Synthesizing Missing Image Content Guided by Another Image

Anonymous CVPR submission

Paper ID 1125

Abstract

Recent advances in deep generative models have shown promising potential in predicting missing pixel values in an image using surrounding context. However, such models can only generate a single prediction, while a hole may have multiple reasonable fillings. To address this limitation, we propose an end-to-end inpainting network to synthesize realistic hole-fillings conditioned on surrounding context and any new image (we call it a guidance image). Our approach introduces an inpainting CNN to localize the best fitting patch in the guidance image, and transfer its content into the hole. A further benefit of our approach is it can synthesize new content in inconsistent regions. Our perceptual experiments on Amazon Mechanical Turk show that our approach generates more realistic results when filling holes in 70% *of comparisons with five baselines.*

1. Introduction

People often wish to remove undesired content from their photos in a realistic way. Often this involves automated image inpainting (also called image completion and hole filling), the task of filling in the lost part of an image.

One line of work for inpainting fills holes with content from another guidance image. These methods [15, 36] often involve cutting and pasting a semantically similar patch from a large database of images, and then blending them together. Although these methods are good at propagating high-frequency details from the guidance image to the hole, there are often inconsistent regions humans can easily detect. As shown in Figure 1, when replacing the road in the original image with another road from a guidance image, a Cut-Paste-Blend [15] method leaves part of the bus in the completed result. It is not consistent with the context.

Alternative methods [10, 9, 22, 6, 8, 37, 27, 39] for inpainting try to hallucinate missing image regions from surrounding context. Yet, traditional texture synthesis approaches [10, 9, 22, 6, 8, 37] cannot capture global structure by extending texture from surrounding regions. More recently, Phatak et al. [27] trained an encoder-decoder CNN

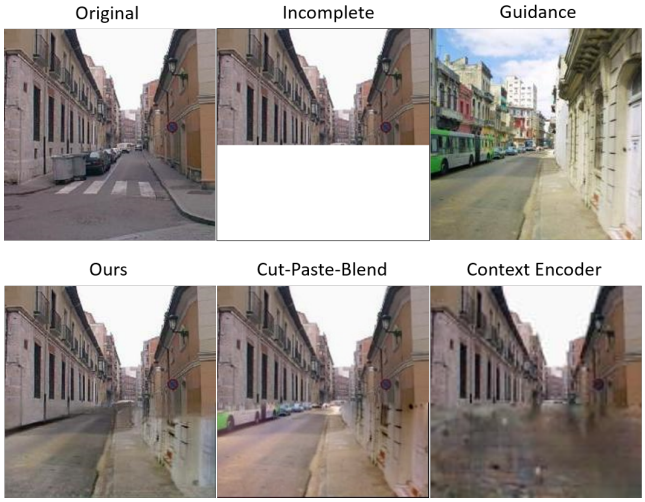

Figure 1. Inpainting results of different methods (bottom row). The hole is created by removing the bounding box of the road in the original image (top row, col 1 and 2). Our method and Cut-Paste-Blend [15] both attempt to transfer the road in the guidance image to the hole (top row, col 3). Context encoder [27] hallucinates content in the hole from context only. The Cut-Paste-Blend method leaves part of the bus that is inconsistent with the context in the result. Our method synthesizes proper building texture in that region.

to predict missing image regions from surrounding pixels. However, it does not address the inherent ambiguity of the problem; e.g., a hole on the lawn may contain a cat, dog, or just grass. A trained model can only generate one result given the context.

To overcome the limitations of the two existing strategies, we introduce an end-to-end neural architecture that synthesizes realistic hole-fillings conditioned on any guidance image. First, an appropriate patch from the guidance image is chosen to fill the hole. We introduce a localization network to locate the best fitting patch in the guidance image and align it with the hole. However, even after localizing the patch from the guidance image, there are often inconsistent regions that do not match with the context if we just paste and blend. Thus, we propose a synthesis net-

work that 1) transfers matching content from the aligned guidance patch to the hole and 2) synthesizes new content to match with the context in inconsistent regions. Compared to approaches based on hallucination from context [10, 9, 22, 6, 8, 37, 27, 39], our approach can obtain multiple realistic hole-fillings given different guidance images. In contrast to Cut-Paste-Blend approaches [15], our approach is able to synthesize new image content in inconsistent regions, and so better match the surrounding context.

We evaluate our proposed inpainting approach to address the following questions: 1) how well does our approach capture true pixel values in the hole?, 2) how often does our approach synthesize realistic images that can fool humans? and 3) how often are our results more realistic than alternative methods? We demonstrate the effectiveness of our approach in capturing true pixel values by an image restoration task. We also conduct human perceptual experiments to show that our approach can synthesize more realistic hole-fillings than alternative approaches. Our automated inpainting approach has the potential to save digital artists from the painful manual effort in creative retouching.

2. Related Work

Image Inpainting: Inpainting is the process of filling in lost parts of images. Existing methods that address inpainting fall into two groups.

The first group uses only the given image context to fill the holes. Classical inpainting [3, 26] fills a hole by propagating isophotes from surrounding context to the hole via diffusion. Texture synthesis methods [10, 9, 2, 22, 6, 8, 37] extend texture from surrounding regions to the holes. Such methods have no means to capture the semantics of the image. More recently, deep neural networks [27, 40, 16, 39] were introduced to regress directly from surrounding context to missing pixel values. While these approaches generate a single result given their learned model, there are multiple reasonable ways to fill the hole. Our model can in contrast generate a diverse set of fillings due to the use of different guidance images.

The second group relies on other images to fill the hole. Scene completion [15] involves cutting, pasting and blending a semantically similar patch from a database of millions of images to the original image. Internet-based inpainting [36] utilizes a more powerful web engine to improve the semantic similarity in nearest neighbor search. These approaches are good at transferring structure and high-frequency details from the guidance image to the hole, but it is rare that the entire image boundary matches perfectly, thus inconsistent regions that fail to match with the context are created. In contrast, our approach synthesizes new consistent content in such regions.

Style Transfer: Image style transfer [12, 19, 13, 11] has demonstrated impressive results in example-based image stylization. It combines the high-level content of one image with the low-level style of another. Our approach, similar to style transfer, combines information of two images, while it differs 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. Our goal is not to transfer low-level style from one image to another, but to synthesize a consistent hole-filling while keeping the high-level content of the guidance image.

Image Harmonization: Image harmonization is the process of adjusting low-level appearances of foreground and background regions to make them compatible when generating realistic composite images. Traditional methods use color and tone matching techniques to ensure appearance consistency, such as multi-scale statistics matching [33], gradient domain [34, 28] or global statistics transferring [29, 30]. Recently, Tsai et al. [35] proposed a learning based method by training a deep CNN to capture both the context and semantic information of the composite images. However, low-level appearance or color adjustments are often inadequate to make the foreground and background compatible. High-level content changes go beyond the scope of harmonization. Accordingly, our approach not only adjusts the appearance of the guidance patch, but also synthesizes new content when necessary to generate more consistent hole-fillings.

3. Network Architecture

3.1. System Overview

To fill a hole in an image given a guidance image, we must first determine which patch in the guidance image best fits in content around the hole. We then use the patch information to help fill the hole wherever the content is consistent and synthesize new content where it is not consistent. To achieve these aims, our inpainting network consists of two components: a localization network and a synthesis network. We describe two networks and how they achieve our aims below. The overall framework is shown in Figure 2.

3.2. Localization network

The localization network decides which patch in the guidance image to use to fill the hole in the original image. It is a CNN, derived from the *VGG* architecture [32]. The input to localization CNN is a concatenation of the incomplete image and the guidance image of size 224×224. Then it uses 13 convolution layers to compute a spatial feature representation of dimensions $7 \times 7 \times 512$, followed by three fully connected layers to predict the six parameters of affine transformation. We concatenate the guidance image and the incomplete image with its binary hole mask as input to the localization network. The hole mask can be inferred from the incomplete image, but it is easier to train with the

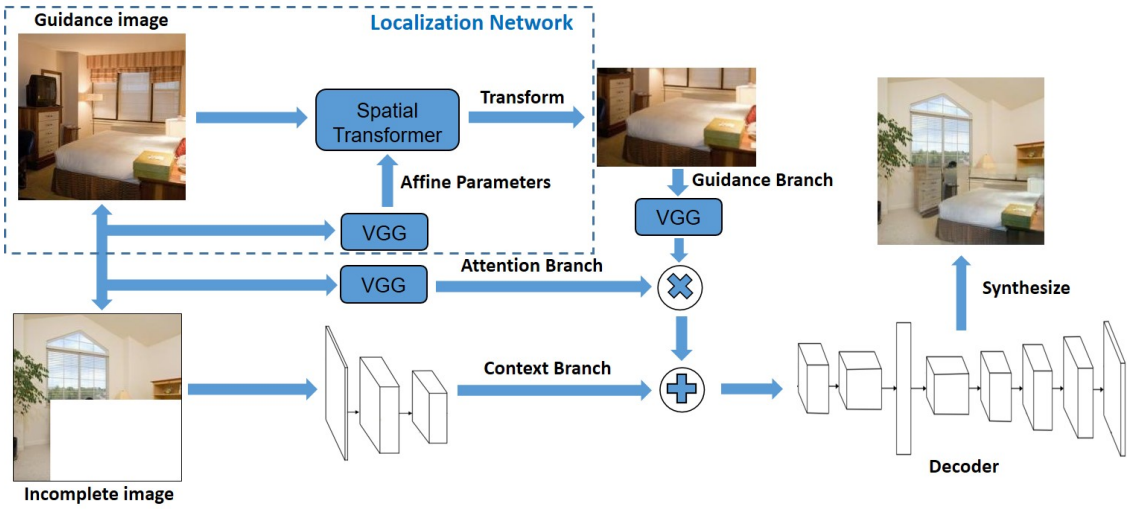

Figure 2. Our proposed inpainting framework includes a 1) localization network which locates the best fitting patch in the guidance image and aligns it with the hole and 2) synthesis network which encodes both the incomplete image and the aligned guidance image to synthesize a reasonable patch to fill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

mask specified explicitly in the input. Batch normalization is adopted after each convolutional layer to improve gradient flow. We have a ReLU nonlinearity after each layer except the last fully connected layer because the output affine parameters can be negative.

We use spatial transformer [18] to align the guidance image with the hole given the predicted transformation parameters. Spatial transformer is able to warp the input feature map based on the transformation parameters using a differentiable image sampling kernel.

3.3. Synthesis Network

The synthesis network synthesizes realistic images conditioned on the incomplete image and a guidance image. It is an encoder-decoder CNN. The encoder encodes the incomplete image and the guidance image into a latent feature representation and the decoder decodes that into a consistent image. The encoder consists of three encoding branches. The context branch encodes high-level features from surrounding context. The guidance branch computes high-level content of the aligned guidance image. Because not all the information in the guidance branch is useful for hole synthesis, to block the features in inconsistent regions from entering the decoder, we have a third attention branch. This branch predicts an attention map indicating inconsistent regions with low weights. The guidance branch is multiplied by the attention map to discard features from inconsistent regions. We insert consistent high-level features of the guidance image directly into the high-level feature map of the incomplete image by adding the guidance and context branch together.

In addition to the high-level content, we also desire to transfer fine-grained details from the guidance patch to the hole. However, fine-grained details are lost when going through the bottleneck layer where the number of neurons is the smallest in the encoder-decoder pipeline. To bypass the bottleneck, we draw skip-layer connections [31] from the guidance branch to the decoder directly to allow useful details to propagate to the output.

Context Branch: The context branch is fully convolutional, following the *VGG* architecture until *conv3_3*. It uses six convolutional layers to encode the incomplete image into an abstract feature representation of dimension $56 \times 56 \times 256$. The input contains an additional channel of binary hole mask.

Guidance Branch: The guidance branch uses pretrained *VGG* on *ImageNet* until *conv3_3*. The dimension of the encoded feature representation is the same as that of the context branch so that they can be easily added later. Our guidance branch uses pretrained models on visual recognition tasks to make sure the output representation encodes high-level semantic features. It is important to point out that we attempt to transfer high-level content from the guidance to the hole. The *conv3_3* feature map contains high-level features while discarding some low-level information we expect to change, such as color and lighting.

Attention Branch: The attention branch has a similar architecture as the context branch. Its input is a concatenation of the incomplete and guidance image. There is another convolutional layer which outputs 1-channel feature map in the end, indicating relative weight for each location. It is followed by a sigmoid layer to bound the relative weights in the range from 0 to 1. The weights in the attention map are invariant across channels. This branch can block inconsistent regions in the guidance feature map by assigning small weights in the attention map.

Decoder: The guidance branch is added to the context branch after it is multiplied by the attention map. The decoder takes the new feature map and synthesizes completed image of the original size. It further uses *conv3_3* to *fc7* in *VGG* architecture to extract a more compact representation of dimension 4096. It is necessary to add fully connected layers because the task of inpainting requires semantic perception of the whole image. If the architecture is limited only to convolutional layers, there is no way to propagate information from a corner of the feature map to another. The compact representation then goes through a series of bilinear upsampling followed by convolution to generate image completion of the original size 224×224. We do not use upconvolutions because they tend to introduce characteristic artifacts [25]. In the end there is a *tanh* layer to constrain the output in the normalized range.

Skip-layer Connection: Encoder-decoder networks downsample the image to encode high-level details, but can lose fine-grained details that we need in synthesis. To recover these fine deteails, we draw skip-layer connections [31] from the guidance branch to the decoder directly to allow useful details to bypass the bottleneck. The *conv1_2* and *conv2_2* feature maps in the guidance branch are concatenated with the corresponding symmetric feature maps in the decoder. We also draw such connections between different layers inside the decoder. These low level feature maps at different hierarchies provide different levels of abstraction.

4. Training Approach

We need a large dataset to train our localization and synthesis networks. Unfortunately no such dataset is available. Inspired by prior work which has shown that models trained on synthetic data can generalize well to real images [4, 38], we create a large-scale synthetic inpainting dataset by creating corrupted patches as the guidance to fill in holes on the original images. This is exemplified in Figure 3. We then describe the loss functions we use to train our localization and synthesis networks.

4.1. Training Data Generation

It requires a large dataset of pairs of incomplete image and guidance image with their corresponding inpainting ground truth to train our localization and synthesis network in a supervised manner. To create such training pairs, we corrupt a patch on a source image by pasting and blending it into a random target image, and then use the corrupted patch as guidance to fill the hole on the source image. Specifically, we first generate a random hole (green bounding box in Figure 3) on the source image and crop out a random patch (red bounding box) inside the hole. Then, we paste and blend the cropped patch (red bounding box) in a random position of a random target image. The blending step adjusts the color and appearance of the source patch, and

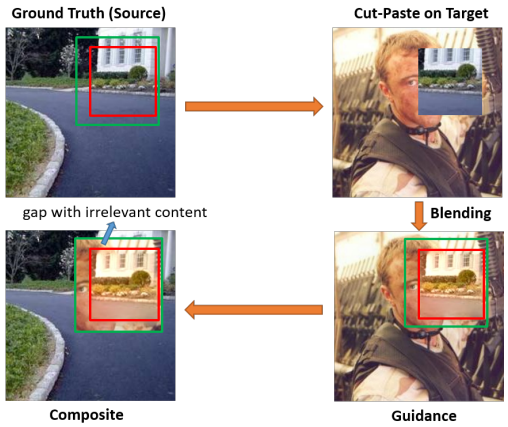

Figure 3. The procedure of generating synthetic training data.

blends in new content from the target image. The blended image finally serves as the guidance image to fill the hole on the source image. The original source image is the inpainting ground truth. Trained on such pairs in supervised manner, the localization network is expected to localize the patch coming from the source image in the blended image while the synthesis network is expected to synthesize the original patch conditioned on the corrupted one.

Note that the guidance patch is sufficiently different from the corresponding ground truth in the dataset we create, as shown in the composite image in Figure 3. Otherwise the solution is trivially copy-pasting the guidance patch to the hole. The blending step adjusts the color and appearance significantly, and there is a gap between the hole (green bounding box) and the cropped patch (red bounding box). In the guidance image, that gap is completely filled with irrelevant content from the target image. There is likely no correlation between the guidance and ground truth in the gap. The synthesis network has to learn to synthesize new content, which is not in the guidance patch, to fill that gap. It is desirable because synthesizing new content for inconsistent regions tends to make the result more realistic, and it is very likely that the inconsistent regions lie near the boundary. The width of the gap are random in a range, making it possible for the synthesis network to learn where to transfer existing content and where to synthesize new content.

Although we train our inpainting network on a synthetic dataset we create, we will show experimentally that it generalizes well on natural images.

4.2. Image Synthesis Loss

We describe here the loss function we use to train our synthesis network. It is important which loss function is chosen because the problem is underconstrained. Minimizing *Mean Square Error (MSE)* encourages averaging of plausible solutions, and so struggles to recover high-frequency details. In contrast, recent work [19, 7, 23] 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) [14] are shown to be successful in image generation as an adaptive loss [27, 17, 23]. Consequently, we combine a perceptual loss on multiple layers of a visual perception network ϕ with an adversarial loss to train the synthesis network:

$$l_g(G) = \frac{1}{N}\sum_I l^{\phi}_{perc}(I, G(I)) + \lambda_{adv} l_{adv}(I)$$

The number of training pairs is N. Layers in perception network provide image representations at increasing levels of abstraction. 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]. The adversarial loss encourages the network to favor image generation that reside on the natural image manifold.

Perceptual Loss: 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 $C_j \times H_j \times W_j$. The output image is $\hat{I} = G(I)$ while the ground truth is I. The perceptual loss is defined as the Euclidean distance between the output feature map and ground truth feature map:

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

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 [14] 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:

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

With this formulation, the discriminator D learns an adaptive loss encouraging the synthesized image to reside on the natural image manifold. The generator G is trained to fool the differentiable discriminator D, creating solutions highly similar to natural images.

The generator G has the architecture defined in Section 3. For discriminator D, following [16], we have two networks: a global discriminator and a local one. The global discriminator takes as input the whole completed image to judge global consistency of a scene while the local discriminator only takes a patch centered around the completed region to encourage local consistency with the surrounding area. The global and local feature representations are concatenated together, followed by a fully-connected layer to output a continuous real/fake probability. When training the generator, the adversarial loss $l_{adv}(I)$ is defined based on the real/fake probability of the discriminator D as:

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

4.3. Localization Loss

We now describe the loss function we use to train our localization network. Although the spatial transformer has a differentiable sampling kernel, 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 direct supervision on the localization network. The goal of localization is to estimate the transformation applied on the guidance image, specifically the parameters of affine transformation. Suppose $x_i(I)$ is a vector representing homogeneous coordinate of the ith sampled point in input source image I and M points are sampled in each image. $T(I)$ is the ground truth affine transformation matrix for image I, which is known when we create our dataset. $\bar{T}(I)$ is the estimated transformation matrix. The localization loss l_{loc} with respect to $\bar{T}$ is defines as the average of Euclidean distances between estimated point positions after transformation and their corresponding ground truth locations:

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

The localization loss encourages the estimated transformation matrix to be close to the ground truth transformation by minimizing differences on sampled points.

4.4. Implementation

We create the synthetic dataset from ADE20K [41]. There are $20,210$ images in the training set of ADE20K and 2000 in the validation set. We generate 50 pairs for each image, resulting in $1,010,500$ pairs for training. The localization network and synthesis network are trained separately with localization loss and synthesis loss. If they are trained jointly, the synthesis network will blur the image generation to accommodate localization errors.

For training CNNs, we use Tensorflow [1] framework. We choose Adam [21] 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.

5. 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(RQ):

- 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 sets of experiments to address the RQs, evaluating synthesis and localization separately. In Sec. 5.1, we measure how well our synthesis network restores true pixel values of the original patch given a corrupted one as guidance. Sec. 5.2 evaluates the effectiveness of our synthesis network in synthesizing realistic images that can fool humans. In Sec. 5.3, we conduct pairwise comparisons between our synthesis and baselines to measure how often our synthesis is preferred to alternative methods. In Sec. 5.4, we compare our localization to Local Context Matching [15] in pairwise comparisons to investigate which localization method results in more realistic synthesis.

5.1. Image Restoration

Baelines: We compare our synthesis network to five baselines that fall into four groups.

- *Texture syntheisis*: Content-Aware-Fill (CAF) using PatchMatch [2].
- *Deep Generative Models*: Context encoder (CE) [27] retrained with *l2* and adversarial loss on ADE20K. High-Resolution image inpainting (HR) [39].
- *Harmonization*: Deep harmonization (DH) [35].
- *Blending*: Poisson blending (PB) [28].

CAF, CE and HR hallucinate the missing region from the context without guidance. DH, PB and our method use a corrupted patch from the original image as guidance.
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 in three metrics: L1 Loss, L2 Loss and PSNR.
Results: As observed in Table 1, our synthesis network consistently offers significant gains over baselines in all three metrics. Our synthesis method decreases the error by **6.02%** and **2.10%** 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 **4.77dB** in PSNR, achieving **20.68dB**. We attribute the success in this task to the nature of our approach: it can transfer the correct structure of the guidance image and synthesize appropriate new content when necessary (fill the gap with new content in this case).

Method	Mean L1 Loss	Mean L2 Loss	PSNR
CAF	15.43%	5.09%	14.38dB
CE	12.91%	3.21%	15.91dB
HR	13.05%	3.29%	15.83dB
DH	18.87%	6.02%	12.73dB
PB	13.63%	3.28%	15.41dB
Ours	**6.89%**	**1.11%**	**20.68dB**

Table 1. Numerical comparison on image restoration. Higer PSNR value is better. % in the table is to facilitate reading.

5.2. Absolute Realism

We also evaluate absolute realism – how often the generated images by our synthesis network are realistic enough to fool humans. In practical image editing applications, users often have the demand to replace an object in an image with new content in another image. Thus, we evaluate the absolute realism of our synthesized images when the guidance is a new natural image different from the source incomplete image. We use a rectangular hole as bounding boxes are often used for selecting/identifying objects. We show people a random image from a mixed dataset of real and synthesized images on Amazon Mechanical Turk (AMT), and ask them to guess if the shown image is real or fake. We balance the number of real and fake images to avoid a strong real/fake prior. Joubert et al. [20] showed that 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. We show each image for 4 seconds to give people sufficient time to determine its realism.
Baselines: We use the same five baselines as in Section 5.1.
Dataset: We use images in the validation set of ADE20K as source incomplete images and retrieve the nearest neighbor in MSCOCO [24] as the guidance image based on semantic features extracted from a pretrained CNN. The guidance image needs to be semantically similar to the source image. Otherwise there is no chance to find a matching patch in the guidance image. We use the segmentation label in ADE20K to crop the bounding box around an object as the hole. To evaluate synthesis separately, we use Local Context Matching [15] as localization for all the synthesis methods. Local context matching 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

Method	Realism
Natural Image	98.67%
CE	7.00%
HR	7.00%
CAF	19.00%
PB	17.00%
DH	25.00%
Ours	**27.00%**

Table 2. User study results evaluating the absolute realism of synthesized images. The numbers in the table are the percentage of images deemed to be real.

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 observed in Table 2, although it is still far from natural images, **27%** of the synthesized images by our approach are deemed to be real. It outperforms all the baselines significantly except for *Deep Harmonization(DH)*. We will show in Section 5.3 that our synthesized images are rated to be much more realistic than *Deep Harmonization* in a more careful pairwise comparison. Despite our synthesis network being trained on a synthetic dataset, it generalizes well to natural images.

Our synthesis network also has the potential to generate multiple realistic hole-fillings given difference guidance images. Figure 4 shows three examples each with four reasonable hole-fillings.

5.3. Relative Realism

To evaluate our method more thoroughly, we evaluate relative realism – how often are the generated images by our synthesis network considered to be more realistic than baselines. We conduct pairwise comparisons between our method and top performing baselines in the aforementioned absolute realism test. Each pair contains two images synthesized for the same incomplete image using the same guidance by our synthesis network and a baseline. The users are asked to select the more realistic image after they are shown a pair side-by-side for unlimited time.

Baselines: We use three top performing baselines from Section 5.2: *Deep Harmonization(DH)*, *Content-Aware-Fill(CAF)*, and *Poisson Blending(PB)*.

Dataset: We use the same dataset as in Section 5.2.

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 than the baselines.

Results: As shown in Table 3, people rate our synthesized images more realistic than baselines in at least **70%** of test images. In particular, our results are more realistic than *Deep Harmonization*, which is the best baseline in the absolute realism experiment, for **71%** of images. There are much bigger differences between our synthesis and baselines than in absolute realism. We conjecture the reason is that in some cases harmonization and blending generate results that look realistic if you just glance at the image. However, if you really compare the results carefully, you will notice the inconsistent regions they leave. In contrast, our synthesis is effective in generating consistent hole-fillings that look more realistic. Figure 5 exemplifies the hole-filling results of different methods. As shown in these examples, 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.

Method	PB	DH	CAF
P(Ours are better)	76%	71%	70%

Table 3. Relative realism evaluation. Each cell lists the percentage of pairs in which synthesized images by our approach are rated more realistic than corresponding baseline. Chance is at 50%.

5.4. Localization

We evaluate the synthesis network in the previous three sections. In this section, we evaluate our localization method by 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 baseline, we conduct a relative realism user study. Each pair to be compared contains inpainting results for the same incomplete image and guidance image using different localization methods. People are asked to choose the more realistic one after the two images are shown side-by-side for unlimited time.

Baseline: We compare to *Local Context Matching* [15] that 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 source incomplete images and 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 5.3.

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 be-

Figure 4. Multiple inpainting results for the same hole. On the left is the incomplete image, followed by four images showing multiple inpainting results given different guidance images, followed finally by the original image.

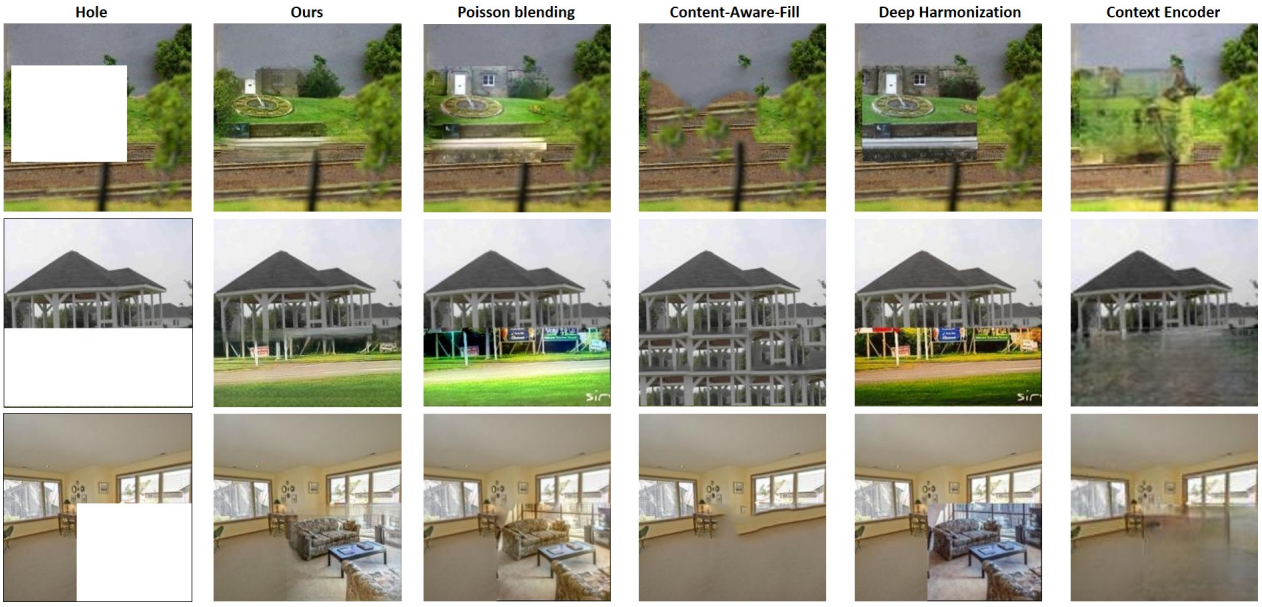

Figure 5. Visual comparisons of inpainting results by different methods. The incomplete image comes from validation set of ADE20K and the guidance image is retrieved from MSCOCO. Our synthesized images are consistent with the context while *PB* and *DH* have inconsistent regions near the boundary. Compared to *CAF* and *CE*, our results look more realistic by transferring matching content of the guidance image to the hole.

cause 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 that well to natural images.

6. 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. 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.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 5

[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. 2, 6

[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. 2

[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. 4

[5] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *arXiv preprint arXiv:1707.09405*, 2017. 5

[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. 1, 2

[7] 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. 4

[8] 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. 1, 2

[9] 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. 1, 2

[10] 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. 1, 2

[11] M. Elad and P. Milanfar. Style transfer via texture synthesis. *IEEE Transactions on Image Processing*, 26(5):2338–2351, 2017. 2

[12] 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. 2

[13] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. *arXiv preprint arXiv:1611.07865*, 2016. 2

[14] 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. 5

[15] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007. 1, 2, 6, 7

[16] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017. 2, 5

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. 5

[18] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 3

[19] 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. 2, 4

[20] 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. 6

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[22] 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. 1, 2

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016. 4, 5

[24] 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. 6

[25] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016. 4

[26] 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. 2

[27] 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. 1, 2, 5, 6

[28] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on graphics (TOG)*, volume 22, pages 313–318. ACM, 2003. 2, 6

[29] F. Pitié and A. Kokaram. The linear monge-kantorovitch linear colour mapping for example-based colour transfer. 2007. 2

[30] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001. 2

[31] 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. 3, 4

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[33] 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. 2

[34] M. W. Tao, M. K. Johnson, and S. Paris. Error-tolerant image compositing. In *European Conference on Computer Vision*, pages 31–44. Springer, 2010. 2

[35] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, X. Lu, and M.-H. Yang. Deep image harmonization. *arXiv preprint arXiv:1703.00069*, 2017. 2, 6

[36] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, volume 2, page 5, 2009. 1, 2

[37] M. Wilczkowiak, G. J. Brostow, B. Tordoff, and R. Cipolla. Hole filling through photomontage. In *BMVC*, volume 5, pages 492–501, 2005. 1, 2

[38] N. Xu, B. Price, S. Cohen, and T. Huang. Deep image matting. *arXiv preprint arXiv:1703.03872*, 2017. 4

[39] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. *arXiv preprint arXiv:1611.09969*, 2016. 1, 2, 6

[40] 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. 2

[41] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proc. CVPR*, 2017. 5