High-Resolution Image Inpainting with Iterative Confidence Feedback and Guided Upsampling

Yu Zeng[1], Zhe Lin[2], Jimei Yang[2], Jianming Zhang[2], Eli Shechtman[2], and Huchuan Lu[1]

[1] Dalian University of Technology, China
zengxianyu18@qq.com,lhchuan@dlut.edu.cn
[2] Adobe Research, USA
{zlin,jimyang,jianmzha,elishe}@adobe.com

Abstract. Existing image inpainting methods often produce artifacts when dealing with large holes in real applications. To address this challenge, we propose an iterative inpainting method with a feedback mechanism. Specifically, we introduce a deep generative model which not only outputs an inpainting result but also a corresponding confidence map. Using this map as feedback, it progressively fills the hole by trusting only high-confidence pixels inside the hole at each iteration and focuses on the remaining pixels in the next iteration. As it reuses partial predictions from the previous iterations as known pixels, this process gradually improves the result. In addition, we propose a guided upsampling network to enable generation of high-resolution inpainting results. We achieve this by extending the Contextual Attention module [39] to borrow high-resolution feature patches in the input image. Furthermore, to mimic real object removal scenarios, we collect a large object mask dataset and synthesize more realistic training data that better simulates user inputs. Experiments show that our method significantly outperforms existing methods in both quantitative and qualitative evaluations. More results and Web APP are available at https://zengxianyu.github.io/iic.

输出confidence和inpainting，值信任高置信度的像素，补全低置信度的像素

模仿

1 Introduction

Image inpainting is a task of reconstructing missing regions in an image. It is an important problem in computer vision and an essential functionality in many imaging and graphics applications, *e.g.* object removal, image restoration, manipulation, re-targeting, compositing, and image-based rendering [9,26,33]

Classical inpainting methods such as [17,25,9] typically rely on the principle of borrowing example patches from known regions or external database images and pasting them into the holes. These methods are quite effective for easy cases with small holes or uniform textured background. They can also handle high-resolution images efficiently. However, due to the lack of high-level structural understanding and ability to generate novel contents, they often fail to produce realistic results when the hole is large.

Deep learning has achieved great success on various dense prediction problems [14,15,13,12,42,43,45,44,36]. Recent research effort on inpainting has shifted

Fig. 1: Results of state-of-the-art methods on real object removal requests [7,4]
.

towards a data-driven learning-based approach [34,22,39,29,40]. These methods train a deep neural network to directly predict the inpainting result given a corrupted image and hole channel as input. The original images before corruption are used as the ground truth to train the network. To generate visually realistic results with sufficient texture details, they often use an adversarial loss based on GANs [20] in addition to a reconstruction loss. These deep generative models show significant improvements on filling holes in complex images but often produce visual artifacts, especially when the hole is large. For large holes, the reconstruction loss is less effective due to the increased ambiguity, leading to undesired predictions during testing as shown in Fig. 1.

In this paper, we aim to address the challenge of filling large holes in high resolution images for real image editing applications, *e.g.*, object removal. We observe that in the failure cases of existing approaches, despite the artifacts, there often exist sub-regions with good predictions. If we trust the good part and treat the remaining region as a new hole and run the model again, then the hole become progressively smaller and the model can produce better results. Inspired by this observation, we propose a novel iterative inpainting method with a feedback mechanism. Our method is based on a deep generative model which not only outputs an inpainting result but also a corresponding confidence map.

The model is encouraged to generate a confidence map that highlights pixels where the prediction error is likely small and can help overcome the prediction ambiguity. Using this confidence map as feedback, our model is trained to progressively fill the hole by trusting only high-confidence pixels inside the hole at each iteration and update the remaining pixels in the next iteration. By predicting what portion of the hole was filled successfully in the previous iteration and using those pixels as "known", our model can gradually improve the result when filling large holes. The proposed confidence prediction scheme is general and can be potentially attached to any deep generative inpainting model.

To generate high-quality inpainting results at high-resolution, we propose a guided inpainting upsampling network as a post-processing method. We achieve this by extending the Contextual Attention module [39] to borrow known high-resolution feature patches in the input image based on the patch similarities computed on the result for down-sampled input. The motivation is that it is easier to train a deep network to generate globally coherent structures for down-sampled inputs as effective receptive fields of neurons are larger; while the surrounding

regions of the high-resolution input can be used to enhance fine-grained texture details inside the hole. In this way, our method decouples high-level structural understanding and low-level texture reconstruction, and can produce results that are both semantically plausible and visually realistic at high resolution.

On the data side, previous methods construct training data by synthesizing square [34,22,39] or irregular holes [29,40]. However, in real applications for removing undesirable objects or scene segments, the hole-filling regions are more likely to be object-shaped. To mimic real use cases, we collect a large set of images with object-shaped holes. We synthesize our data to include both of the following two common situations: 1) holes overlap the foreground objects to simulate the distracting objects occluding the foreground, and 2) holes appear only in the background to simulate the case of unwanted objects occluded by salient foreground objects.

In summary, our contributions are three-fold:

- We address the challenge of completing large missing regions in images by proposing an iterative inpainting method with a confidence feedback loop.
- We propose a guided up-sampling network as a post processing step to enable generation of high-resolution inpainting results.
- We introduce a new procedure to synthesize training data for building deep generative models for real object removal applications.

2 Related work

Earlier image inpainting methods rely on existing content to fill the holes. Diffusion-based methods [8,10] propagate neighboring appearances to the target holes, but they often generate significant artifacts when the holes are large or texture variation is severe. Patch-based methods [17,25,9] search for most similar patches from valid regions to complete missing regions. Drori *et al.* [16] propose to iteratively fill missing regions from high to low confidence with similar patches. Although they also use a map to determine the region to fill in each iteration, the map is predefined based on spatial distances from unknown pixels to their closest valid pixels. The above methods use real image patches sampled from the input to fill the holes and can often generate high-quality results. However, they lack high-level structural understanding and cannot generate entirely new content that does not exist in the input image. Thus, their results may not be semantically consistent to regions surrounding the holes.

By learning from a large corpus of data, deep learning based inpainting methods can understand the semantic structure of the input image and hence can handle more difficult cases. To produce sharper results, these methods typically adopt adversarial training inspired by GANs [20]. Pathak *et al.* [34] made a first attempt to use a convolutional neural network (CNN) for hole filling. Iizuka *et al.* [22] use two discriminators for adversarial training to make the inpainted content both locally and globally consistent. Yu *et al.* [39] propose a deep generative model with contextual attention to explicitly utilize surrounding image features as references in the latent feature space. Zeng *et al.* [41] propose to use region affinity from a high-level feature map to guide the completion of missing regions

in the previous low-level feature map of a single input. Our upsampling network is similar in spirit of using coarse scale information to guide the generation of fine-grained details but different in architecture and functionality; it upsamples a low-resoluiotn results by filling the fine-grained details from the high-resolution input. Yang *et al.* [38] upsamples the results of a similar network with a neural patch search and vote approach followed by an optimization. Our method uses a related neural patch-vote approach but avoids the slow optimization. The above methods use square holes in their training data, which causes a bias to rectangular holes. To address this, Liu *et al.* [29] collect estimated occlusion/dis-occlusion masks between two consecutive frames of videos and use them to generate holes for training. The resulting masks are highly irregular and do not represent well holes typical to an image inpainting task. They also propose partial convolution layers to infer missing pixels conditioned only on valid pixels. Yu *et al.* [40] introduce free-form masks by simulating random strokes and generalizes partial convolution to gated convolution that learns to select features for each channel at each spatial location across all layers. Although these irregular holes lead to more diverse samples, they do not represent well real inpainting use-cases.

Most recently, a few works have been introduced to study progressive inpainting models. Zhang *et al.* [46] adopt a UNet generator with an LSTM in the bottleneck. It takes a sequence of inputs with large to small holes in the image center and generates a sequence of corresponding outputs. Guo *et al.* [21] propose to gradually fill a hole using consecutive residual blocks. They use partial convolutions in these blocks and update the hole mask according to the invalid pixels selected by partial convolutions. Oh *et al.* [32] propose an onion-peel network that progressively fills the hole from the hole boundary for video hole filling. All of the above methods fill the holes from the boundary to inner regions in a *predefined* sequence. Different from them, our proposed method jointly predicts a confidence map when generating an inpainting result. Using the confidence map from the previous iteration as feedback, it can automatically detect regions with bad fill to revise in following iterations. To our best knowledge, it is the first attempt to model confidence of predictions in inpainting and the first iterative inpainting method to fill holes with a confidence-driven feedback loop.

3 Approach

Our inpainting method consists of two models: an iterative inpainting model (Fig. 3 (a)) with confidence feedback and a guided upsampling network ((Fig. 3 (b)) that upsamples a low-resolution result by factor of 2 using the high resolution (HR) input as guidance. In this section, we first describe how we prepare data for building and evaluating our model, and then introduce the details of our iterative inpainting model and guided upsampling network.

3.1 Data generation

Previous approaches to image inpainting typically construct their training and testing data pairs by corrupting the original images with square-shaped [39,34,22] or highly irregular holes [29,40], as shown in the first two columns in Fig. 2. Images with holes are the input and the original images are the corresponding

Fig. 2: Comparison of input with holes. The first two columns are real object removal requests on the Web [5,2]. The second two are from PConv [29] and ContextAttention [39], respectively. The right two are our samples with object-shaped holes.

ground-truth. However, in real-world inpainting use cases such as distracting region removal, the regions users typically want to remove are objects or scene segments, which are rarely of square or highly irregular shapes, as shown in the middle two columns of Fig. 2.

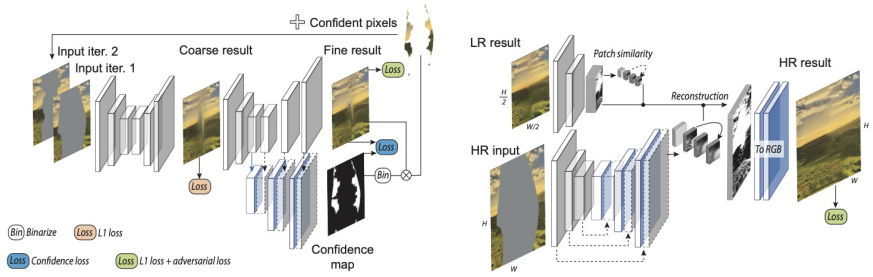
To mimic a more typical image editing scenario, in this work, we synthesize training samples with realistic holes. We collect 82,020 object masks from densely annotated segmentation datasets, including video segmentation [11], semantic segmentation [18], salient object segmentation [27,35,19], and human parsing [28]. The object masks from semantic segmentation datasets are from various classes and have different shapes and sizes. Salient object segmentation datasets contain large objects that form samples with large holes in our training data. We also use human parsing datasets to generate human-shaped masks as removing distracting people from photos is a common task.

Finally, we use a mix of random strokes [40] and the object masks as holes to create a more diverse training dataset and overcome a bias towards object shaped holes. The images for synthesizing training samples are from two sources: the Places2 [47] dataset and salient object segmentation dataset [37]. More specifically, we collect 61,525 images with pixel-level annotations of salient objects. We use 1,000 of them as testing samples, and the rest (60,525) are merged with Places2 for training and validation. For images in Places2, we sample randomly the location of the holes so they can appear in any region and may overlap with the main object. For images originating from salient object segmentation datasets, we subtract from the holes the intersection area with the salient objects. This is to simulate the case of removing distracting regions occluded by salient objects. As shown in the last two columns of Fig. 2, such generated samples with holes are more similar to real cases than those of previous approaches.

3.2 Inpainting model

We adopt a generative approach based on generative adversarial networks (GANs) [20]. Thus our model has a generator and a discriminator. Fig. 3 (a) illustrates an overall structure of the generator. It is a cascade of a coarse and a fine networks, similar to [39]. The coarse network takes an incomplete image and the corresponding hole mask as input to produce a coarse completed image. Then, the coarse result is passed to the fine network to generate a final completed image and a confidence map. The fine network has one encoder and two decoders: an image decoder that predicts the inpainting image result, and a confidence decoder that returns a corresponding confidence map of the predicted image. To

两个分支预测置信度和原图融合



(a) (b)

导向上采样模块

Fig. 3: The overall structure. (a) Iterative inpainting with confidence feedback. (b) Guided upsampling.

make confidence prediction aware of the full generation process, we let the confidence decoder take as input all the feature layers up to the bottleneck of the image decoder, as illustrated by the dashed blocks in Fig. 3 (a).

We use a PatchGAN discriminator [23] with spectral normalization as in [40] for adversarial training. It takes as input either the inpainted image or the ground-truth image and classifies each patch of the input image as real or fake. Its output is a score map rather than a single score, where each element corresponds to a local region of the input image covered by its receptive field.

Generative inpainting loss We train our model on the realistic hole dataset described in Sec. 3.1. We use L1 reconstruction loss on the coarse level. On the fine level, we use both L1 and a hinge adversarial loss with spectral normalization [30] applied on the discriminator. The loss for the discriminator D is:

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_{data}(x)}[\mathrm{ReLU}(\mathbb{1}-D(x))]+\mathbb{E}_{z\sim p_z(z)}[\mathrm{ReLU}(\mathbb{1}+D(G(z)\circ m+z))], \quad (1)$$

x是原图，预测->1，z是空洞图m中间是1，预测标签是-1

where x denotes the real (ground truth) image and z represents the incomplete image of which the pixels inside the hole are set to zero; m represents the hole mask, in which the pixels having value one belong to the hole; $G(\cdot)$ represents the image decoder; $\circ$ denotes element-wise multiplication; the inpainting result $G(z) \circ m + z$ is composed by the generated content $G(z)$ inside the hole and the original content z outside the hole. Let y denote the output of the image decoder, *i.e.* $y = G(z)$, then the loss for the inpainting result is:

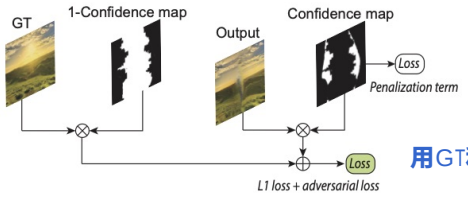
$$\mathcal{L}_G = \mathbb{E}_{z,x\sim p(z,x)}[L(y)], \text{where} L(y) = \mathrm{ReLU}(\mathbb{1} - D(y \circ m + z)) + \|y - x\|_1 . \quad (2)$$

GAN损失

Confidence prediction loss We make the confidence decoder detect good regions by using its output map as spatial attention on the predicted image when calculating the loss. Let c denote the confidence map, i.e., output of the confidence decoder of which each element is constrained to $[0, 1]$ by a sigmoid function, we define the following loss for the confidence decoder:

$$\mathcal{L}_C = \mathbb{E}_{z,x\sim p(z,x)}[L(y\circ c+x\circ(1-c))+\lambda(\left\|(\mathbb{1}-c)\circ m\right\|_1+\left\|(\mathbb{1}-c)\circ m\right\|_2)], \quad (3)$$

where λ is a hyperparameter controlling the size of the confident area. We set it to 0.1 in all evaluations but also provide sensitivity analysis of it in the experiments.



用GT和输出融合作监督？？不合理

Fig. 4: Loss for confidence prediction.

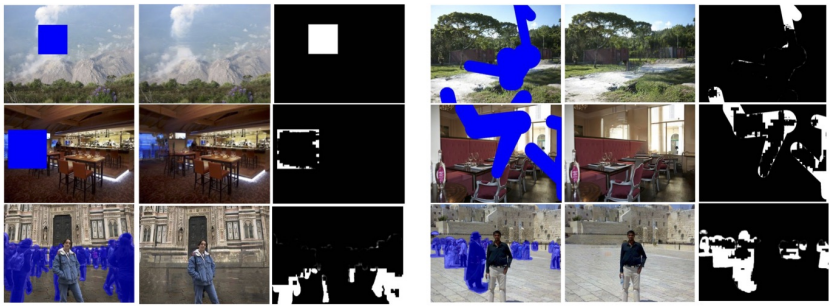


Fig. 5: Results and confidence maps. Blue masks indicate holes. Brighter pixels in confidence maps are of higher confidence. We use solid blue masks for synthetic samples and transparent ones for real object removal cases.

To minimize $\mathcal{L}_C$, the map c should highlight confidence regions, i.e. pixels contributing less to the overall loss. To prove this, we assume that: (1) $L(y)$ can be written as the summation over local pixel-wise losses, *i.e.* $L(y) = \sum_{i \in \mathcal{H}} l(y_i)$ where $l(y_i) \geq 0$ is an unknown local loss function, $\mathcal{H}$ is the index set of pixels inside the holes and y_i is a pixel of the generator output y, and (2) $l(x_i) = 0$ for every ground-truth pixel x_i. For simplification, we consider c as binary and let $\mathcal{C}$ to be the index set of non-zero elements of c, then $\mathcal{L}_C$ can be re-written as:

$$\begin{aligned} \mathcal{L}_C &= \mathbb{E}_{z,x\sim p(z,x)}[L(y \circ c + x \circ (1-c)) - \lambda \|c\|] \\ &= \mathbb{E}_{z,x\sim p(z,x)}[\sum_{i \in \mathcal{C}} l(y_i) + \sum_{i \in (\mathcal{H}-\mathcal{C})} l(x_i) - \lambda |\mathcal{C}|] \\ &= \mathbb{E}_{z,x\sim p(z,x)}[\sum_{i \in \mathcal{C}} l(y_i) - \lambda |\mathcal{C}|]; \end{aligned} \tag{4}$$

for a single sample y, the loss is the summation of local loss over $\mathcal{C}$. Therefore, the minimum is achieved when $\mathcal{C}$ covers the set of pixels with the smallest local loss values. Intuitively, the first term in Equation 3 encourages the confidence map to have high response where the loss $L(y)$ is small, as $\mathcal{L}_C$ is expected to be smaller by choosing low-loss area from the generator output $y \circ c$ and replacing high-loss are with ground-truth $x \circ (1-c)$; the second term penalizes a trivial solution of all-zero confidence maps by encouraging the confident region to cover as much of the missing region as possible. Fig. 4 illustrates how to compute $\mathcal{L}_C$.

Fig. 5 shows examples of inpainting results and the corresponding confidence maps. We can see that the confidence maps tend to highlight good regions of the result. As one may expect, confident regions are often located close to the hole

Algorithm 1: $G(\cdot)$: generator; $C(\cdot)$: confidence decoder

Input : Incomplete image z_1, hole mask m_1, the number of iterations T
Output: Completed image y_T

Set initial confidence map $c_0 = 0.5m_1$
for $t \in \{1, ..., T\}$ **do**
  Get confidence map $c_t = C(z_t) \circ m_t$
  Get mask of regions to update $u_t = \text{Binarize}(c_t - c_{t-1} \circ m_t)$
  Update mask $m_{t+1} = m_t - u_t$
  if $t = 1$ **then**
    Initialize completed image $y_1 = z_1 + G(z_1) \circ m_t$
  else
    Update competed image $y_t = G(z_t) \circ u_t + y_{t-1} \circ (1 - u_t)$
  end
  Update incomplete image $z_{t+1} = y_t \circ m_{t+1}$
end

Fig. 6: Inpainting results as iterations increase.

boundaries (e.g., the 2nd row). However, there are also other cases: 1) in easy cases like filling a hole in the sky, all generated pixels can have high confidence (e.g., 1st row, left); 2) flat regions tend to be more confident than highly-textured regions; 3) artifacts usually have low confidence (e.g., third row, right).

Iterative inpainting We can use the confidence decoder output to identify confident sub-regions in the inpainting result and run the inpainting model again and repeat the process. In each iteration, we set the confident pixels as "valid" pixels in the new input and set the remaining low-confidence regions as new holes for the next iteration to process. Overall, holes are shrinking as the iteration goes so that the network should be more certain about the generated result.

Algorithm 1 describes the iterative inpainting process in details. In the first iteration, we initialize the completed image by filling the whole missing region with the generated content and set the pixels of which confidence is below 0.5 as the missing regions for the second iteration. From the second iteration, a pixel is replaced by new generated one if its confidence increases over the previous iteration. When training, we iterate twice. We also fix the number of iterations during the testing. There is no convergence issue because our algorithm always keep the current-best complete prediction inside the hole at every iteration. Fig. 6 shows inpainting results in three consecutive iterations. As iteration goes, lines are connected and distorted area or artifacts are corrected.

Fig. 7: Effect of guided upsampling network. (a) results obtained by the iterative inpainting model on original size; (b) running the iterative model on half size and using guided upsampling network to upsample to the original size.

3.3 Guided upsampling

Our iterative inpainting model is trained on low-resolution (LR) (256×256) so it is not ideal to directly apply it to high-resolution (HR) inputs. To solve this issue, we propose a guided inpainting upsampling network to generate a HR inpainting result given a LR inpainting result. We propose a new architecture extending the contextual attention module [39] which can match and use feature patches from valid surrounding areas to help synthesize the hole pixels.

As illustrated in Fig. 3 (b), the proposed guided upsampling network consists of two shallow networks, one for learning patch similarity and the other for image reconstruction. Their feature maps are of different sizes, but we can split them into an equal number of patches using different patch sizes so that patches of the similarity network feature map have 1:1 correspondence to patches of the reconstruction network feature map which allows us to use shared indices to represent patches. Let $\mathcal{H}$ and $\mathcal{V}$ to be the index set of patches inside the holes and the valid patches, respectively. Valid patches are those with at least one pixel outside the holes, and others are taken as patches inside the hole. The patch similarity network calculates the cosine similarity s_{ij} between a pair of patch i, j. The reconstruction network is a shallow encoder-decoder network with skip connections from each layer of the encoder to the mirrored layer of the decoder. Before converting an HR feature map (of the HR input size) into an HR inpainting result, each feature patch inside the holes is replaced with a weighted sum of valid patches. Let ϕ_i to be an HR feature patch. The patch replacement in the HR feature maps can be summarized as follow:

$$\phi_i = \sum_{j \in \mathcal{V}} s'_{ij} \phi_j, \quad i \in \mathcal{H}, \tag{5}$$

where s'_{ij} is the softmax of s_{ij}:

$$s'_{ij} = \frac{\exp(s_{ij})}{\sum_{j \in \mathcal{V}} \exp(s_{ij})}. \tag{6}$$

Then the HR feature maps are transformed to an output image by two convolution layers ("ToRGB" in Fig. 3 (b)). The loss on the HR output is a combined L1 and adversarial loss, the same as in Equation 2. As mentioned earlier, we take the patches with at least one valid pixel as valid patches. For missing regions reconstructed by these partially valid patches, we simply take them as holes and run the previously described iterative inpainting model one more time. By separating high-level similarity learning and low-level texture reconstruction, the

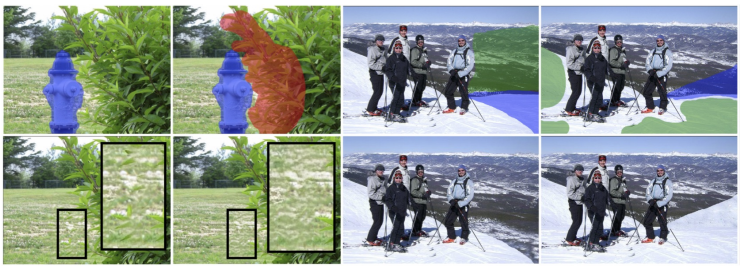


Fig. 8: Effect of user control. The blue masks indicate holes, red and green ones respectively indicate regions avoid or specified to be used for filling the holes. The second example shows that our method with user control can also be applied to interactive image layout manipulation.

proposed guided upsampling network can generate inpainting results that are both semantically reasonable and visually realistic, as shown in Fig. 7.

Another advantage of guided upsampling network is that it allows some user control over the results. As the contents inside the holes are constructed only using patches in $\mathcal{V}$, we can adjust the results by removing from or adding to $\mathcal{V}$. For example, as shown in Fig. 8, we can exclude the region that we do not want to copy, or specify a reference region used for filling the hole.

4 Experiments

4.1 Implementation details

We implement our method using Python and Pytorch. Detailed network architectures can be found in the supplementary material[3]. We train the models with Adam [24] optimizer; the learning rate set to 0.0001. The training batch size is 64. As the Places2 is much larger than the saliency dataset, we sample an equal number of images from Places2 dataset and saliency dataset to constitute each batch to prevent the model from ignoring the scarce samples. We use 256×256 patches for training the iterative inpainting model and train the guided upsampling network to upsample 256×256 results to 512×512. We randomly take 400 images from the training split of Places2 and the saliency dataset as validation samples and generate holes on them as described in Sec. 3.1. The model is trained until the PSNR on this validation set does not increase. The 1,000 testing images are kept unseen during training. When testing, the number of iterations for iterative inpainting is set to 4.

4.2 Comparison with state-of-the-art methods

We evaluate quantitative scores and visual quality of two variants of our method: *i.e.* **Ours***: the iterative inpainting model running on original input without guided upsampling and **Ours**: the iterative inpainting model running on $2\times$ downsampled input and then using the guided upsample model to obtain the

[3] https://zengxianyu.github.io/iic

Table 1: Quantitative evaluation and user preference of various methods. P.c.: preference count in user study.

Method	Object shaped holes			Irregular holes (Places2)			Square holes (Places2)			User study
	L1 Loss	PSNR	SSIM	L1 Loss	PSNR	SSIM	L1 Loss	PSNR	SSIM	P. c.
[9]	.0273	25.64	.8780	.0288	22.87	.8549	.0432	19.19	.7922	13
[22]	.0292	24.23	.8653	.0385	20.95	.8185	.0386	20.16	.7950	2
[40]	.0243	26.07	.8803	.0245	24.31	.8718	.0430	19.08	.7984	8
[31]	.0246	26.24	.8871	.0221	24.78	.8701	.0368	20.30	.8017	10
Ours*	.0194	28.20	.8985	.0203	25.43	.8828	.0361	20.21	.8130	62
Ours	.0205	27.67	.8949	0220	24.70	.8744	.0384	19.69	.8063	80

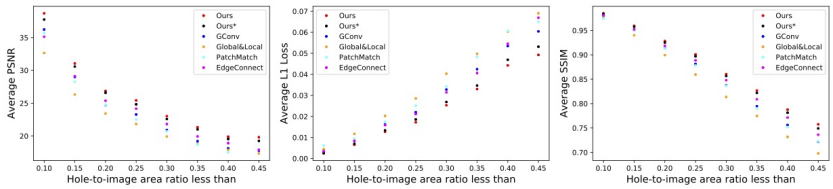


Fig. 9: Quantitative comparisons when hole size varies.

results of original size. We compare our methods with four state-of-the-art methods: Global&Local [22], PatchMatch [9], GConv [40] and EdgeConnect [31]. Comparison with more methods can be found in supplementary material.

Quantitative evaluation We evaluate two variants of our method and state-of-the-art methods on the test set of 1,000 images with object shaped holes. These images are of various size, from short side 256 to long side 1024. For random 500 images of them, we exclude salient objects from holes to simulate the case of distracting objects behind the main objects. For the rest 500 images the holes are placed randomly. For fair comparisons with previous methods, we also evaluate on the standard Places2 validation set resized to 256×256 with 128×128 center square holes as in most previous methods *e.g.* [22,39,41], and irregular holes as in [40,37]. We use L1 loss, PSNR, and SSIM as they are most commonly used metrics in image inpainting. Tab. 1 shows quantitative comparisons of our method with state-of-the-art methods. Both variants of our method compare favourably against previous methods. Without guided upsampling and running on original resolution, **Ours*** model tends to generate smoother results, which are favored by these scores at per-pixel basis. To validate superiority of our method on filling large holes, we show scores measured on images with holes of different sizes in Fig. 9. The X-axis represents the range of hole-to-image area ratio and Y-axis represents average L1 loss, SSIM and PSNR over all samples of which the hole-to-image area ratio are in the corresponding range. For example, the first column is averaged over samples whose hole-to-image ratio is less than 0.1 and the second column is averaged over those greater than 0.1 but less than 0.15. For small holes, all methods perform almost equally well. It is increasingly more difficult to fill holes when their size grow. So the SSIM, PSNR of all methods

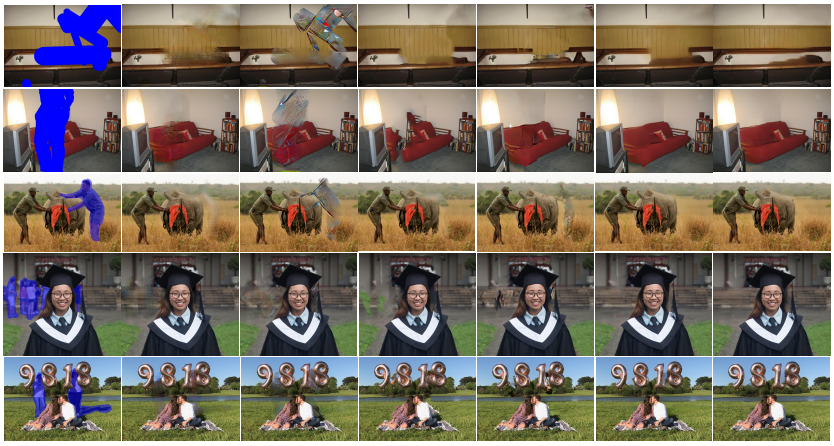


Fig. 10: Visual comparison on mid-resolution images. Zoom-in to see the details. Images are compressed due to limited submission file size. More results can be found in the supplementary material.

decrease and L1 loss increases as the hole-to-image ratio increases. When it comes to larger holes, our method performs better.

Visual quality Fig. 10 shows visual comparisons of two variants of our method and state-of-the-art methods on synthetic samples and real object removal tasks [1,6,3]. As shown in the figure, existing deep learning based methods do not work well in real requests. They often generate artifacts removing a large object. PatchMatch can generate clear texture, however, since it does not have a semantic understanding of input images, its results are not semantically reasonable. Both **Ours*** and **Ours** can provide reasonable alternatives for the region to remove. **Ours*** tends to generate smoother results. In comparison, by reconstructing LR results using patches from HR inputs, **Ours** method is better at keeping fine-grained details. Its results are similar to PatchMatch in terms of texture but more reasonable in terms of structure.

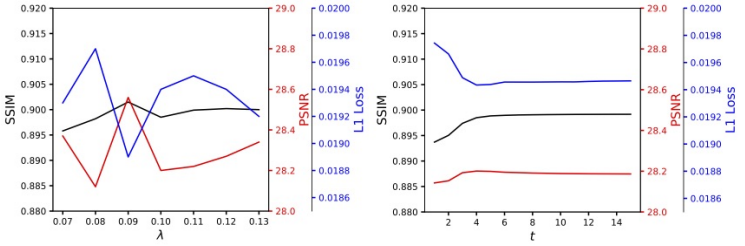
To evaluate visual quality of our method, we conduct a user study on 25 real object removal cases collected from object removal requests on the Web. All images are resized to make the short side equal to 512. Each input image with a marked region to remove and the results of different methods are shown in random order to 11 users and we ask them to select a single best result. Each combination of input and results are shown twice, and a valid vote is counted only when a user selects the same result twice. Finally we collect 175 valid votes. The user study results are shown in Tab. 1. Both variants of our method are preferred more than previous methods. **Ours** model with guided upsampling tends to generate results that are less smooth and with more clear texture, which are often favored by users.

4.3 Ablation study

First, to validate the proposed confidence prediction mechanism, we separately evaluate the results of high-confidence (> 0.5) and low-confidence (≤ 0.5) regions inside the hole. The results are in the bottom two rows of Tab. 2, which

Table 2: Effect of each component. IT: iterative inpainting; CF: confidence feedback; RT: realistic training data as described in 3.1; GU: guided upsampling. *PC* represents preference counts in user study. Time is measured in seconds on 512×512 input. *User ctrl.* indicates whether a method allows user control.

IT	CF	RT	GU	L1 Loss	PSNR	SSIM	PC	Time	User ctrl.
				.0205	27.79	.8903	-	.064	×
√				.0204	27.57	.8925	-	.301	×
√	√			.0200	28.06	.8952	-	.323	×
√	√	√		.0194	28.20	.8985	62	.323	×
√	√	√	√	.0205	27.67	.8949	80	.182	√
Confidence> 0.5				.0033	36.38	.981	-	-	-
Confidence≤ 0.5				.0165	30.00	.923	-	-	-



(a) (b)

Fig. 11: (a) Sensitivity analysis of λ. (b) Effect of increasing test iterations.

indicates that the prediction in high-confidence regions are significantly better than low-confidence regions. We show the effect of realistic training data, iterative inpainting, and guided upsample model in the first to the third rows of Tab. 2. The first row corresponds to our baseline model without confidence decoder trained on the Places2 dataset using irregular and square holes. The second row shows the effect of conducting progressive inpainting (IT) in a predefined boundary-to-center manner. For this setting, we evenly split each hole into four parts based on distance transformation and run the baseline model four times for each input. Each time we fill the part closest to the hole boundaries and update the hole mask accordingly. The third row shows the iterative inpainting method with confidence feedback (CF) trained on Places2 using irregular and square holes. By predicting confidence map, it can automatically correct wrong inpainted pixels and gradually improve the results, which yields better performance than predefined progressive inpainting in terms of quantitative scores. The sixth row corresponds to **Ours*** model described in previous sections. The comparison between the third and the fourth row shows the effect of including realistic training (RT) samples. All the variants discussed above output results of the same size as the input. So when evaluating these models, we give them the original input and do not apply post processing on their results. To analyze the effect of guide upsampling, we first run the proposed iterative inpainting method on 2× downsampled input images, and then upsample the results to

Fig. 12: Inpainting results on 1024×1024 images. Zoom-in to see the details. Images are compressed due to limited submission file size.

the original resolution using guided upsampling. This corresponds to **Ours** described previously, and the effect is shown in the fifth row of Tab. 2. By running the iterative process on the downsampled input, it significantly cuts down the overall run-time.

The sensitivity analysis of λ (Eqn. 3) is shown in Fig. 11 (a). The performance is not very sensitive to λ when it changes in a small range. For example, for $\lambda \in [0.7, 0.13]$, PSNR is in $[28.1, 28.5]$. Fig. 11 (b) shows the effect of increasing test iterations. More test iterations generally lead to better scores, especially in the first four iterations. We fix the number of iterations to 4 during testing.

Fig. 12 shows inpainting results of **Ours*** and **Ours** on input images of size 1024×1024. As the guided upsampling network lifts the LR result to HR by utilizing features from a HR input, it brings the details from existing contents to generated contents, resulting in a more visually pleasant HR output. It also can be reflected from the user study, in which **Ours** is preferred by users more frequently. However, reconstructing with existing patches is a constraint on generation, making it less free and difficult to restore exactly the original content in the missing region. As a result, **Ours** has lower quantitative scores than **Ours*** as shown in the last row of Tab. 2.

5 Conclusion

We propose a high-resolution image inpainting method for large object removal. Our model predicts the inpainting result as well as its confidence map, which is used to revise unsatisfactory regions in an iterative manner. To improve visual quality for high-res inputs, we first obtain a low-res result and then reconstruct it using high-res neural patches. Furthermore, we collect a large object masks dataset and synthesize realistic training samples that simulate realistic user inputs. Experiments show that our method outperforms existing methods and achieves better visual quality.

Acknowledgements

The paper is supported in part by National Key R&D Program of China under Grant No. 2018AAA0102001, National Natural Science Foundation of China under grant No. 61725202, U1903215, 61829102, 91538201, 61771088,61751212, Fundamental Research Funds for the Central Universities under Grant No. DUT19GJ201, Dalian Innovation leaders support Plan under Grant No. 2018RD07.

References

1. can someone please remove my co-worker on the right and just leave the rhino and my friend on the left? https://www.reddit.com/r/PhotoshopRequest/comments/82v6x1/specific_can_someone_please_remove_my_coworker_on/
2. Can someone please remove the backpack and 'lead' from my sons back? would love to have this picture of my kids without it! https://www.reddit.com/r/PhotoshopRequest/comments/6szh1i/specific_can_someone_please_remove_the_backpack/
3. can someone please remove the people holding the balloons and their shadows from this engagement photo? https://www.reddit.com/r/PhotoshopRequest/comments/8d12tw/specific_can_someone_please_remove_the_people/
4. Can someone remove the woman in purple please? will give reddit gold! https://www.reddit.com/r/PhotoshopRequest/comments/6ddjg3/paid_specific_can_someone_remove_the_woman_in/
5. Could someone help me remove background people - especially the guys head? will venmo $5. https://www.reddit.com/r/PhotoshopRequest/comments/b2y0o5/specific_paid_could_someone_help_me_remove/
6. Could someone please remove the people in the background if at all possible! https://www.reddit.com/r/PhotoshopRequest/comments/6f0g4k/specific_could_someone_please_remove_the_people/
7. If possible, can anyone help me remove the people on the side, esp the people facing towards the camera :) thank you. https://www.reddit.com/r/PhotoshopRequest/comments/anizco/specific_if_possible_can_anyone_help_me_remove/
8. Ballester, C., Bertalmio, M., Caselles, V., Sapiro, G., Verdera, J.: Filling-in by joint interpolation of vector fields and gray levels. IEEE Transactions on Image Processing **10**(8), 1200–1211 (2001)
9. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing **28**(3), 24 (2009)
10. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: The 27th Annual Conference on Computer Graphics and Interactive Techniques (2000)
11. Caelles, S., Pont-Tuset, J., Perazzi, F., Montes, A., Maninis, K.K., Van Gool, L.: The 2019 davis challenge on vos: Unsupervised multi-object segmentation. arXiv:1905.00737 (2019)
12. Ding, H., Cohen, S., Price, B., Jiang, X.: Phraseclick: Toward achieving flexible interactive segmentation by phrase and click. In: European conference on computer vision. Springer (2020)
13. Ding, H., Jiang, X., Liu, A.Q., Thalmann, N.M., Wang, G.: Boundary-aware feature propagation for scene segmentation. In: IEEE International Conference on Computer Vision (2019)
14. Ding, H., Jiang, X., Shuai, B., Liu, A.Q., Wang, G.: Context contrasted feature and gated multi-scale aggregation for scene segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
15. Ding, H., Jiang, X., Shuai, B., Liu, A.Q., Wang, G.: Semantic correlation promoted shape-variant context for segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
16. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion **22**(3), 303–312 (2003)
17. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: The 28th Annual Conference on Computer Graphics and Interactive Techniques. pp. 341–346. ACM (2001)

18. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88**(2), 303–338 (2010)
19. Fan, D.P., Cheng, M.M., Liu, J.J., Gao, S.H., Hou, Q., Borji, A.: Salient objects in clutter: Bringing salient object detection to the foreground. In: European Conference on Computer Vision (2018)
20. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (2014)
21. Guo, Z., Chen, Z., Yu, T., Chen, J., Liu, S.: Progressive image inpainting with full-resolution residual network. In: Proceedings of the 27th ACM International Conference on Multimedia. ACM (2019)
22. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (ToG) **36**(4), 107 (2017)
23. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1125–1134 (2017)
24. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
25. Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis **24**(3), 795–802 (2005)
26. Levin, A., Zomet, A., Peleg, S., Weiss, Y.: Seamless image stitching in the gradient domain. In: European Conference on Computer Vision. pp. 377–389. Springer (2004)
27. Li, G., Yu, Y.: Deep contrast learning for salient object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
28. Liang, X., Liu, S., Shen, X., Yang, J., Liu, L., Dong, J., Lin, L., Yan, S.: Deep human parsing with active template regression. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(12), 2402–2414 (2015)
29. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: European Conference on Computer Vision (2018)
30. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
31. Nazeri, K., Ng, E., Joseph, T., Qureshi, F., Ebrahimi, M.: Edgeconnect: Generative image inpainting with adversarial edge learning. arXiv preprint arXiv:1901.00212 (2019)
32. Oh, S.W., Lee, S., Lee, J.Y., Kim, S.J.: Onion-peel networks for deep video completion. In: IEEE International Conference on Computer Vision (2019)
33. Park, E., Yang, J., Yumer, E., Ceylan, D., Berg, A.C.: Transformation-grounded image generation network for novel 3d view synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3500–3509 (2017)
34. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
35. Wang, J., Jiang, H., Yuan, Z., Cheng, M.M., Hu, X., Zheng, N.: Salient object detection: A discriminative regional feature integration approach. International Journal of Computer Vision **123**(2), 251–268 (2017)
36. Wang, L., Zhang, J., Wang, O., Lin, Z., Lu, H.: Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2020)

37. Xiong, W., Yu, J., Lin, Z., Yang, J., Lu, X., Barnes, C., Luo, J.: Foreground-aware image inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
38. Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
39. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5505–5514 (2018)
40. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: IEEE International Conference on Computer Vision (2019)
41. Zeng, Y., Fu, J., Chao, H., Guo, B.: Learning pyramid-context encoder network for high-quality image inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1486–1494 (2019)
42. Zeng, Y., Lin, Z., Yang, J., Zhang, J., Shechtman, E., Lu, H.: High-resolution image inpainting with iterative confidence feedback and guided upsampling. In: European Conference on Computer Vision. Springer (2020)
43. Zeng, Y., Lu, H., Zhang, L., Feng, M., Borji, A.: Learning to promote saliency detectors. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
44. Zeng, Y., Zhuge, Y., Lu, H., Zhang, L.: Joint learning of saliency detection and weakly supervised semantic segmentation. In: IEEE International Conference on Computer Vision (2019)
45. Zeng, Y., Zhuge, Y., Lu, H., Zhang, L., Qian, M., Yu, Y.: Multi-source weak supervision for saliency detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2019)
46. Zhang, H., Hu, Z., Luo, C., Zuo, W., Wang, M.: Semantic image inpainting with progressive generative networks. In: 2018 ACM Multimedia Conference on Multimedia Conference. pp. 1939–1947. ACM (2018)
47. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(6), 1452–1464 (2017)