Image Inpainting

Jin Woo, Baik, Stanford University, CA

Abstract—Recently, there have been lots of progress in image inpainting field and state-of-the-art techniques with learning-based methods. However, inpainting results from these techniques are still blurry or awkward. From this project, I will generate clear and plausible outputs to improve image quality. To achieve this, I proposed Recurrent Convolution and adopted Coarse-to-refine structure [2]. In addition, I proposed Sobel Edge Discriminator, which is a discriminator using Sobel Edge detection algorithm [1]. Moreover, I added frequency separation loss. Since the high frequency components are less important in the coarse network, I applied a low pass filter to get low frequency components. In contrast, I used a high pass filter in refine stage so that network could more focus on details. From these proposed methods, my model achieved better performance than other methods.

Index Terms—Recurrent Convolution, Sobel Edge Discriminator, Frequency Separation Loss, Coarse to Refine.

1 Introduction

Image inpainting is a task which fills missing pixels with semantically and perceptually plausible contents. We can remove unwanted objects through inpainting by using a mask which covers the unwanted region. To produce appropriate reconstruction, it should be semantically plausible and match with the existing backgrounds. These processes could be done by using deep learning techniques.

There are a lot of image inpainting techniques. I adopted coarse-to-refine structure [2]. This structure fills the masked region through coarse network and refine network. The coarse network takes an original image and the mask as an input to generate coarse output. Then, the refine network takes coarse output as an input to generate refined output. By using this structure, we could expand the receptive field and stabilize the training stage.

Based on this coarse-to-refine structure, I have proposed 3 ideas; Recurrent Convolution, Sobel Edge Discriminator, and frequency separation loss. Recurrent Convolution (or RNN) is widely used when the network deals with sequential data such as music, movie, and natural languages. Since my model uses coarse-to-refine structure, which the output from coarse network becomes an input of refine network, our model can be treated as a sequential model. This is the reason why I adopted RNN.

Sobel Edge Discriminator originates from Sobel Edge Detection [1]. I tried to use the edge information suggested in other image inpainting paper [3] to increase the performance. The overall quality has been greatly improved by using the edge information. Nevertheless, there are some limitations. One of the limitation is that if the edge map is not accurate, the subsequent network cannot produce an appropriate output. In my model, Sobel edge discriminator takes the edge map as input and determines whether the edge map is real or fake.

Frequency separation loss is designed to catch the structure of an input more precisely. Since the basic information of an image is concentrated in low frequency regions, coarse network focuses on low frequency components by eliminating the high frequency details. This method makes the coarse network simple, while maintaining core information of an image. In contrast, refine network focuses on high frequency information(details).

By implementing these techniques, performance was greatly improved comparing with other inpainting models [3], [4], [5]. Especially, our model shows the best performance at landscape images such as Places365.

2 Related Work

There are some popular methods for inpainting task. **Globally and Locally Consistent Image completion** [7] has two discriminators: global discriminator and local discriminator. The global discriminator makes an output consistent with the global context and the local discriminator makes an output consistent with local regions which corresponds to the masked region. This model is composed of three networks: completion networks, local discriminator, and global discriminator. The completion network fills the masked region using CNN structure. Then, the global and local discriminator evaluate whether the output is consistent with the original image globally and locally.

Generative Image Inpainting with Contextual Attention [2] introduces the concept of 'contextual attention layer'. First, the input feature is divided into two regions. One is called 'Foreground' which corresponds to the masked region and the other is called 'Background'. The objective of this layer is to find a patch in background which matches with foreground. To achieve this, background should be transformed into convolutional filters with cosine similarity. Then, the convolution operation is applied to the foreground and convolutional filters. By applying the softmax operation to the output, attention scores for each pixel can be achieved. Finally, the foreground region can be reconstructed by applying deconvolution to the attention score.

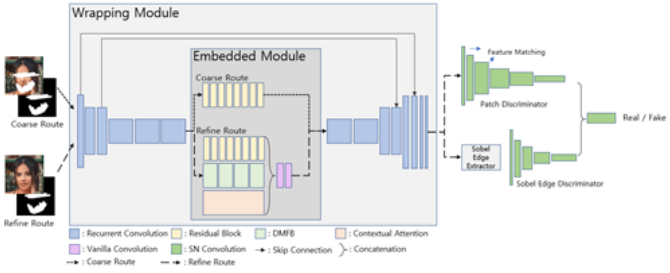
Free-Form Image Inpainting with Gated Convolution [4] solves the problem of vanilla convolution which treats all pixels as valid ones. While most of image inpainting networks used hard-gating mask, this model used soft mask. Gating corresponds to a soft mask which is updated automatically with convolution. By using this mechanism, the network can selectively learn masked region.

Image Fine-Grained Inpainting [6] introduced 'dense multi-scale fusion network (DMFN)'. Unlike common dilated convolutions, DMFB uses hierarchical features which come from various kinds of convolution having different dilation rates. In addition, the author proposes a new concept called 'self-guided regression loss' which corrects low-level features by using normalized discrepancy map.

Recurrent Neural Network(RNN) is designed to deal with sequential data such as music, movies, and natural language. Since sequential data has different length, traditional neural network is not adequate. The main difference between RNN and traditional neural network is that traditional neural network only uses current input, while RNN uses past information by using hidden states.

Edgeconnect: Generative Image Inpainting with Adversarial Edge Learning [3] uses edge information when generating an image. It first draws an edge map of masked region. Then the network uses edge map to fill masked region. The network is composed of two stages. First stage is called edge generation stage. In this stage, the network generates edge map of the masked region. The next stage is called image completion stage. It fills the masked region by using the edge map generated at the first stage.

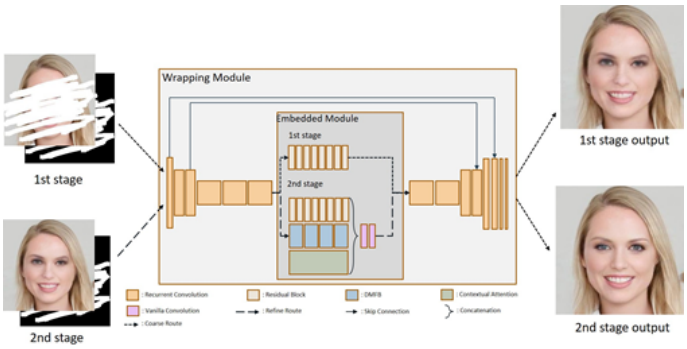
3 Methods



This is an overall structure of my model. There are two main modules: wrapping module and embedded module. After wrapping module, the result goes through patch discriminator and Sobel edge discriminator which determines whether the generated image is real or fake. The results from these discriminators are concatenated and make the final decision whether the generated image is real or fake.

Wrapping module is composed of encoder-decoder network. Inputs(original image and mask) are encoded using recurrent convolution and passes through embedded module. Then, the output of embedded module is decoded using recurrent convolution.

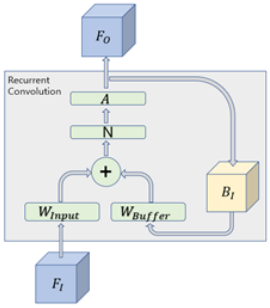
Embedded module is composed of coarse route and refine route. Coarse route is composed of residual block and refine route is composed of residual block, DMFB [6] and contextual attention [2] which the concatenation of these blocks goes through vanilla convolution. Further details are elaborated in **Appendix B**.



The picture above shows the overall flow of my model. First, feed masked image and the mask into the network to get coarse output. Then, feed coarse output and the mask into the network to get final output.

3.1 Recurrent Convolution

The concept of 'recurrent convolution' comes from Recurrent Neural Network(RNN). Coarse-refine structure is adopted in my model. One of the problem with this structure is that network becomes too long since coarse stage and refine stage are connected in series. From this reason, generator loss might not reach to the end of a stage. To handle this problem, I adopted RNN structure. Features that come from the coarse stage are reused at refine stage.



F_I: Input Features
F_O: Output Features
B_I: Input Buffer
W_{Input}: Conv for Input Features
W_{Buffer}: Conv for Buffer
N: Instance Normalization
A: *ELU activation function*

$$F_O = B_I$$
$$F_O = A[(F_I * W_{Input}) + (B_I * W_{Buffer})]$$

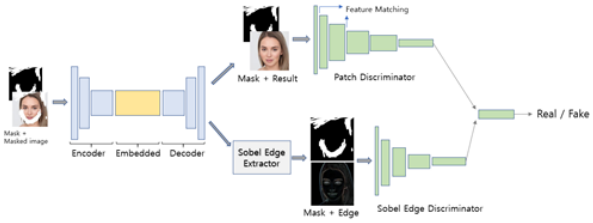
The picture above shows the design of a recurrent convolution. Initially, buffer is set to zero. Convolution operation is applied to the inputs (input features and buffer) and their weights. Then, sum them up and apply instance normalization and activation function (ELU function) as followed. Finally, the output is achieved, and outcome is stored in buffer. By using this method, coarse and refine networks could be used simultaneously, while solving the gradient vanishing problem. This is the reason why recurrent convolution is proposed.

3.2 Sobel Edge Discriminator

Sobel Edge Discriminator is a discriminator which uses Sobel edge detection [1]. It uses Sobel edge extractor which role is to extract edge information of an image. Horizontal and vertical edge information could be achieved by using horizontal line detector and vertical line detector. From this information, the whole edge of an image could be achieved.

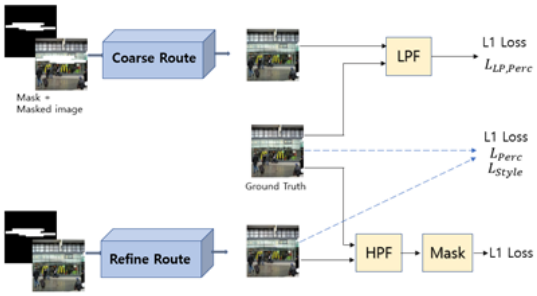
It looks similar to Edgeconnect [3] since both models use edge information. However, there is a difference between my model and Edgeconnect. Edgeconnect uses edge

information in the second stage. It draws the edge first and puts it into subsequent network to get a complete image. So, if the edge map is awkward, the completed image will be awkward as well. However, in my model, the network learns edge information and output simultaneously.



The picture above shows how Sobel edge discriminator is applied. After the network produces an image, the result is fed into the patch discriminator and Sobel edge extractor. Then, the outputs of two discriminators are concatenated and determine whether the generated image is real or fake.

3.3 Frequency Separation Loss



I have also added frequency separation loss. High frequency details are less important in the coarse stage, so I used low pass filter to focus on low frequency information to catch basic information. In contrast, I used high pass filter at the refine stage to achieve high frequency details. In addition, the L1 loss between the ground truth and the reconstructed image was applied at the refine stage.

4 Results and Analysis

The network was trained by CelebaHQ dataset for human face inpainting and Places365 dataset was used for place inpainting. For the evaluation, Google search face images were used for face inpainting and Places365 testset were used for place inpainting. I have compared the result with 'Free-form image inpainting with gated convolution' [4], 'Edgeconnect: Generative image inpainting with adversarial edge learning' [3], and 'Pluralistic image completion' [5].

Hinge loss is used for discriminator with learning rate of 0.004, which is commonly used in GAN.

$$L_D = \mathbb{E}_{x\sim P_{data}(x)}[ReLU(1 - D(x)] + \mathbb{E}_{z\sim P_z(z)}[ReLU(1 + D(G(z))]$$

The total generator loss can be represented as below. The learning rate of generator is 0.001.

$$L_{Total} = 0.01L_G + 5L_1 + L_{lpf} + L_{hpf} + L_{LP,Perc} + L_{Perc} + 250L_{Style} + L_{FM}$$

The generator loss is composed of hinge loss, L1 loss, perceptual loss and style loss. These losses make the network robust to variations. The details of these losses are shown as below.

$$L_G = -\mathbb{E}_{z\sim P_z(z)}[D(G(z))]$$

$$L_{FM} = \mathbb{E}\left[\sum_{i=1}^{s}\frac{1}{N}\sum_{}^{N}\left|D_i(I_{gt}) - D_i(I_{pred,refine})\right|\right]$$

$$L_{lpf} = \frac{1}{N}\sum_{}^{N}|F_L(I_{gt}) - F_L(I_{pred,refine})|$$

$$L_{hpf} = \frac{1}{N}\sum_{}^{N} M * |F_H(I_{gt}) - F_H(I_{pred,refine})|$$

$$L_{l1} = \frac{1}{N}\sum_{}^{N}|I_{gt} - I_{pred,refine}|$$

$$L_{perc} = \mathbb{E}\left[\sum_{i}\|\Phi_i(I_{gt}) - \Phi_i(I_{pred,refine})\|_1\right] \quad i = \text{relu1\_1, relu2\_1, relu3\_1, relu4\_1, relu5\_1}$$

$$G_j^{\Phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j}\sum_{h=1}^{}\sum_{w=1}^{}\Phi_j(x)_{h,w,c}\Phi_j(x)_{h,w,c'}$$

$$L_{style} = \sum_{j}\|G_j^{\Phi}(I_{gt}) - G_j^{\Phi}(I_{pred,refine})\|_1 \quad j = \text{relu2\_2, relu3\_4, relu4\_4, relu5\_2}$$

For the optimizer, I used Adam optimizer with a learning rate scheduler of beta1 and beta2 as 0.5 and 0.9. For the learning, I have iterated 200,000 times and set the learning rate decay as 0.75 after 100,000 iterations.

4.1 Quantitative Result

For the evaluation metrics, L1 error, L2 error, perceptual loss, PSNR, and SSIM were used. In addition, the masks were divided into 3 parts by mask rates from 10∼20%, 20∼30%, and 30∼40%.

4.1.1 *CelebaHQ dataset*

CelebaHQ dataset comparison

Mask rate (%)	10~20				20~30				30~40			
Type	Gated	Edge	Plural	Ours	Gated	Edge	Plural	Ours	Gated	Edge	Plural	Ours
L1 error (%) ▼	4.95	5.79	**4.11**	5.76	6.54	6.70	**6.32**	6.54	9.86	9.19	10.12	**8.76**
L2 error (%) ▼	1.47	1.32	1.47	**1.28**	2.03	1.54	2.72	**1.43**	4.38	2.79	4.81	**2.58**
Perceptual Loss ▼	0.431	0.554	**0.358**	0.542	0.592	0.632	**0.512**	0.604	0.813	0.794	**0.739**	0.762
PSNR ▲	29.575	29.803	29.693	**29.920**	27.853	28.748	27.847	**29.152**	24.026	25.888	24.468	**26.012**
SSIM ▲	0.933	0.917	**0.941**	0.920	0.904	0.898	**0.912**	0.905	0.849	0.854	0.854	**0.863**

Mask 10~20 % Best Mask 20~30 % Best Mask 30~40 % Best

▼ : Lower is better
▲ : Higher is better

My model performs best at 30∼40% mask rate. My model shows the best performance for all categories except for the perceptual loss. And for the 10∼20% and 20∼30% rates, my model shows best performance at L2 error and PSNR.

4.1.2 Places 365 dataset

Places365 dataset comparison

Mask rate (%)	10~20				20~30				30~40			
Type	Gated	Edge	Plural	Ours	Gated	Edge	Plural	Ours	Gated	Edge	Plural	Ours
L1 error (%) ▼	14.25	12.70	14.60	**11.82**	16.47	15.52	18.89	**14. 02**	28.24	27.55	32.087	**24.840**
L2 error (%) ▼	12.25	10.03	10.53	**9.99**	12.80	11.17	13.89	**10.53**	25.62	22.06	26.217	**19.999**
Perceptual Loss▼	0.682	0.592	0.718	**0.546**	0.927	0.847	1.001	**0.775**	1.311	1.205	1.413	**1.118**
PSNR ▲	23.311	24.112	22.835	**24.237**	21.617	22.270	21.313	**22.572**	19.114	20.104	18.890	**20.308**
SSIM ▲	0.899	0.898	0.880	**0.907**	0.858	0.856	0.832	**0.871**	0.769	0.768	0.729	**0.789**

Mask 10~20 % Best Mask 20~30 % Best Mask 30~40 % Best

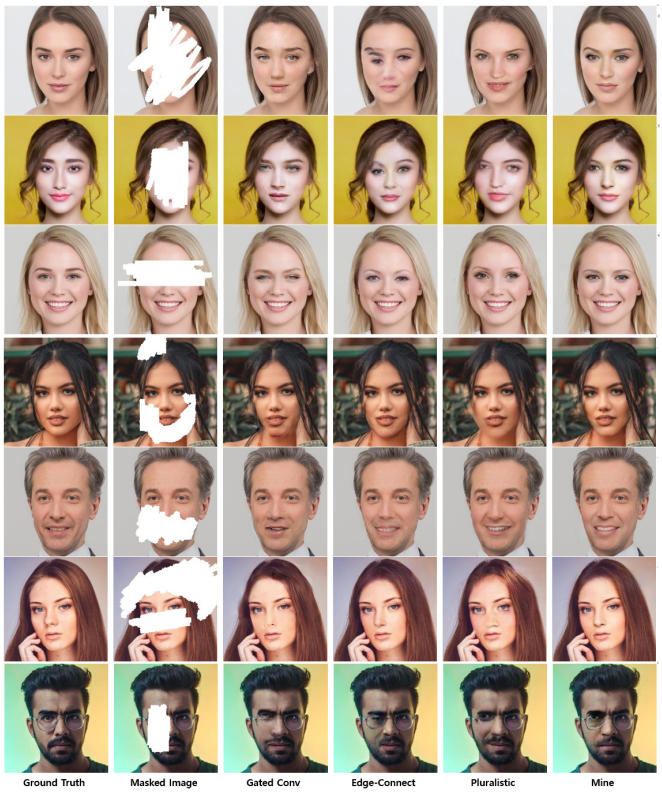
▼ : Lower is better
▲ : Higher is better

For Places365 dataset, my model outperforms other models in all categories.

4.2 Qualitative Result

Each result is aligned with the order of ground truth – masked image - Gatedconv – Edgeconnect – Pluralistic – mine.

4.2.1 CelebaHQ dataset

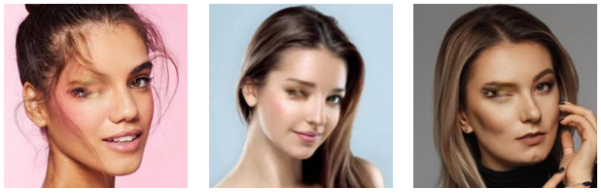


4.2.2 Places 365 dataset

5 Conclusion

As shown in the **4. Result and Analysis**, my model generates better than other methods. These performances can be attributed to my proposed ideas. First, I proposed recurrent convolution. By using recurrent convolution, network size could be reduced, so the network became robust to gradient vanishing problem. Next, I proposed Sobel edge discriminator. By using Sobel edge discriminator, the network could simultaneously learn edge information and image generation. Lastly, I added frequency separation loss. This method made the coarse network to focus on low frequency components and the refine network to focus on high frequency components.

Coarse Output



Refine Output

Even though the model shows good performance, there is still a problem to be solved. For example, as shown in the picture above, there are some cases which coarse outputs were better than refine outputs. From this result, we can conclude that there might be a problem with refine network. For the future study, refine network could be modified to solve this issue.

APPENDIX A
CODE

You can find the code in https://github.com/jwbaik96/Image-inpainting.git

APPENDIX B

B.1 Horizontal and vertical filters

```
filter_h = [[[[1, 2, 1], [0, 0, 0], [-1, -2, -1]]]]
filter_v = [[[[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]]]
```

B.2 Sobel edge extractor

```
R_edge_h = conv2d(R_imgs, filter_h, stride=1, padding=1)
G_edge_h = conv2d(G_imgs, filter_h, stride=1, padding=1)
B_edge_h = conv2d(B_imgs, filter_h, stride=1, padding=1)
R_edge_v = conv2d(R_imgs, filter_v, stride=1, padding=1)
G_edge_v = conv2d(G_imgs, filter_v, stride=1, padding=1)
B_edge_v = conv2d(B_imgs, filter_v, stride=1, padding=1)

R_edge = R_edge_h ** 2 + R_edge_v ** 2
G_edge = G_edge_h ** 2 + G_edge_v ** 2
B_edge = B_edge_h ** 2 + B_edge_v ** 2
```

B.3 Patch discriminator

Type	Kernel	Stride	Outputs	Padding	Dilation rate	Nonlinearity
Disconv	5×5	2	48	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	96	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	192	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	192	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	192	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	384	SAME	1	LeakyReLU(0.2)

B.4 Sobel edge discriminator

Type	Kernel	Stride	Outputs	Padding	Dilation rate	Nonlinearity
Disconv	5×5	4	24	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	48	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	96	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	192	SAME	1	LeakyReLU(0.2)
Disconv	5×5	2	192	SAME	1	LeakyReLU(0.2)

B.5 Low pass filter / High pass filter

```
low_pass_filter = ones(size=(1, 1, 5, 5)) / 25.0
high_pass.filter = [[[[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]]]]


R_low_imgs = conv2d(R_imgs, low_pass_filter, stride=1)
G_low _imgs = conv2d(G_imgs, low_pass_filter, stride=1)
B_low _imgs = conv2d(B_imgs, low_pass_filter, stride=1)


R_high_imgs = conv2d(R_imgs, high_pass_filter, stride=1)
G_high_imgs = conv2d(G_imgs, high_pass_filter, stride=1)
B_high_imgs = conv2d(B_imgs, high_pass_filter, stride=1)
```

B.6 Generator

Layer	Type	Kernel	Stride	Outputs	Padding	Nonlinearity
Encoder_layer1 (skip connection)	RecurrentConv	5	1	16	SYMMETRIC	ELU
Encoder_layer2	RecurrentConv	3	2	32	SYMMETRIC	ELU
Encoder_layer3 (skip connection)	RecurrentConv	3	1	32	SYMMETRIC	ELU
Encoder_layer4	RecurrentConv	3	1	64	SYMMETRIC	ELU
Encoder_layer5	RecurrentConv	3	1	64	SYMMETRIC	ELU
Encoder_layer6	RecurrentConv	3	1	64	SYMMETRIC	ELU
Decoder_layer1	RecurrentConv	3	1	64	SYMMETRIC	ELU
Decoder_layer2	RecurrentConv	3	1	64	SYMMETRIC	ELU
Decoder_layer3	RecurrentDeConv	3	1	32	SYMMETRIC	ELU
Decoder_layer4 (skip connection)	RecurrentConv	3	1	32	SYMMETRIC	ELU
Decoder_layer5	RecurrentDeConv	3	1	16	SYMMETRIC	ELU
Decoder_layer6 (skip connection)	RecurrentConv	3	1	16	SYMMETRIC	ELU
Decoder_layer7	RecurrentConv	3	1	8	SYMMETRIC	ELU
Decoder_layer8	RecurrentConv	3	1	3	SYMMETRIC	NONE

REFERENCES

[1] Sobel, Irwin, "An Isotropic 3x3 Image Gradient Operator" in Presentation at Stanford A.I. Project, 1968

[2] Yu, Jiahui and Lin, Zhe and Yang, Jimei and Shen, Xiahui and Lu, Xin and Huang, Thomas S, "Generative image inpainting with contextual attention," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp5505-5514.

[3] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning. arxiv 2019," arXiv preprint arXiv:1901.00212.

[4] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 4471–4480.

[5] Zheng, Chuanxia and Cham, Tat-Jen and Cai, Jianfei, "Pluralistic image completion," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp1438-1447

[6] Zheng Hui, Jie Li, Xiumei Wang and Xinbo Gao, "Image Fine-Grained Inpainting"

[7] Iizuka, Satoshi and Simo-Serra, Edgar and Ishikawa, Hiroshi, "Globally and Locally Consistent Image Completion," in Association for Computing Machinery, 2017