Image Extension with Contour Patch Matching and Generative Adversarial Network

Xiaochen Zhou
zhouxiaochen@wustl.edu

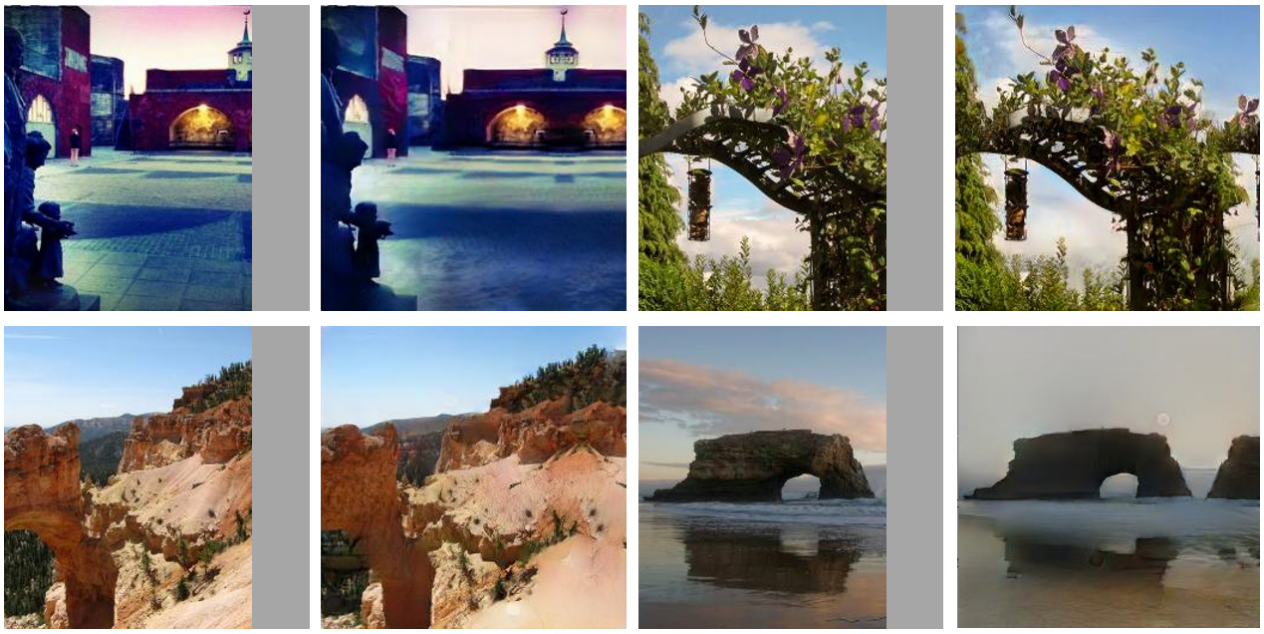


Figure 1: Image extension results by our approach. The left image is the source image required to be extended and the right image is the extended image.

Abstract

Image completion is a useful tool for image editing. While image inpainting is a top research project in the past few dacades and have obtain some state-of-the-art achievements, directly applying these approaches to image extension will not provide promising results. In this work, we introduce a novel pipeline for image extension built on a patch-based methods and learning-based methods. Our pipeline achieves strong results on image extension with satisfying texture details and semantical rationality. We also show the model's ability of panaroma generation.

1 Introduction

Image completion is a technique that estimates a complete image from an input with missing pixels. This technique can be used as automatical image editing tools such as image inpainting, image extension, object removal and etc. Althrough many approaches have been proposed for image completion, such as patch-based image synthesis [1] [2] [3] and learning-based approaches, it remains a challenge not only owing to the high resolution demand but also the necessity for both texture and sematic rationality. On this observation, we come out with a novel work based on patch-based and learning-based methods to deal with image extension, a harder task in image completion.

Our approach builds on image quilting [3]. Image quilting extends the image by searching and selecting patches from the source image, placing the selected patch on the source image with overlapping areas, and optimizing the stitching boundaries to make the texture changes smoothly. Although Image quilting approach obtain charming results on source images with repeating textures, the performance on complicated images like photograph is not convincing. Inspired by sparse smart contour [9], where they extract the sparse contours from the source image, save image information on the contours and

use these contours to reconstruct the image, our approach extends the images on the contour domain, and then reconstruct the extended contour into RGB images.

We first extract the contour of the source image and save the gradient information on the contour. To extend the contour image, we select both contour patches and RGB source patches with minimum errors on the overlapping area as the reference, place the patch on the source image and find a minimum error cut path through the error matrix by dynamic programming. Then we build a generative adversarial network for image reconstruction. We build a coarse-to-fine unet architecture as the generator, which takes the extended contour with image information as the input, and generate the RGB image.

In summary, in this paper we present:

- an approach to extend the contour image relied on reusing content in the source image.
- a neural network structure that reconstructs high-resolution RGB images.
- a novel pipeline for high-resolution image extension by traditional methods and deep learning.

2 Related Work

Methods for image completion can be divided into two categories: classical methods, which use non-parametric computer vision and texture synthesis approaches to address the problem, and learning-based methods, which deal with this task with parametric machine learning, generally in the form of deep convolutional neural networks.

Classical methods commonly use patch-based algorithms to complete the missing area. For image inpainting, patch match [1] helps to find matching patches from the source image to the target image. They introduces a rapid method to search and propagate the patches and build the correspondence. It is efficient for image completion, while the performance is not natural and satisfying enough. For image extension, photo uncrop [2] takes an photograph as an input, and selects a subset of Internet images taken near the input photograph, and then reprojects, and composites into a larger image around the input using the underlying scene geometry. However the system is complex and requires users interactions. Besides, the system and the performance are depended on the related dataset instead of the source image itself, which brings limitations for this approach. Image quilting [3] extends the image by searching and selecting patches from the source image, placing the selected patch on the source image with overlapping area, and optimizing the stitching boundaries by dynamic programming. Although the performance is outstanding on source image with repeating texture, results generated by image quilting are not satisfying.

In recent years, learning-based approaches have achieved great progress. The first significant learning-based approach for image inpainting was Context Encoder [4]. They trained an encoder-decoder network to fill in the squared blank area in an image, and used a combination of L2 regression on pixel values and an adversarial loss [6] as the supervised signal. [5] built up a architecture with a local-global discriminator, where the global sent the whole image as the input and the local discriminator picked the generated area for the missing part. Deep learning methods present strong ability for image inpainting task, while extending the image from one boundary obtains less information compared with image inpainting, which leads to image extension much more challenging and the approaches for image inpainting would not work well on image extension task. Also, deep learning approaches are often trained and used datasets on specific scene categories. They would not always do good job on all scene categories and all types of photos. Althrough [7] built an GAN architecture for image extension which generated promising extended results, they still take similar architecture with image inpainting taks. The keypoint was the new discriminator that they took advantage of from conditional projection discriminator [8], which may boost the performance of the image extension.

3 Proposed Approach

3.1 Overview

In our approach, we show a new pipeline that would extend the image easier and would be able to do well on most of scene categories and all types of photos. Our approach is built based on two methods: image quilting [3] and sparse smart contour [9]. We first extract the contour of the input image and extend the contour, then we reconstruct the contour into RGB image.

3.2 Image Contour Extension

3.2.1 Contour with Information

To extend the image, the first step is generating the contour with image information. We first extract the contour of the source image by canny edge detector [10], and compute the gradient of the image on X and Y axis. Let us denote c as contour and D_x and D_y as gradient on X and Y axis respectively. Then the contour with information is computed as:

$$C = c \times D_x \oplus c \times D_y$$

where C denotes contour with information, $\times$ denotes element-wise product and $\oplus$ denotes concatenating on

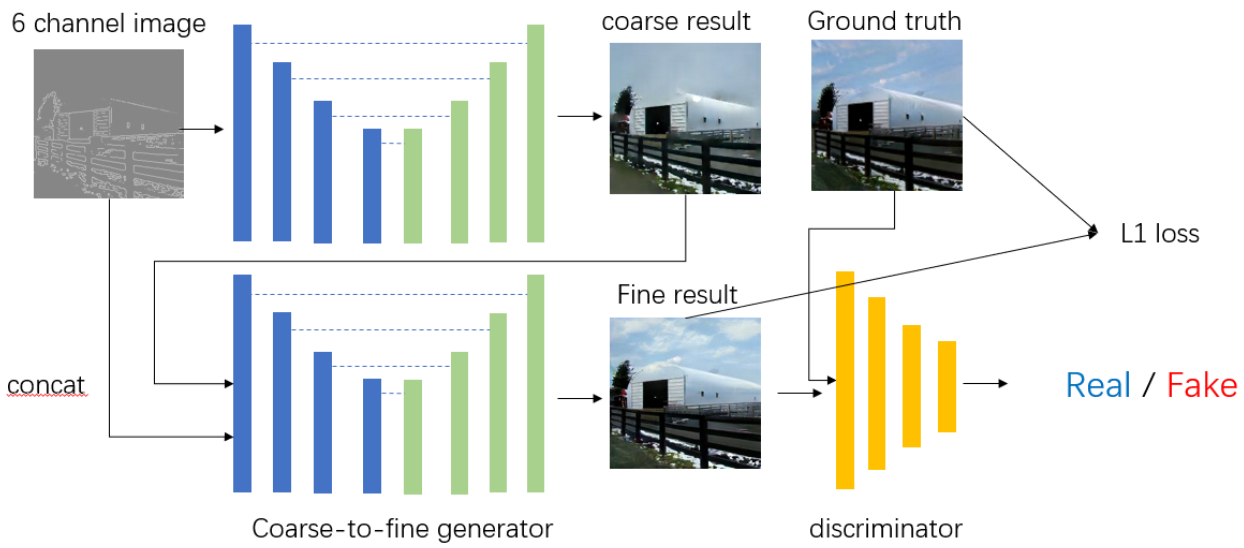


Figure 2: The architecture for the coarse-to-fine Unets

channel axis. In this case, C would shape like $[X, Y, 6]$. We will call this 6-channel contour.

3.2.2 Candidate Patch Pair Selection

In this section we will pick up a RGB patch and the related 6-channel patch as the candidate patch pair. We slice the source image and the contour image into small patches with overlapping area. The stride for slice window is 1. We call the patch required to be extended in the original image as target patch, and the patches selected from the sliced image as source patches. Let us denote l_{rgb} as the L2 distance of the overlapping area between the selected RGB patch and the target image patch, l_C as the L2 distance of selected 6-channel contour patch and 6-channel contour of the target patch. Besides, assuming that when we extend the image on one axis, the patches on the same coordinate with the source patch on the other axis could be more semantically reasonable, we denote l_{axis} as the L1 distance of coordinate distance. For instance, when we extend the target image on X axis, l_{axis} would be computed as $l_{axis} = ||Y_{target} - Y_{source}||$, where Y_{target} and Y_{source} denotes the Y coordinate for target patch and the source patch. Then the loss will be:

$$l_{total} = l_{rgb} + \alpha l_C + \gamma l_{axis}$$

where α and γ are the weights for l_C and l_{axis}. In this case, we will pick up the related patch pair with the lowest loss error as the candidate patches.

Computing L2 distance and L1 distance is time-consuming. If the source image is large and the amount of sliced patches is too high, the time cost for the approach will be unacceptable. In this case, we come out with a pre-selection method. We compute the average intensity of the overlapping areas for both source patches and target patches, and compute the average intensity difference as the loss. For loss function in contour domain, we compute the sum of the XOR of the the original 1-channel contours for the source patch and target contour patch. The bit-wise XOR operation can significantly accelerate the computation. The larger the sum is, the more the contour mismatching is from source contour and the target contour and the worse the matching would be. We will select the top K patches with the lowest pre-selection error, and find the candidate patch pair through them.

3.2.3 Minimum Error Boundary Cut

We want to find the cut on the overlapping area between the candidate patch and the source patch. This can be done by dynamic programming.

Suppose that the source patch will be stitched on the down side of the target patch, we need to find a horizonal cut path to split the overlapping area. We compute and sum up the L2 distance of the overlapping area in both RGB patches and 6-channel patches and add them together. We denote this distance as e. Then we traverse $e(i = 1..N)$ and compute the cumulative minimum error E for all paths. We also build a matrix P to save the address of previous node:

$$E_{i,j} = e_{i,j} + min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$$

$$P_{i,j} = \underset{y}{\text{argmin}}(E_{i-1,y}), s.t. y \in \{-1, 0, 1\}$$

The minimum value of the last column of E will indicate the end point of the path and we can trace back and find

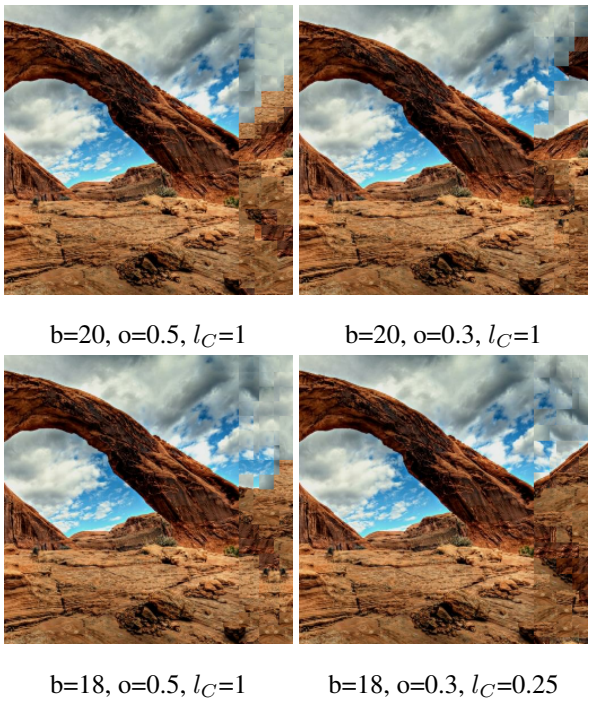
the path with the minimum error. The approach for vertical cut path is similar. When we have two overlapping areas, we split the areas by two paths meet in the middle. The RGB source patch and 6-channel source patch will be stitched to the target patch with the guidance of the minimum error cut path in both RGB image domain and contour domain. We recursively apply this approach until the image is extended to the goal size.

3.3 Coarse-to-fine Unet

The architecture for image reconstruction is consisted of two unets [11], a convolutional encoder and decoder with skip connections between layers of the encoder and decoder. The architecture is shown in Figure 2.

The first network reconstruct the overall structure and the general color of the output. The network is trained with an L1 pixel loss between the reconstruct coarse image and the ground-truth image, which leads to the low-frequencies of the output. The second Unet introduce details and textures to the image. The input of the second network is the combination of the coarse output from the first Unet and 6-channel contour, and trained by L1 pixel loss and an adversarial loss, which will bring more textures and details to the reconstructed image. When the second Unet is under training, the weights in the first Unet are fixed. To increase the stablility of GAN training, we adopt progressive GAN [12] in the second Unet where the layers for both generator and discriminator are added gradually to the architecture. The second Unet generates low resolution images at the beginning where only few layers are in the generator and discriminator. When current layers have been trained well, next new layer will be added, and the quality and resolution of the output will be sharpened.

The inputs for the first training are 6-channel contour images generated from the training dataset, not the extended 6-channel contour images we generate in the previous section. When the network is well trained on training dataset, we will finetune the network with the combination of training contour images and our extended contour images. We call the 6-channel contours from the training dataset "True contours" and the extended contours "fake contours". $L1$ pixel loss is still computed by ground truth and images generated through real contours, while the discriminator will distinguish whether the images are real or fake among ground truth images, images from true contours and images generated from fake contours.



b=20, o=0.5, l_C=1 b=20, o=0.3, l_C=1

b=18, o=0.5, l_C=1 b=18, o=0.3, l_C=0.25

Figure 3: Different parameters for image extension. b denotes size of patches, o denotes the ratio of the overlapping area for patches, and l_C denotes the weight for 6-channel contour loss

4 Experiments

4.1 Parameters for Candidate Patch Selection

Parameter settings for the candidate patch selection can affect the results of the 6-channel contour image extension. For the selection, we would consider the size of the patch, the ratio of the overlapping area and the whole patch and the weight for 6-channel contour loss l_C. The following are some samples of extended images. We show the extended RGB image instead of 6-channel contour for a clear visualization. In our experiments we set the patch size to 18, overlapping ratio to 0.3, weight for l_C to 0.25. The weight for coordinate distance l_{axis} is 0.01.

4.2 Results for extension

In this section, we present the results for the reconstruction. Figure 4 shows the reconstruction of the images sized as 256×256.The first column shows the source image. The middle two images show the extended 6-channel contour image and the RGB images in image contour extension process. The last column is the reconstructed image. The contour provide the structure information. Then GAN generates the rational overall structure and sharp texture

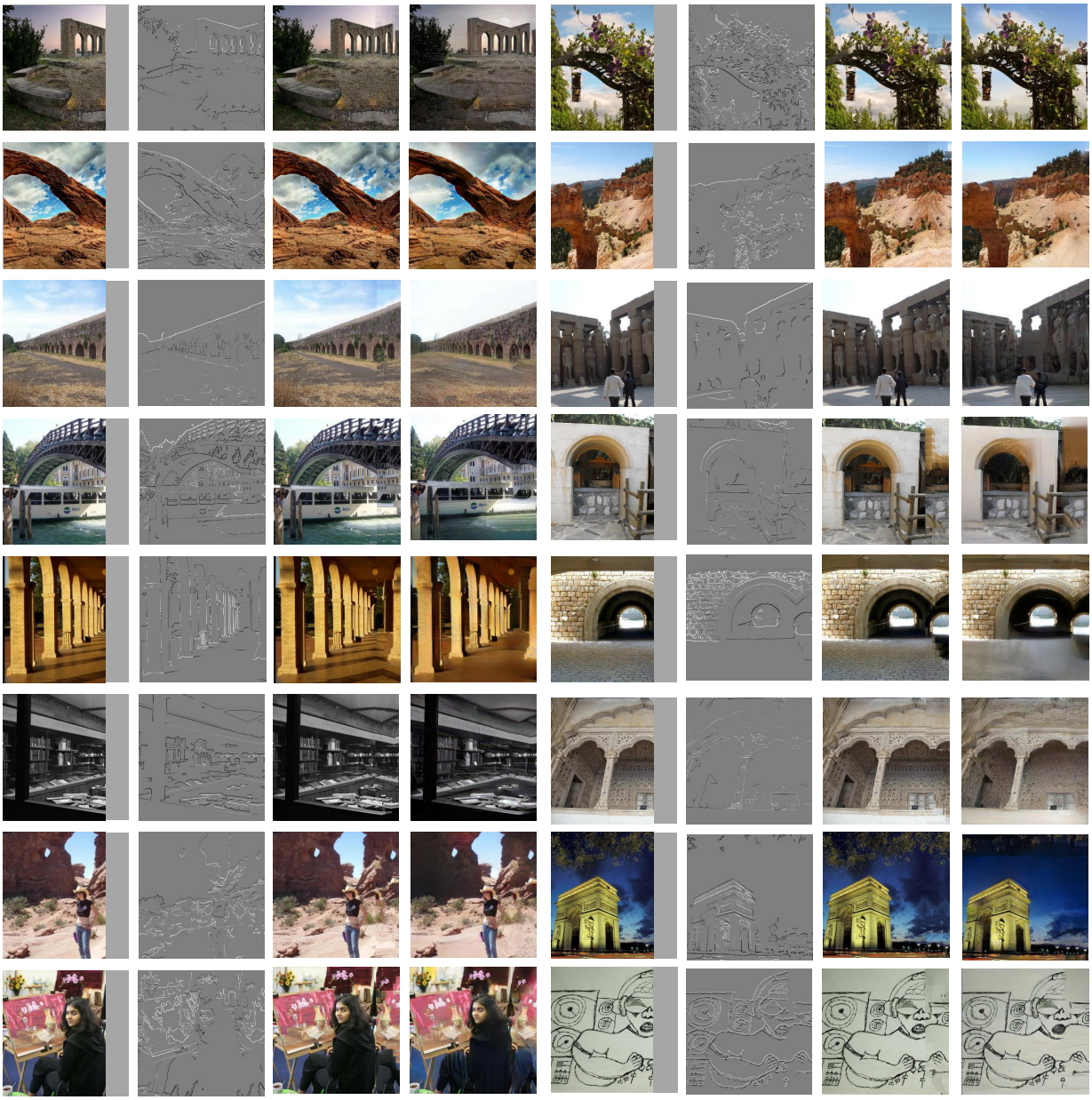


Figure 4: Results for image extension. The first column shows the source image. The middle two images show the extended 6-channel contour image and the RGB images with patches copied and pasted in image contour extension process. The last one is the reconstructed image.

details for the whole image.

We also explore our model's ability of panorama extension. Figure 6 shows the panorama extension results. To generate panorama images, we continue extension based on the generated images. We use the extended 6-channel from last extending iteration instead of extract 6-channel contour images based on the generated image, and use original 6-channel and RGB patches for next extension. This helps the 6-channel contour to save more information from the source image. For each iteration, we will extend 64 units on one axis, such as from 256×256 to 256×320. We recursively apply our model and then generate the panorama results.

5 Conclusion

In this work, we build a novel pipeline to extend the contour image relied on reusing content in the source image.

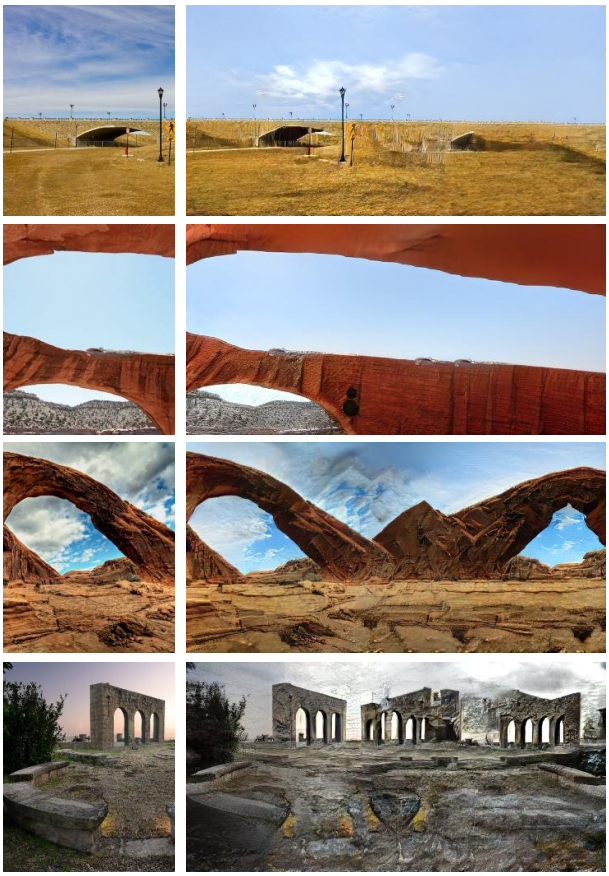


Figure 5: Results for panaroma extension

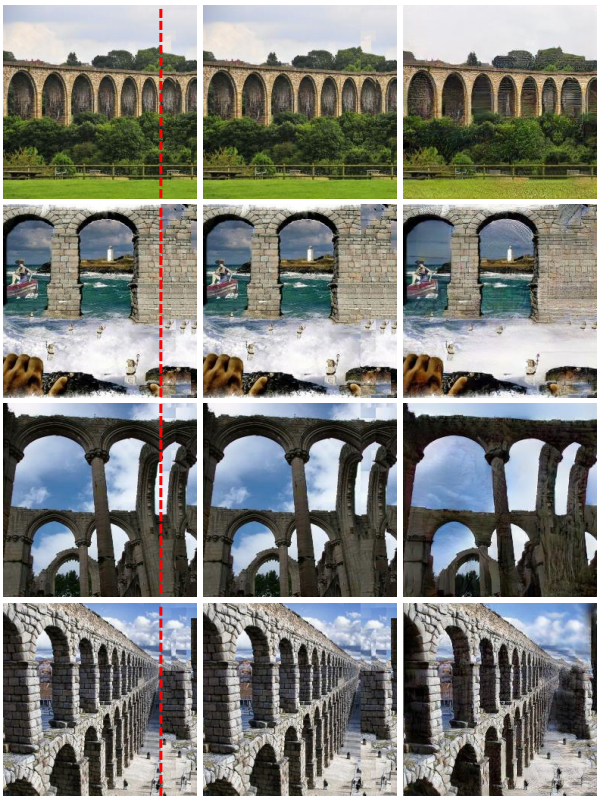


Figure 6: Comparison between image extended with copying and pasting patches and extended from contour and GANs. The right part of the first column are the extended area.

The source image is firstly extended in contour domain based on image quilting, and a coarse-to-fine Unet architecture is used to reconstruct high-resolution RGB images. Our approach can generate high-resolution image with high texture and semantic rationality. Also, users can generate panaroma images by recursively apply our model on their images.

References

[1] Barnes C, Shechtman E, Finkelstein A, et al. PatchMatch: A randomized correspondence algorithm for structural image editing[C]//ACM Transactions on Graphics (ToG). ACM, 2009, 28(3): 24.

[2] Shan Q, Curless B, Furukawa Y, et al. Photo uncrop[C]//European Conference on Computer Vision. Springer, Cham, 2014: 16-31.

[3] Efros A A, Freeman W T. Image quilting for texture synthesis and transfer[C]//Proceedings of the 28th annual conference on Computer graphics and interactive techniques. 2001: 341-346.

[4] Pathak D, Krahenbuhl P, Donahue J, et al. Context encoders: Feature learning by inpainting[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2536-2544.

[5] Iizuka S, Simo-Serra E, Ishikawa H. Globally and locally consistent image completion[J]. ACM Transactions on Graphics (ToG), 2017, 36(4): 1-14.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, DavidWarde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc., 2014.

[7] Krishnan D, Teterwak P, Sarna A, et al. Boundless: Generative Adversarial Networks for Image Extension[C]//2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 10520-10529.

[8] Miyato T, Koyama M. cGANs with projection discriminator[J]. arXiv preprint arXiv:1802.05637, 2018.

[9] Dekel T, Gan C, Krishnan D, et al. Sparse, smart contours to represent and edit images[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 3511-3520.

[10] Green B. Canny edge detection tutorial[J]. Retrieved: March, 2002, 6: 2005.

[11] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[C]//International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015: 234-241.

[12] Karras T, Aila T, Laine S, et al. Progressive growing of gans for improved quality, stability, and variation[J]. arXiv preprint arXiv:1710.10196, 2017.