High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis

Chao Yang¡1, Xin Lu§2, Zhe Lin†2, Eli Shechtman‡2, Oliver Wang*2, and Hao Li‖1,3,4

[1]University of Southern California
[2]Adobe Research
[3]Pinscreen
[4]USC Institute for Creative Technologies

Abstract

Recent advances in deep learning have shown exciting promise in filling large holes in natural images with semantically plausible and context aware details, impacting fundamental image manipulation tasks such as object removal. While these learning-based methods are significantly more effective in capturing high-level features than prior techniques, they can only handle very low-resolution inputs due to memory limitations and difficulty in training. Even for slightly larger images, the inpainted regions would appear blurry and unpleasant boundaries become visible. We propose a multi-scale neural patch synthesis approach based on joint optimization of image content and texture constraints, which not only preserves contextual structures but also produces high-frequency details by matching and adapting patches with the most similar mid-layer feature correlations of a deep classification network. We evaluate our method on the ImageNet and Paris Streetview datasets and achieved state-of-the-art inpainting accuracy. We show our approach produces sharper and more coherent results than prior methods, especially for high-resolution images.

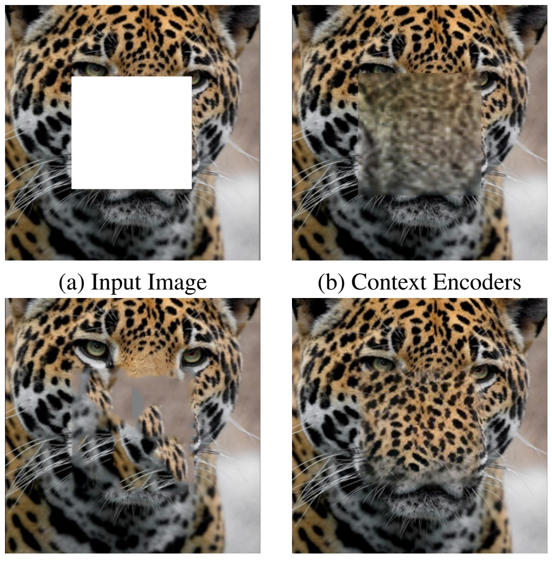

Figure 1. Qualitative illustration of the task. Given an image (512×512) with a missing hole (256×256) (a), our algorithm can synthesize sharper and more coherent hole content (d) comparing with Context Encoders [32] (b) and PatchMatch [1] (c).

1. Introduction

Before sharing a photo, users may want to make modifications such as erasing distracting scene elements, adjusting object positions in an image for better composition, or recovering image content in occluded image areas. These, and many other editing operations, require automated hole-filling (image completion), which has been an active research topic in the computer vision and graphics communities for the past few decades. Due to its inherent ambiguity and the complexity of natural images, general hole-filling remains challenging.

Existing methods that address the hole-filling problem fall into two groups. The first group of approaches [14, 13, 27, 26] rely on texture synthesis techniques, which fill in the hole by extending textures from surrounding regions [6, 12, 40, 41, 23, 24, 2]. A common idea in these techniques is to synthesize the content of the hole region in a coarse to fine manner, using patches of similar textures. In [12, 41], multiple scales and orientations are introduced to find better matching patches. Barnes et al. [2] proposed PatchMatch as a fast approximate nearest neighbor patch search algorithm. Although such methods are good at propagating high-frequency texture details,

¡chaoy@usc.edu
§xinl@adobe.com
†zlin@adobe.com
‡elishe@adobe.com
*owang@adobe.com
‖hao@hao-li.com

they do not capture the semantics or global structure of the image. The second group of approaches hallucinate missing image regions in a data-driven fashion, leveraging large external databases. These approaches assume that regions surrounded by similar context likely possess similar content [19]. This approach is very effective when it finds an example image with sufficient visual similarity to the query but could fail when the query image is not well represented in the database. Additionally, such methods require access to the external database, which greatly restricts possible application scenarios.

More recently, deep neural network is introduced for texture synthesis and image stylization [15, 16, 28, 3, 39, 22]. In particular, Phatak et al. [32] trained an encoder-decoder CNN (Context Encoders) with combined ℓ_2 and adversarial loss [17] to directly predict missing image regions. This work is able to predict plausible image structures, and is very fast to evaluate, as the hole region is predicted in a single forward pass. Although the results are encouraging, the inpainting results of this method sometimes lack fine texture details, which creates visible artifacts around the border of the hole. This method is also unable to handle high-resolution images due to the difficulty of training regarding adversarial loss when the input is large.

In a recent work, Li and Wand [28] showed that impressive image stylization results can be achieved by optimizing for an image whose neural response at mid-layer is similar to that of a content image, and whose local responses at a low convolutional layers resemble local responses from a style image. Those local responses were represented by small (typically 3×3) *neural patches*. This method proves able to transfer high-frequency details from the style image to the content image, hence suitable for realistic transfer tasks (e.g., transfer of the look of faces or cars). Nevertheless, transferring of more artistic styles are better addressed by using gram matrices of neural responses [15].

To overcome the limitations of aforementioned methods, we propose a hybrid optimization approach that leverages the structured prediction power of encoder-decoder CNN and the power of neural patches to synthesize realistic, high-frequency details. Similar to the style transfer task, our approach treats the encoder-decoder prediction as the global content constraint, and the local neural patch similarity between the hole and the known region as the texture (*style*) constraint.

More specifically, the content constraint can be constructed by training a global content prediction network similar to Context Encoders, and the texture constraint can be modeled with the image content surrounding the hole, using the patch response of the intermediate layers using the pre-trained classification network. The two constrains can be optimized efficiently using backpropagation with limited-memory BFGS. In order to further handle high-resolution images with large holes, we propose a multi-scale neural patch synthesis approach. For simplicity of formulation, we assume the test image is always cropped to 512×512 with a 256×256 hole in the middle. We then create a three-level pyramid with a step-size of two, downsizing the image by half at each level. It renders the lowest resolution 128×128 with 64×64 hole. We then perform the hole filling task in a coarse-to-fine manner. Initialized with the output of content prediction network at lowest level, at each scale (1) we perform the joint optimization to update the hole, (2) upsample to initialize the joint optimization and set content constraint for the next scale. We then repeat this until the joint optimization is finished at the highest resolution (Sec. 3).

We show experimentally that the proposed multi-scale neural patch synthesis approach can generate more realistic and coherent results preserving both the structure and texture details. We evaluate the proposed method quantitatively and qualitatively on two public datasets and demonstrate its effectiveness over various baseline and existing techniques as shown in Fig. 1 (Sec. 4).

The main contributions of this paper are summarized as follows:

- We propose a joint optimization framework that can hallucinates missing image regions by modeling a global content constraint and local texture constraint with convolutional neural networks.
- We further introduce a multi-scale neural patch synthesis algorithm for high-resolution image inpainting based on the joint optimization framework.
- We evaluate the proposed method on two public datasets and demonstrate its advantage over baseslines and existing techniques.

2. Related Work

Structure Prediction using Deep Networks Over the recent years, deep convolutional neural network (CNN) approaches have significantly helped advance image classification performance, as presented in [25, 36, 37, 20]. Meanwhile, researchers have started utilizing deep neural networks for structured prediction [29, 4, 30, 7, 38, 17, 18, 21, 9, 31], semantic segmentation of images [29, 4, 30], and image generation [17, 18, 7, 31]. We are motivated by the structured prediction power of these recent CNNs and in this work apply it to the problem of hole filling. Unlike the image generation tasks discussed in [11, 17, 18, 7], where the input is a random noise vector and the output is an image, our goal is to predict the content in the hole, conditioned on the *known image regions*. Recently, a hole filling autoencoder-type network has been explored in [32], which achieved promising results by combining an L_2 loss with an adversarial loss form a discriminator network. In our work, we use this approach as a global

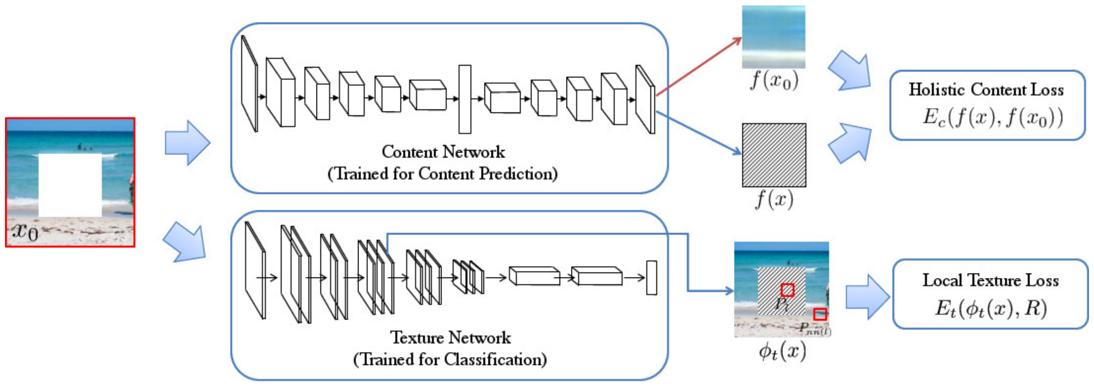

Figure 2. Framework Overview. Our method solves for an unknown image ($\tilde{x}$) using two joint loss functions, the holistic content loss (E_c) and the local texture loss (E_t) shown here. The holistic content loss is derived by feeding the image (x) into a pre-trained content prediction network and comparing the output ($f(x)$) with the *reference* content prediction ($f(x_0)$). The local texture loss is derived by feeding x into an pre-trained network (referred to as texture network) and comparing local neural patches on its feature maps.

content prediction network [32] to initialize our multi-scale neural patch synthesis algorithm at its lowest scale-level.

Style Transfer In order to create realistic image textures, our work is motivated by the recent success of neural style transfer [15, 16, 28, 3, 39, 22]. These approaches are largely used to generate images combining the "style" of one image and the "content" of another image. We show similar techniques able capable of solving the image inpainting problem. Given an incomplete image, the content of the missing region can be iteratively refined by transferring the textures and the high-frequency details from the known region.

3. The Approach

3.1. Framework Overview

We seek an inpainted image $\tilde{x}$ that optimizes over the loss function, which is formulated as a combination of three terms: the holistic *content* term, the local *texture* term, and the *tv-loss* term. The content term is a global structure constraint that captures the semantics and the global structure of the image, and the local term refines the local textural to by making it consistent with the known region. Both the content term and the texture term are computed using a pre-trained network with fixed parameters (Figure 2).

To model the content term, we first train the holistic **content network** f. The input is an image with the central squared region removed and filled with the mean color, and the ground truth image x_t is the original content in the center. We trained on two datasets, as discussed in Section 4. Once the content network is trained, we can use the output of the network $f(x_0)$ as the content constraint for joint optimization.

The goal of the texture term is to ensure that the image content in the missing region is "similar" locally to that of its surrounding. We define similarity on *neural patches*, which encode mid-level image information, and have been successfully used in the past to capture image style [28]. To compute this, we feed the image x into a pre-trained VGG network [35] (we refer to this network as local **texture network** in this paper) and enforce that the response of small (typically 3×3) neural patches inside the hole region is similar to neural patches *outside* the hole at a pre-determined feature layer of the network. In practice we use the combination of *relu3_1* and r*elu4_1* layer to compute neural patches. In our implementation, the content network adapts from the context encoder network [32], and we use VGG-19 as the texture network.

We iteratively update x by minimizing the joint loss through gradient backpropagation on the two convolutional networks. During the optimization, we fix the parameters of the content prediction network and the texture network, and only update the image x by iterative gradient backpropagation.

This proposed framework naturally applies to high-resolution image inpainting problem by separating content and texture terms. Given a high-resolution image with a large hole, we first downsize the image and obtain an reference content. We then upsample this result and take it as an initialization for joint optimization at a finer scale. Applying the texture term to arbitrary resolution images is straightforward with the "fully convolutional" nature of the texture network. Meanwhile, we upsample the content reference to compute the content. In addition, the proposed framework works well for arbitrary shape holes with the texture term.

In the following, we present the details of the two constraints, loss functions, and the optimization algorithm.

3.2. The Joint Loss Function

Given the input image x_0 we would like to find the unknown output image x. We use R to denote a hole region in x, and R^ϕ to denote the corresponding region in a feature map $\phi(x)$ of a convolutional network. $h(\cdot)$

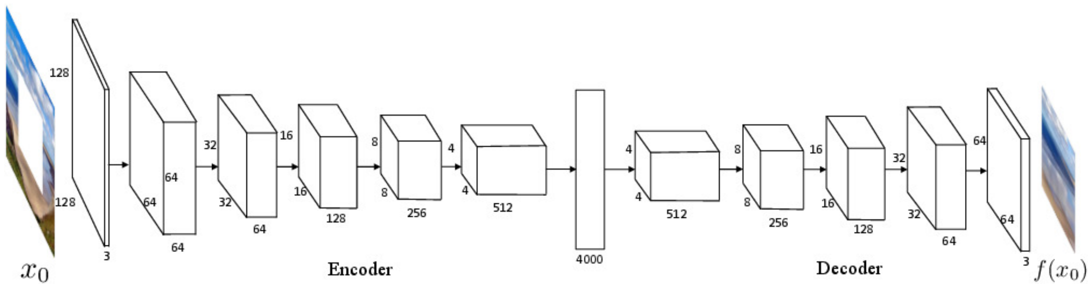

Figure 3. The network architecture for structured content prediction. Unlike the ℓ_2 loss architecture presented in [32], we replaced all ReLU/ReLU leaky layers with the ELU layer [5] and adopted fully-connected layers instead of channel-wise fully-connected layers. The ELU unit makes the regression network training more stable than the ReLU leaky layers as it can handle large negative responses during the training process.

defines the operation of extracting a sub-image or sub-feature-map in a rectangular region, i.e. $h(x, R)$ returns the color content of x within R, and $h(\phi(x), R^\phi)$ returns the content of $\phi(x)$ within R^ϕ, respectively. Given an image x, the content prediction network is represented by function $f(x)$ and texture network is denoted by t.

The optimal reconstruction (hole filling) result $\tilde{x}$ is obtained by solving the following minimization problem:

$$\tilde{x} = \arg\min_x E_c(f(x), f(x_0)) + \alpha E_t(\phi_t(x), R^\phi) + \beta \Upsilon(x), \tag{1}$$

where $\phi_t(\cdot)$ represents a feature map at an intermediate layer in the texture network t, and α is a weight reflecting the importance between the two terms. We set α and β to 5e−6 in our implementation.

The first term E_c in Equation 1 models **the holistic content constraint**, which penalizes the forward prediction $f(x)$ of the image x based on its deviation from $f(x_0)$:

$$E_c(f(x), f(x_0)) = \| f(x) - f(x_0) \|_2^2 \tag{2}$$

The second term E_t in Equation 1 models **the local texture constraint**, which penalizes the discrepancy of the appearance of texture patterns inside and outside the hole. To measure the texture consistency, we first choose a certain feature layer in the network t, and extract its feature map ϕ_t. For each local query patch P of size $s \times s \times c$ in the hole R^ϕ, we find its most similar patch outside the hole, and compute the loss by averaging the distances of the query patch and its nearest neighbor.

$$E_t(\phi_t(x), R) = \frac{1}{|R^\phi|} \sum_{i \in R^\phi} \| h(\phi_t(x), P_i) - h(\phi_t(x), P_{nn(i)}) \|_2^2 \tag{3}$$

where $|R^\phi|$ is the number of patches sampled in the region R^ϕ, P_i is the local neural patch centered at location i, and $nn(i)$ is the computed as

$$nn(i) = \arg\min_{j \in \mathcal{N}(i) \wedge j \notin R^\phi} \| h(\phi_t(x), P_i) - h(\phi_t(x), P_j) \|_2^2 \tag{4}$$

where $\mathcal{N}(i)$ is the set of neighboring locations of i excluding the overlap with R^ϕ. The nearest neighbor ($nn(i)$) can be computed as a layer in a CNN, as shown in [28]. We now describe how we train the content network used in the loss function E_c.

We also add the TV loss term to encourage smoothness:

$$\Upsilon(x) = \sum_{i,j} ((x_{i,(j+1)} - x_{i,j}^2) + (x_{i+1,j} - x_{i,j}^2)) \tag{5}$$

3.3. Content Prediction Network

A straightforward way to learn the content prediction network is to train a regression network f to make the response $f(x_0)$ of an input image x_0 (with the unknown region) approximate the ground truth x_g at region R. Recent studies have used various loss functions for image restoration tasks in general, for instance, ℓ_2 loss, SSIM loss [42, 10, 33], ℓ_1[42] loss, perceptual loss [22], and adversarial loss [32]. We experimented with ℓ_2 loss and adversarial loss. For each training image, the ℓ_2 loss is defined as:

$$L_{l2}(x, x_g, R) = \| f(x) - h(x_g, R) \|_2^2 \tag{6}$$

The adversarial loss is defined as:

$$L_{adv}(x, x_g, R) = \max_D E_{x \in \mathcal{X}} [\log(D(h(x_g, R))) + \log(1 - D(f(x)))] \tag{7}$$

where D is the adversarial discriminator.

We jointly use the ℓ_2 loss and the adversarial loss as in the Context Encoders [32] as:

$$L = \lambda L_{l2}(x, x_g, R) + (1 - \lambda) L_{adv}(x, x_g, R) \tag{8}$$

where λ is 0.999 in our implementation.

We set R to be a square region and fix its location and scale in the network input and restrict the network output to predict only the missing region R. We tried to randomize the location of missing regions, but found that the training required a much larger dataset with a larger capacity network. This is because the network has to both localize the unknown region and predict the content in the hole during training. Instead, we set R to be at the center of the input training image, and the hole pixels are initialized as constant mean color pixels subtracted by the CNN. Experiment shows that despite this assumption during training, we are still able to predict the content for arbitrary holes in the image.

The content network architecture is shown in Figure 3, which is motivated by the ℓ_2 loss architecture proposed in [32], with two main differences. One is that we replaced all ReLU/leaky-ReLU layers with ELU layers [5] because we found that with ReLU/leaky-ReLU, the network loss can increase after a few iterations, and the network does not train properly. The second difference is that we use the standard fully-connected layer instead of the channel-wise fully connected layer, as we do not need to compress the network size with just the ℓ_2 loss (no adversarial loss [32]).

3.4. Texture Network

We use the VGG-19[35] network pre-trained for ImageNet classification as the texture network, and use the *relu3_1* layer and the *relu4_1* layer to calculate the texture term, which made the optimization faster and slightly more accurate than only using *relu3_1*. As an alternative, we experimented with the content network discussed in the previous section as the texture network, but found the results worse than using the pre-trained VGG. This can be explained by the fact that the VGG network has been trained for semantic classification, so features of its intermediate layers manifest strong invariance on texture distortions on the same semantic regions, which helps infer more accurate reconstruction of the hole content.

3.5. Optimization

We solve the joint loss minimization problem by iterative gradient backpropagation with Limited-Memory BFGS. In the implementation, we first compute $f(x_0)$ by forward propagating the image x_0 with the hole R initialized with mean color value. After that, we use $f(x_0)$ to initialize x (i.e. $x_1 = f(x_0)$) and also as the content reference to evaluate E_c in each iteration of the optimization process. In our implementation, we fixed the content reference to $f(x_0)$ Each iteration involves a forward propagation of both the content network and the texture network to compute the loss, and a backward propagation to compute the gradient with respect to the image pixels. For the texture constraint, we average the gradient update on overlapping pixels between patches and put a slightly larger weight on patches around the hole border to ensure smooth appearance transitions around the hole boundaries.

Algorithm 1: High-Resolution Hole Filling with Multi-Scale Neural Patch Synthesis

```
Input: Image with mean value in the hole $x_0^S$,
       the content network $f$, the texture network $t$
1:  Compute $x_0^1$ by downsizing $x_0^S$
2:  Compute content reference $f(x_0^1)$
3:  for $s \in [1, 2, \ldots, S]$
4:      Update $\tilde{x}^s$ according to Equation 1
5:      Set optimization initialization $x_0^{s+1}$ by upsampling $\tilde{x}^s$
6:      Set the content reference at scale $s+1$ of $f^{s+1}(x_0)$
        by upsampling $f(x_0^1)$
7:  end for
8:  Return $\tilde{x}^S$
```

High-Resolution Image Inpainting Given a high-resolution image with a hole, we generate muti-scale input x_0^s, $s = 1, 2, \ldots, S$, where S is the number of scales. $s = 1$ is the coarsest scale and $s = S$ is the original resolution of the input image. We proceed with this optimization in an iterative multi-scale fashion. We first downsize the input to the coarsest scale x_0^1, compute content reference of $f(x_0^1)$. In practice, we double the width and height when upsampling to a new scale. In each scale, we update $\tilde{x}^s$ according to Equation 1, set optimization initialization x_0^{s+1} by upsampling $\tilde{x}^s$, and set the content reference at scale $s+1$ of $f^{s+1}(x_0)$ by upsampling $f(x_0^1)$. We therefore iteratively reach the high-resolution inpainting results. The algorithm is summarized in **Algorithm 1**.

4. Experiments

This section evaluates our proposed approach visually and quantitatively. We first introduce the datasets. We then compare our approach with other methods, and demonstrate its effectiveness in high-resolution image inpainting. At the end of this section we show its application in a real distractor removal scenario. Please refer to the supplementary material for more results and comparisons.

Datasets We evaluate the proposed approach on two different datasets: Paris StreetView [8] and ImageNet [34]. Labels or other information associated with these images are not used. The Paris StreetView contains 14,900 training images and 100 test images. ImageNet has 1,260,000 training images, and 200 test images that are randomly picked from the validation set. We also picked 20 images with distractors to test out our algorithm for real distractor removal scenarios.

Experimental Settings We compare our method with several baseline methods. First, we compared with results of Context Encoders trained with ℓ_2 loss. Second, we compare our method with the best results that Context Encoders have achieved using adversarial loss [32], which is the state-of-the-art in the area of image inpainting using deep learning. Finally, we compared with PatchMatch, which is one of the state of the art in the patch synthesis area. Results of these comparisons demonstrate the effectiveness of the proposed joint optimization framework.

While comparisons with baselines show the effectiveness of the overall joint optimization algorithm and the role of the texture network in joint optimization, we further analyze the role of the content network in the joint optimization by running experiments without the content network.

Finally, we show our results on high-resolution image inpainting, comparing to PatchMatch and Context Encoders (ℓ_2 and adversarial loss). In the high-resolution situation, we show significant improvement comparing to baseline approaches in terms of visual quality.

Quantitative Comparisons We first compare our method quantitatively with baseline methods on low-resolution images (128×128) on the Paris StreetView dataset. Results in Table 1 show that our method achieves highest numerical performance. We attribute this to the nature of our method – it is able to infer the correct structure of the image comparing with PatchMatch, and also capable of transferring texture detail from known regions comparing with the Context Encoders (Fig. 4). The results that we outperform PatchMatch show that the content network contribute to predict a reasonable structure. The results that we outperform Context Encoders demonstrate the effectiveness of the neural patch synthesis approach enforced by the texture network.

Method	Mean L1 Loss	Mean L2 Loss	PSNR
Context Encoders ℓ_2 loss	10.47%	2.41%	17.34 dB
PatchMatch	12.59%	3.14%	16.82 dB
Context Encoders (ℓ_2 + adversarial loss)	10.33%	2.35%	17.59 dB
Our Method	**10.01%**	**2.21%**	**18.00 dB**

Table 1. Numerical comparison on Paris StreetView dataset. Higher PSNR value is better. Note % in the Table is to facilitate reading.

The role of the content network in joint optimization On the one hand, we utilize the output of the content network as an initialization in the joint optimization. We have demonstrate its performance in the last section. In this section, we demonstrate the role of the content network serving as the content constraints in joint optimization by dropping the content constraint term and only using the texture contraint term in the joint optimization. We compare inpainting results with and without the content contraints. As shown in Fig. 6, without using the content term guiding the optimization, the structure of the inpainting results is corrupted. This explains the usefulness of the content term in the proposed joint optimization with Titan X.

High-Resolution image inpainting We demonstrate our result of high-resolution image (512×512) inpainting in Fig. 5 and Fig. 8 by comparing with PatchMatch and Context Encoders (ℓ_2 + adversarial loss)). As shown in the

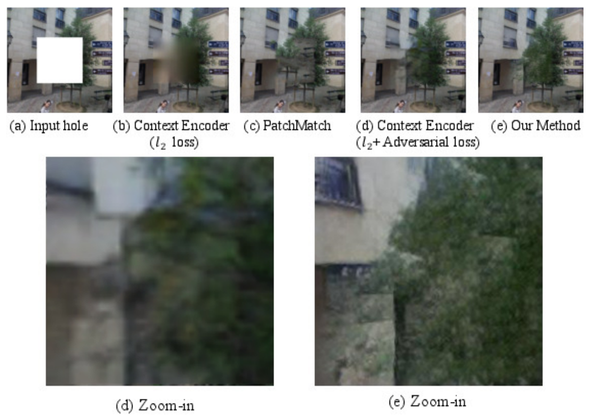
(a) Input hole
(b) Context Encoder (l_2 loss)
(c) PatchMatch
(d) Context Encoder (l_2+ Adversarial loss)
(e) Our Method
(d) Zoom-in
(e) Zoom-in

Figure 4. Comparison with Context Encoders (ℓ_2 loss), Context Encoders (adversarial loss), and PatchMatch. Our methods performs better than Context Encoders (using both ℓ_2 and adversarial loss) as it transfers the texture from boundaries to holes to create sharper result. Our method woks better than PatchMatch as we infer the correct structure.

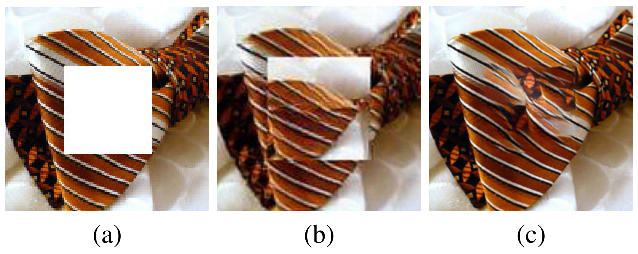

Figure 6. Evaluation of different components. (a) input image. (b) result without using content constraint. (c) our result.

figures, results of Context Encoders struggle to generate high-resolution images themselves due to network capacity and difficulty in training. Meanwhile, we found that PatchMatch generates visually pleasing results by propagating high-fidelity textures, but frequently fails to capture the high-level semantics and structures. Our multi-scale, iterative approach combines the advantage of both, producing visually coherent inpainting results with high-frequency details. With torch implementation, our algorithm takes roughly 3 min to fill in a 256×256 hole of a 512×512 image with GPU.

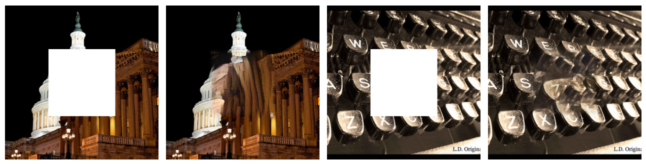

Figure 7. Failure cases of the proposed joint optimization approach.

Real-World Distractor Removal Scenario Finally, our algorithm is easily extended to handle arbitrary shape of holes. This is performed by estimating a bounding square around the arbitrary hole, filling mean-pixel value inside the hole, and forming an input by cropping the image to make the square bounding box at the center of the input and resizing the input to the content network input

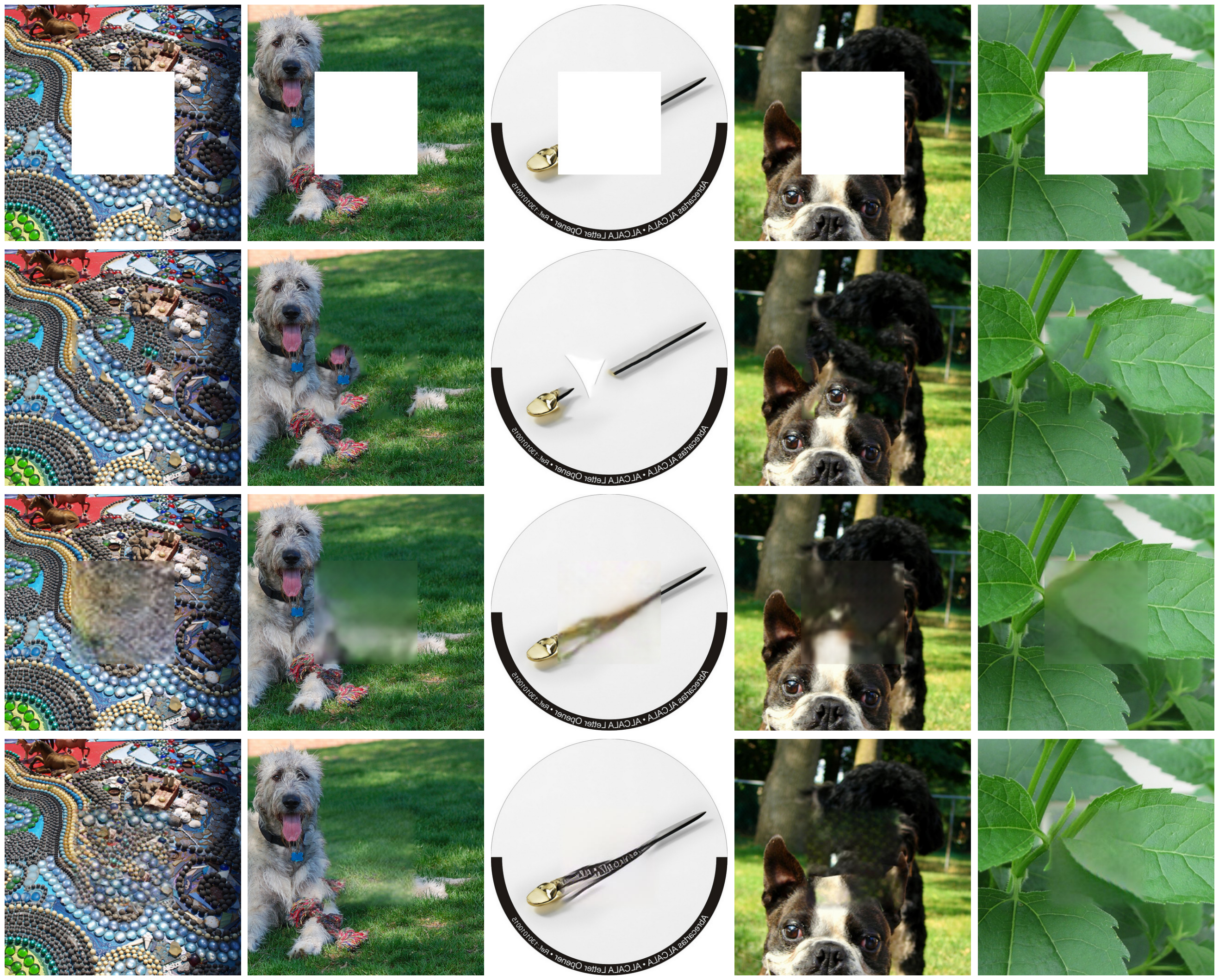
Figure 5. Visual comparisons of ImageNet result. From top to bottom: input image, PatchMatch, Context Encoders (ℓ_2 and adversarial loss), our result. All images are scaled from 512×512 to fit the page size.

size. We then ran forward propagation with the trained content network. In joint optimization, the texture network has no limits on the shape and location of holes in nature. This is an additional benefits by separating the content and texture terms. As Context Encoders is limited to square holes, we present our results in Fig. 9 by comparing with PatchMatch. As shown in the figure, the proposed joint optimization approach better predicts the structures, and deliver sharp and photo-realistic results.

5. Conclusion

We advanced the state of the art in semantic inpainting using neural patch synthesis. The insight is that the texture network is very powerful in generating high-frequency details while the content network gives strong prior about the semantics and global structure. There are cases when our approach introduces discontinuity and artifacts (Fig. 7) when the scene is complicated. In addition, the speed remains a bottleneck of our algorithm. We aim to address these issues in future work.

6. Acknowledgment

This research is supported in part by Adobe, Oculus & Facebook, Huawei, the Google Faculty Research Award, the Okawa Foundation Research Grant, the Office of Naval Research (ONR) / U.S. Navy, under award number N00014-15-1-2639, the Office of the Director of National Intelligence (ODNI) and Intelligence Advanced Research Projects Activity (IARPA), under contract number 2014-14071600010, and the U.S. Army Research Laboratory (ARL) under contract W911NF-14-D-0005. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, ARL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

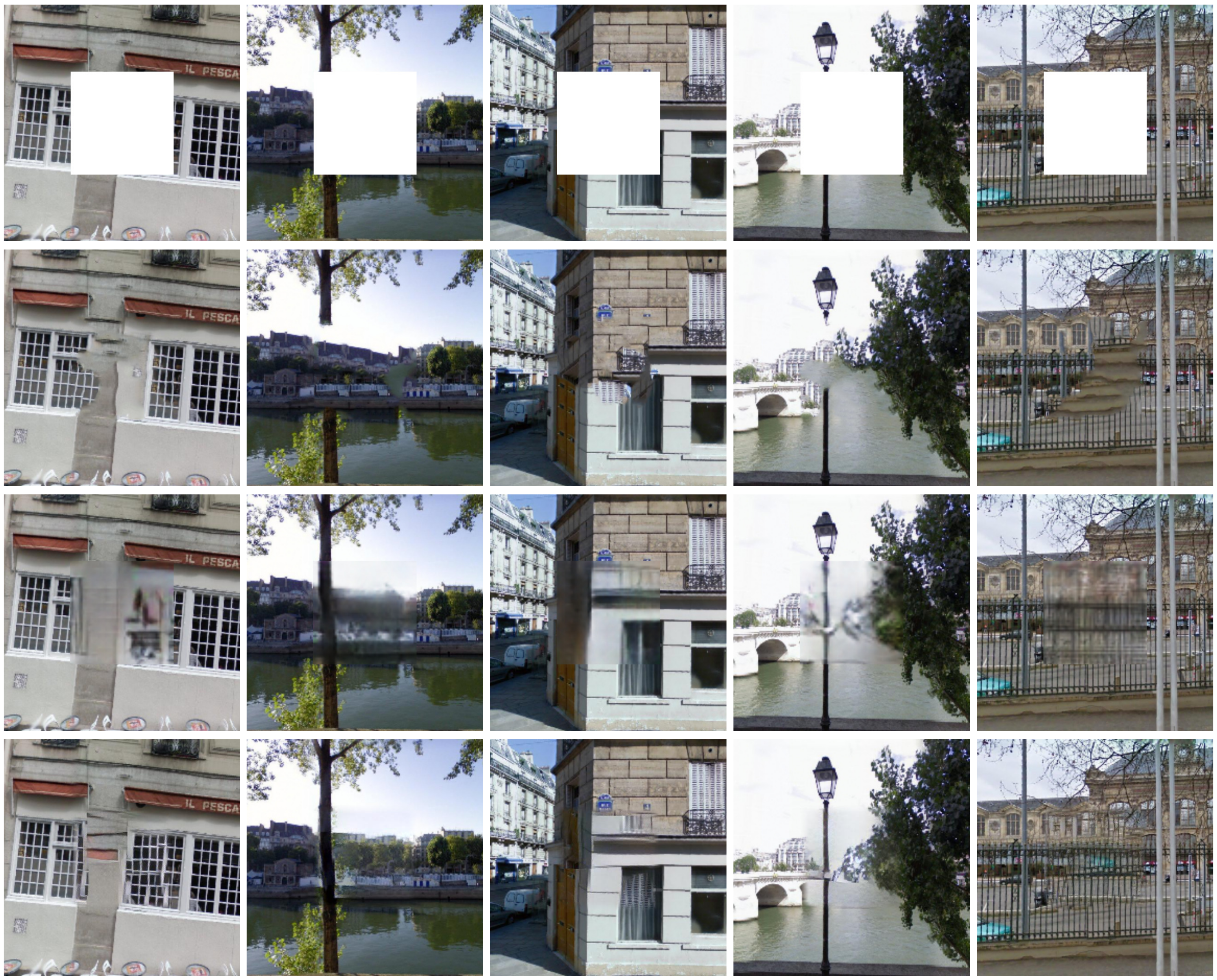
Figure 8. Visual comparisons of Paris Streetview result. From top to bottom: input image, PatchMatch, Context Encoders (ℓ_2 and adversarial loss), our result. All images are scaled from 512×512 to fit the page size.

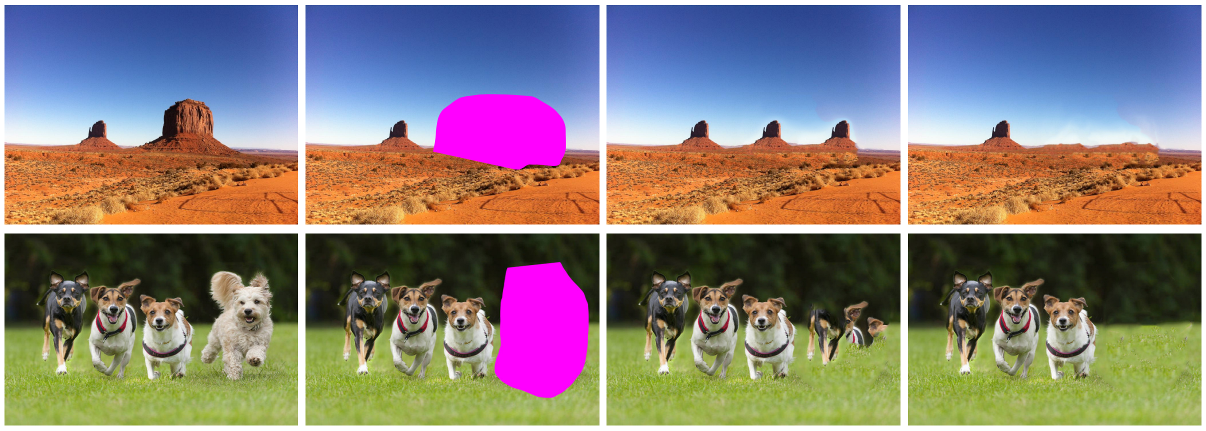
Figure 9. Arbitrary object removal. From left to right: input image, object mask, PatchMatch result, our result.

References

[1] https://research.adobe.com/project/content-aware-fill. 1

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *TOG*, 28(3):24:1–24:11, 2009. 1

[3] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artwork. In *arXiv:1603.01768v1*,

2016. 2, 3

[4] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 2

[5] D. Clevert, T. Unterhiner, Hochreiter, and S. Fast and accurate deepnetwork learning by exponential linear unites (ELUS). In *ICLR*, 2016. 4, 5

[6] A. Criminisi, P. Pérez, and K. Toyama. Object removal by exemplar-based inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–721 – II–728 vol.2, 2003. 1

[7] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *arXiv:1506.05751v1*, 2015. 2

[8] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes paris look like paris? *TOG*, 31(4), 2012. 5

[9] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *arXiv:1602.02644v1*, 2015. 2

[10] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *arXiv:1602.02644v1*, 2016. 4

[11] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, pages 1538–1546, 2015. 2

[12] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *TOG*, 22(3):303–312, 2003. 1

[13] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH*, pages 341–346, 2001. 1

[14] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, pages 1033–1038, 1999. 1

[15] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. In *arXiv:1508.06576v2*, 2015. 2, 3

[16] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In *NIPS*, 2015. 2, 3

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 2

[18] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *arXiv:1511.08446v2*, 2015. 2

[19] J. Hays and A. A. Efros. Scene completion using millions of photographs. *TOG*, 26(3), 2007. 2

[20] K. He, X. Y. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *arXiv:1409.1556v6*, 2016. 2

[21] D. Im, C. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. In *arXiv:1602.05110v1*, 2016. 2

[22] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. In *arXiv:1603.08155v1*, 2016. 2, 3, 4

[23] N. Komodakis. Image completion using global optimization. In *CVPR*, pages 442–452, 2006. 1

[24] N. Komodakis and G. Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *TIP*, 16(11):2649–2661, 2007. 1

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 2

[26] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *TOG*, 24(3):795–802, 2005. 1

[27] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. In *ACM SIGGRAPH*, pages 277–286, 2003. 1

[28] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *arXiv:1601.04589v1*, 2016. 2, 3, 4

[29] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 1–9, 2015. 2

[30] J. Long, E. Shelhamer, and T. Darrell. Learning deconvolution network for semantic segmentation. In *ICCV*, pages 1520–1528, 2015. 2

[31] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *arXiv:1601.06759v1*, 2016. 2

[32] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Effros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 1, 2, 3, 4, 5

[33] K. Ridgeway, J. Snell, B. Roads, R. Zemel, and M. Mozer. Learning to generate images with perceptual similarity metrics. In *arXiv:1511.06409v1*, 2015. 4

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 5

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 3, 5

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 2

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Very deep convolutional networks for large-scale image recognition. In *arXiv:1409.1556v6*, 2015. 2

[38] L. Theis and M. Bethge. Generative image modeling using spatial LSTMs. In *arXiv:1603.03417v1*, 2015. 2

[39] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *arXiv:1603.03417v1*, 2016. 2, 3

[40] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *CVPR*, pages 120–127, 2001. 1

[41] M. Wilczkowiak, G. Brostow, B. Tordoff, and R. Cipolla. Hole fill through photomontage. In *BMVC*, pages 492–501, 2005. 1

[42] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Is l_2 a good loss function for neural networks for image processing? In *arXiv:1511.08861v1*, 2015. 4