluation Method: We measure absolute realism of the synthesized images by the fraction of images deemed to be real in the AMT study.

Results: As shown in Table 2, although still far from natural image quality, **33%** and **36%** of the synthesized images by our approach are deemed to be real with Retrieval (a) and (b) respectively. It outperforms all baselines significantly except for only slightly beating *Content-Aware-Fill(CAF)* in Retrieval (a). We will show in Section 4.3 that our synthesized images are rated to be more realistic than *Content-Aware-Fill* in a more careful pairwise comparison. Methods that hallucinate results without guidance (CE, HR, GLCIC) do not perform well in this experiment. One of the reasons is that the holes are generally large and it is challenging for these methods to hallucinate a large region. Despite our synthesis network being trained on a synthetic dataset, our results show it can generalize well to real images.

4.3. Relative Realism

We next investigate how often the generated images by our synthesis network are considered to be more realistic than those created by baselines. We conduct pairwise comparisons between our method and all the baselines in the aforementioned absolute realism study. Each pair contains two images synthesized by our synthesis network and a baseline for the same incomplete image using the same guidance image. The users are asked to select the more realistic image after they are shown a pair side-by-side for an unlimited time.

Implementation: We use the same implementation as in Section 4.2.

Baselines: We use the seven baselines used in Section 4.2.

Dataset: We use the same dataset as in Section 4.2.

Evaluation Method: We compute the relative realism by the percentage of pairs in which our synthesized images are considered to be more realistic by people (i.e., AMT crowd workers) than the baselines.

Results: As shown in Table 3, people rate our synthesized images more realistic than baselines for at least **66%** of test images. For example, our results are more realistic than *Content-Aware-Fill*, which is the best baseline in the absolute realism experiment for **70%** of images (using Retrieval method (a) to find a guidance image). These results reveal greater improvements of our synthesis over baselines than is evident from the absolute realism study. Our results are similar whether using Retrieval method (a) or (b) to choose a guidance image.

We show qualitative results from our results and the baselines in Figure 4. As observed, our synthesis network generates more realistic hole-fillings that transfer both high-level structure and necessary fine-grained details from the guidance image, while maintaining consistency with the context of the original image. Our approach can also synthesize diverse inpainting results, as shown in Figure 1.

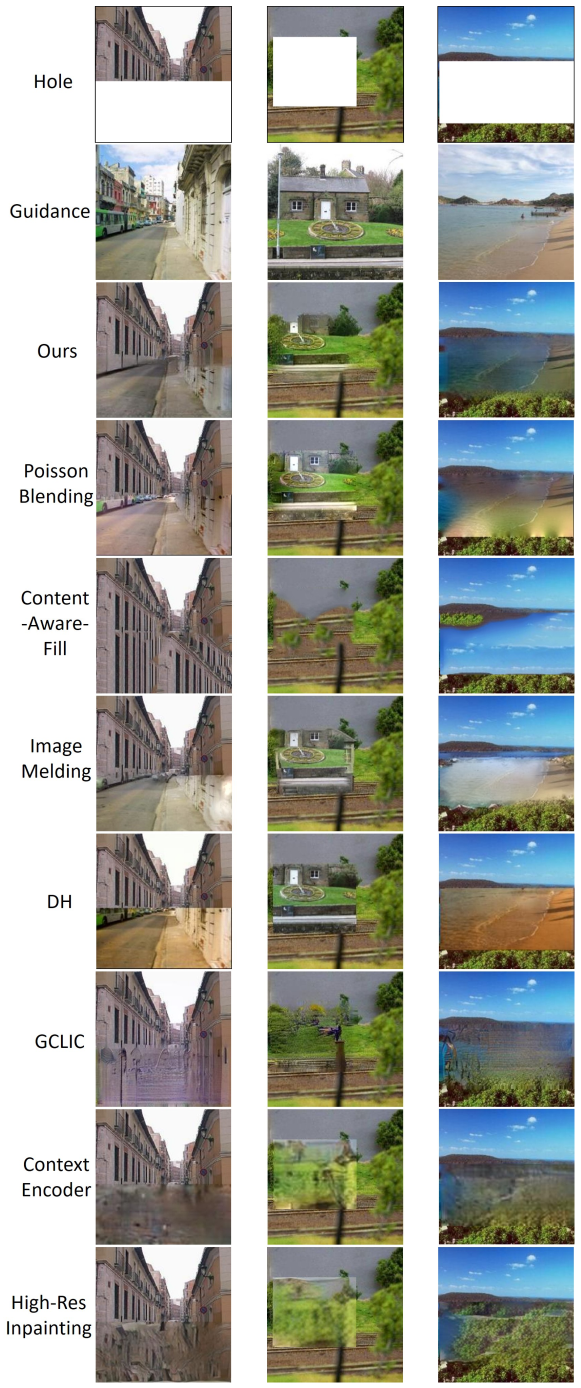

Figure 4. Visual comparison of inpainting results by different methods. Compared to methods that do not use guidance images (*CAF* [2], *CE* [28], *GLCIC* [18] and *HR* [40]), our results look more realistic by transferring matching content of the guidance image to the hole. Compared to the baselines that use guidance images (*PB* [29], *DH* [36] and *IM* [7]), our approach is better at avoiding artifacts along the boundaries of the hole by synthesizing new content for inconsistent regions.

4.4. Localization

While the previous sections evaluate the synthesis, we evaluate in this section our localization method with an ablation study. We use our synthesis network to fill in holes for all localization methods. To draw a direct comparison between our method and baselines, we conduct a relative realism user study. Each pair that is compared contains inpainting results for the same incomplete image and guidance image, but using different localization methods. People are asked to choose the more realistic image after the two images are shown side-by-side for an unlimited time.

Baseline: We compare to *Local Context Matching* [16], which minimizes pixelwise SSD error in *Lab* color space. It has no knowledge of global structure or semantics.

Dataset: We use images in the validation set of ADE20K as incomplete images and use the full images to retrieve semantically similar images as guidance from MSCOCO. We randomly sample 50 images in ADE20K for evaluation, each with 10 retrieved neighbors in MSCOCO as guidance images. Thus, each hole has 10 different hole-fillings. It is more robust to retrieve multiple guidance images than a single one to evaluate localization because it is possible that the guidance image does not contain a good matching patch for hole filling. There are 500 relative comparisons.

Evaluation Method: We use the same approach as in Section 4.3.

Results: The synthesized images based on our localization is rated to be more realistic than *Local Context Matching* in **53.8%** of all the pairwise comparisons. We conjecture that the gain comes from consideration of global structure. Our localization network processes the whole image while *Local Context Matching* only takes into account local information in the context. The gain is relatively small probably because the distribution of features for localization in the synthetic images is different from that in natural images. The localization network trained on the synthetic dataset does not generalize as well to natural images.

5. Conclusion

We introduce an end-to-end inpainting model to localize a matching patch in the guidance image and transfer its content to the hole followed by synthesizing a realistic hole-filling. We conduct four studies to evaluate the power of our approach in localizing fitting patches and synthesizing realistic images. Despite training on a synthetic dataset, our synthesis network generalizes well to natural images. The human perceptual experiments show that our approach synthesizes more realistic images than all the baselines.

Acknowledgments

This work is supported by a National Science Foundation award to Danna Gurari (IIS-1755593). We thank crowd workers for participating in our studies.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24–1, 2009.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.

[5] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 1, 2017.

[6] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

[7] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.*, 31(4):82–1, 2012.

[8] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[9] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *ACM Transactions on graphics (TOG)*, volume 22, pages 303–312. ACM, 2003.

[10] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[11] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[12] M. Elad and P. Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, 2017.

[13] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[14] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[16] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007.

[17] J.-P. Heo, Z. Lin, X. Shen, J. Brandt, and S.-E. Yoon. Shortlist selection with residual-aware distance estimator for k-nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2009–2017, 2016.

[18] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017.

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[20] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[21] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[22] O. R. Joubert, G. A. Rousselet, D. Fize, and M. Fabre-Thorpe. Processing scene context: Fast categorization and object interference. *Vision research*, 47(26):3286–3297, 2007.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics (ToG)*, 24(3):795–802, 2005.

[25] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[27] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[28] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[29] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[30] F. Pitié and A. Kokaram. The linear monge-kantorovitch linear colour mapping for example-based colour transfer. 2007.

[31] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

[32] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[34] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister. Multi-scale image harmonization. In *ACM Transactions on Graphics (TOG)*, volume 29, page 125. ACM, 2010.

[35] M. W. Tao, M. K. Johnson, and S. Paris. Error-tolerant image compositing. In *European Conference on Computer Vision*, pages 31–44. Springer, 2010.

[36] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, X. Lu, and M.-H. Yang. Deep image harmonization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, volume 2, page 5, 2009.

[38] M. Wilczkowiak, G. J. Brostow, B. Tordoff, and R. Cipolla. Hole filling through photomontage. In *BMVC*, volume 5, pages 492–501, 2005.

[39] N. Xu, B. Price, S. Cohen, and T. Huang. Deep image matting. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[40] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.

[41] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016.

[42] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. *arXiv preprint arXiv:1801.07892*, 2018.

[43] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proc. CVPR*, 2017.

[44] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

[45] Z. Zhu, H.-Z. Huang, Z.-P. Tan, K. Xu, and S.-M. Hu. Faithful completion of images of scenic landmarks using internet images. *IEEE transactions on visualization and computer graphics*, 22(8):1945–1958, 2016.