FINAL PROJECT

B05902021 You-Chien, HSU
B05902120 Yu-Ting, TSENG

Image Inpainting

PREFACE

- <u>Motivation</u>

  Hole Filling — How About Recovering the Corrupted Image?

- <u>Problem Definition</u>

  How to Implement Image Inpainting?

  — *Image Inpainting is the task of filling holes in an image!*

METHODOLOGY

- <u>Traditional Method</u>

  Patch Match

  Globally and Locally Consistent

- <u>New Method</u>

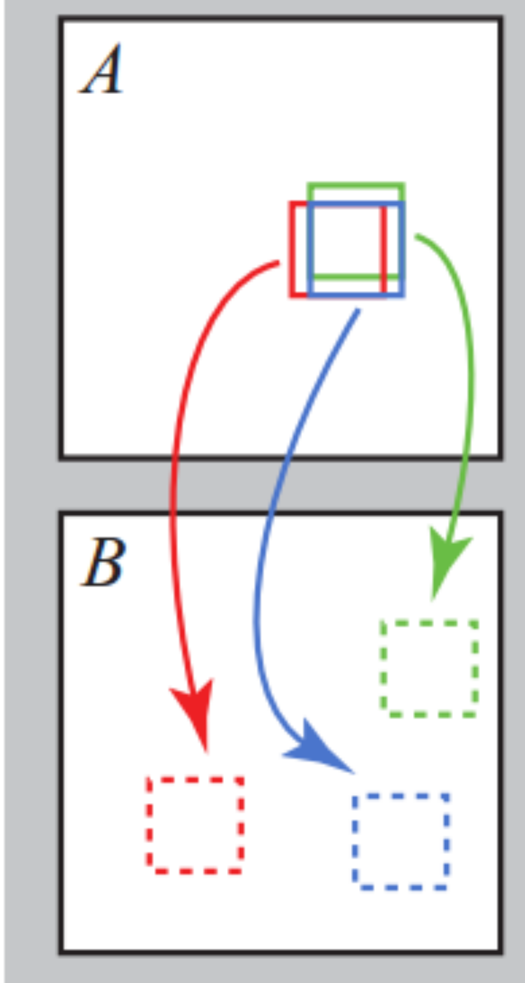
  Partial Convolution

APPROACH I — PATCH MATCH

- Concept Recover by the other area (patch) in the image.

APPROACH I — PATCH MATCH

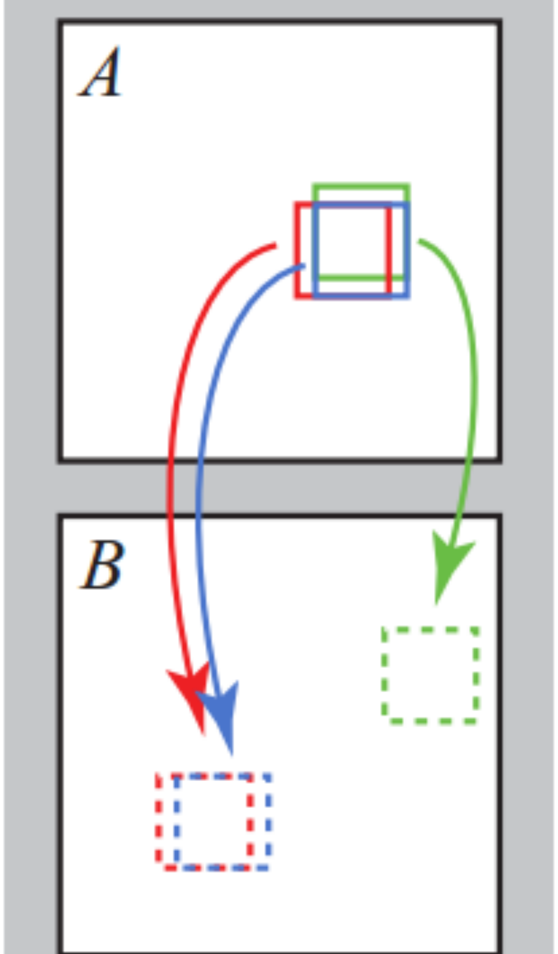
- <u>What is "Patch"?</u> A window that contains several pixels.
- <u>Why "Patch"?</u> More information than one pixel.
- <u>Algorithm</u> Step #1 Initialization
  1. Decide which to be the target pixel.
  2. Let the pixel with offset be patch *p*.
  3. Compare to another patch in the image.
  4. Compute how similar they are.
  - ✳ If two patches are similar enough,
    we can continue on Step #2.



(a) Initialization

APPROACH I — PATCH MATCH

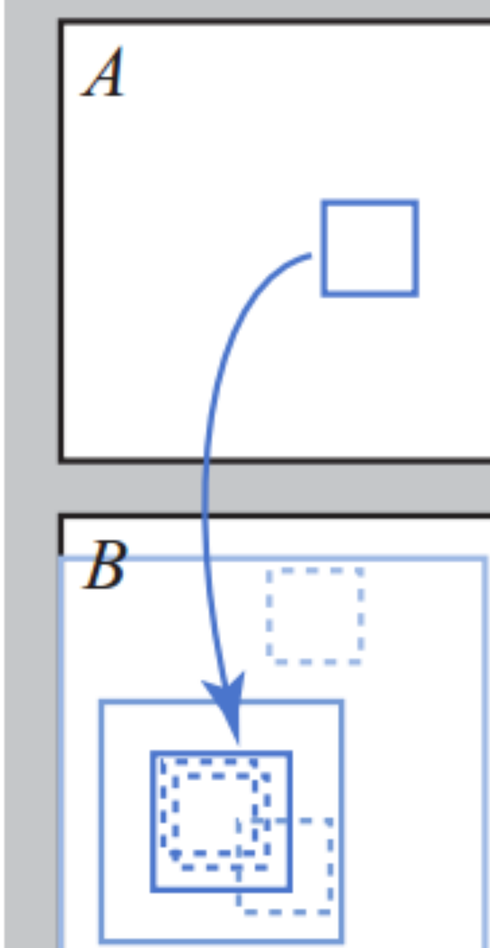
- What is "Patch"? A window that contains several pixels.
- Why "Patch"? More information than one pixel.
- Algorithm Step #2 Propagation
  1. Check the neighboring patches to find whether we can get better similarity.
  - If successfully get better similarity, updates the match of patches.



(b) Propagation

APPROACH I — PATCH MATCH

- What is "Patch"? A window that contains several pixels.
- Why "Patch"? More information than one pixel.
- Algorithm Step #3 Searching
  1. Check the different offsets to find whether we can get better similarity.
  2. Search the radius starts with the size of the image and halved each time until it's 1.
  - ✳ If successfully get better similarity, updates the match of patches.



(c) Search

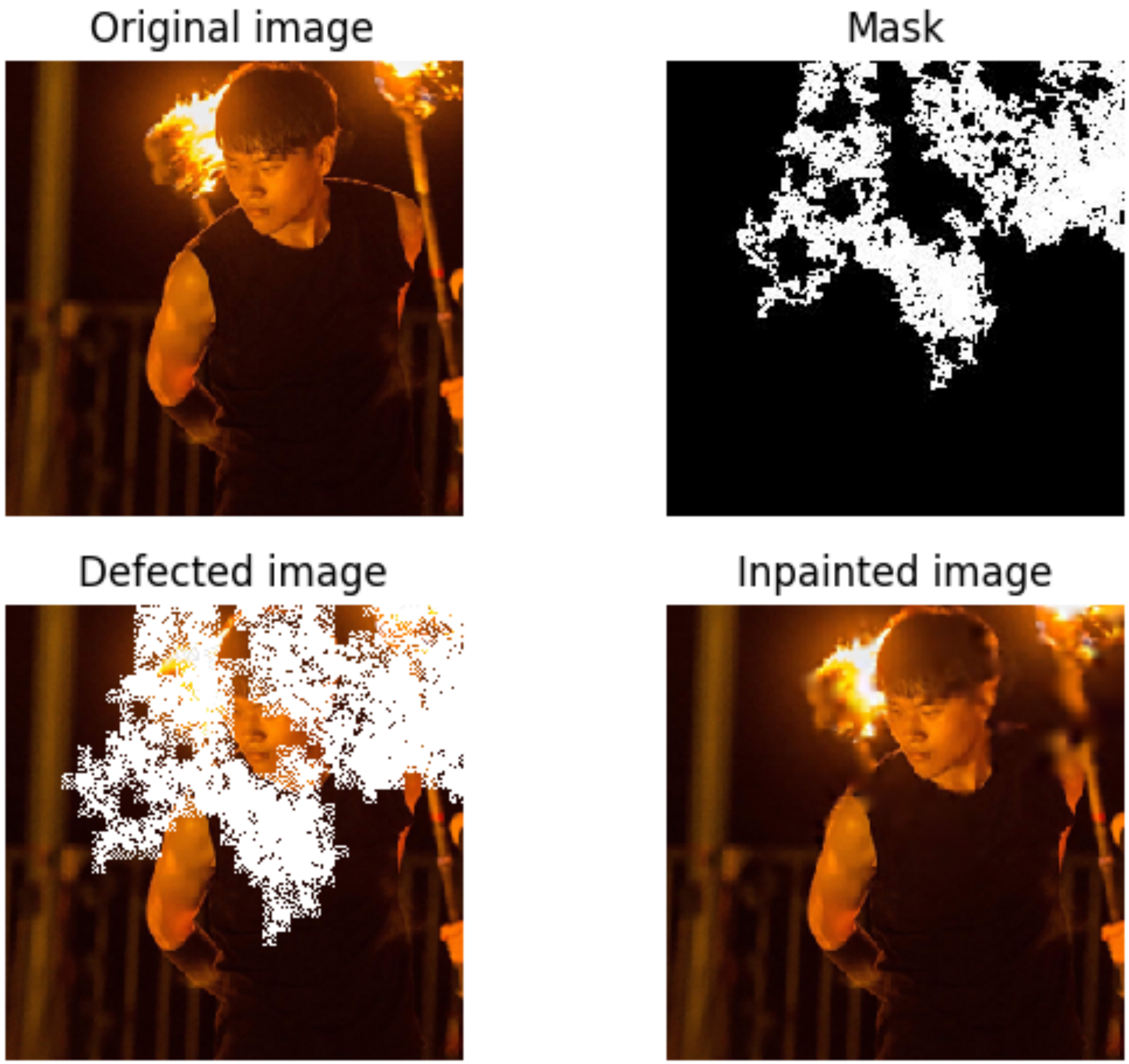
APPROACH I — PATCH MATCH

➤ <u>Implementation</u> “skimage”

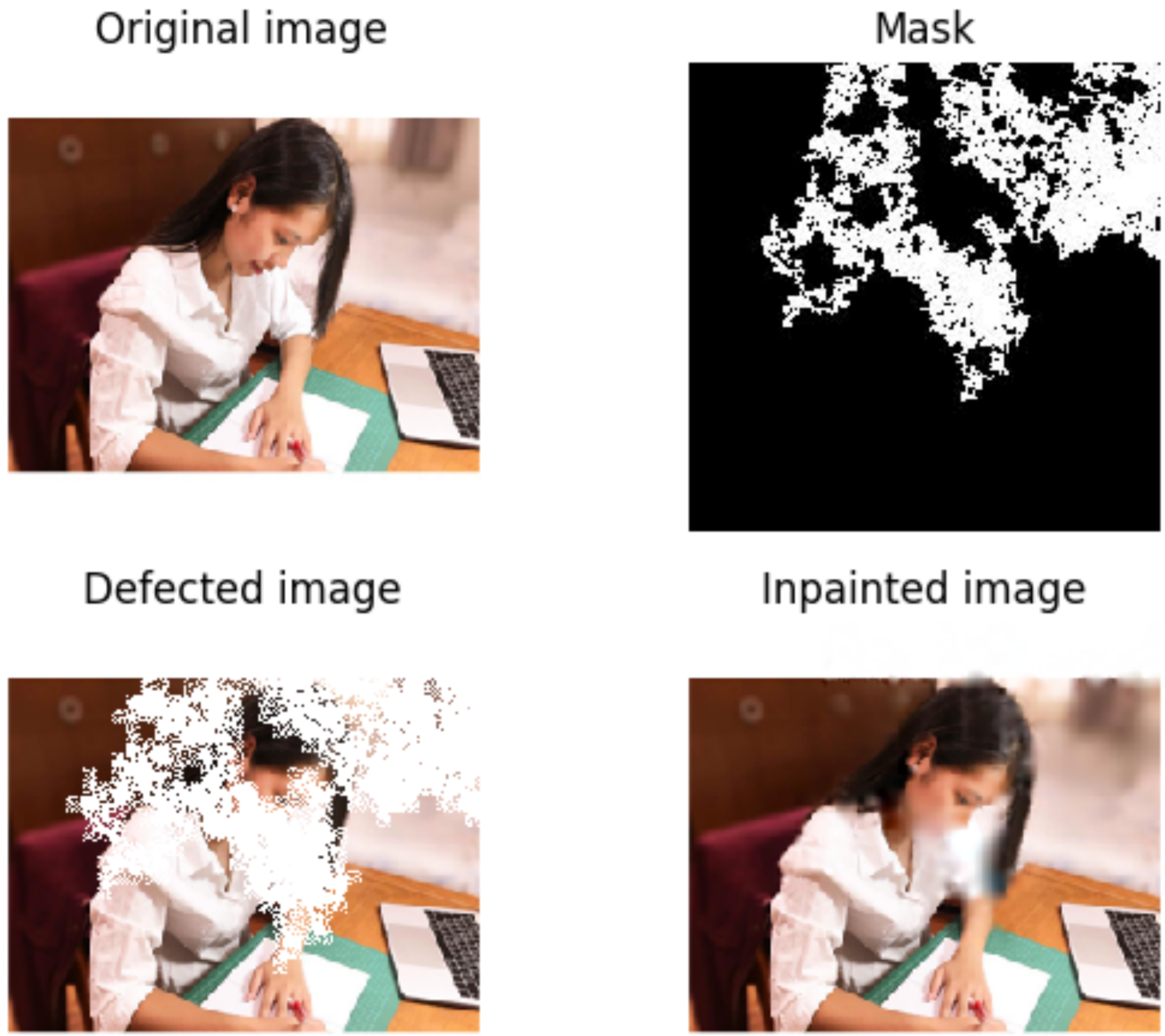
plot_inpaint.py

```python

#filename = input("Please input the filename: ")
filename = 'cute.png'
image_orig = mpimg.imread(filename)
mask = np.invert(mpimg.imread('mask04.jpg'))

print("Filename: ", filename)

image_defect = image_orig.copy()
for layer in range(image_defect.shape[-1]):
    image_defect[np.where(mask)] = 1

print('F')

image_result = inpaint.inpaint_biharmonic(image_defect, mask,
                                          multichannel=True)

print('Finish')

fig, axes = plt.subplots(ncols=2, nrows=2)
ax = axes.ravel()

ax[0].set_title('Original image')
ax[0].imshow(image_orig)
```

APPROACH I — PATCH MATCH



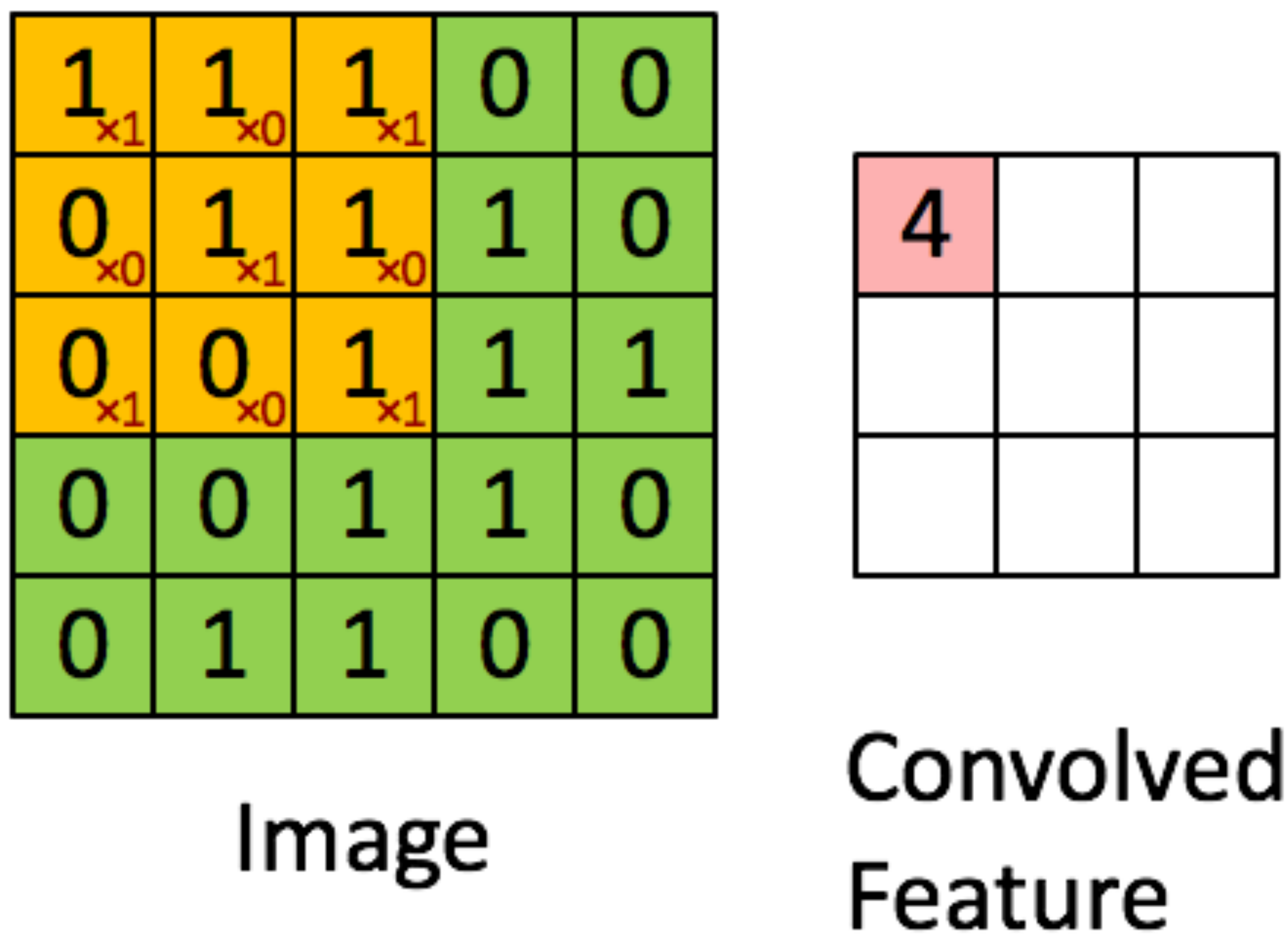
APPROACH I — PATCH MATCH



APPROACH II — PARTIAL CONVOLUTION

- <u>Concept</u>

  Based on Convolution Neural Network.



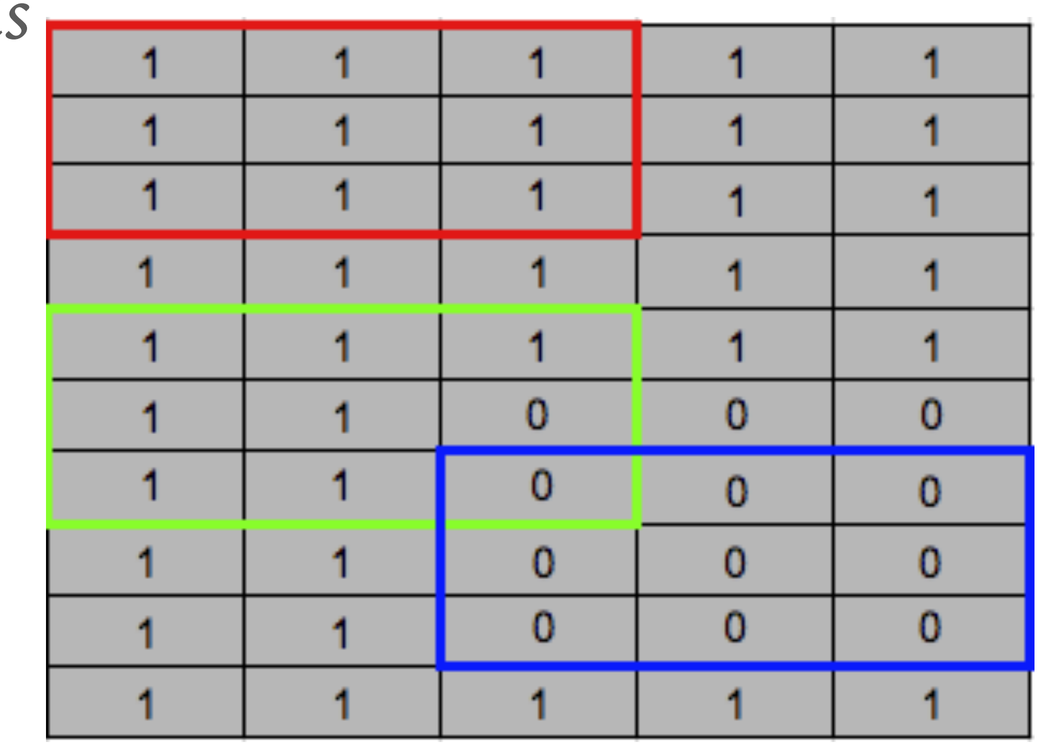
APPROACH II — PARTIAL CONVOLUTION

➤ <u>Algorithm</u>

Mask and image both involve in training.

Only implement in validated area (in mask).

- *Red: All 1, standard convolution*
- *Green: Some 0, learn from neighbor pixels*
- *Blue: Don't do anything at first*
- *Each pixel in mask will becomes 1!*



APPROACH II — PARTIAL CONVOLUTION

- Algorithm

Evaluate the results through loss function.

Pixel Loss — how smoothly the predicted hole transited to surrounding context.

$$\mathcal{L}_{hole} = \|(1-M)\odot(I_{out}-I_{gt})\|_1 \text{ and } \mathcal{L}_{valid} = \|M\odot(I_{out}-I_{gt})\|_1$$

Perceptual Loss — the difference between original image and resultant image.

$$\mathcal{L}_{perceptual} = \sum_{n=0}^{N-1}\|\Psi_n(\mathbf{I}_{out})-\Psi_n(\mathbf{I}_{gt})\|_1 + \sum_{n=0}^{N-1}\|\Psi_n(\mathbf{I}_{comp})-\Psi_n(\mathbf{I}_{gt})\|_1$$

Style Reconstruction Loss —

$$\mathcal{L}_{style_{out}} = \sum_{n=0}^{N-1}\left\|K_n\left(\left(\Psi_n(\mathbf{I}_{out})\right)^{\intercal}\left(\Psi_n(\mathbf{I}_{out})\right) - \left(\Psi_n(\mathbf{I}_{gt})\right)^{\intercal}\left(\Psi_n(\mathbf{I}_{gt})\right)\right)\right\|_1$$

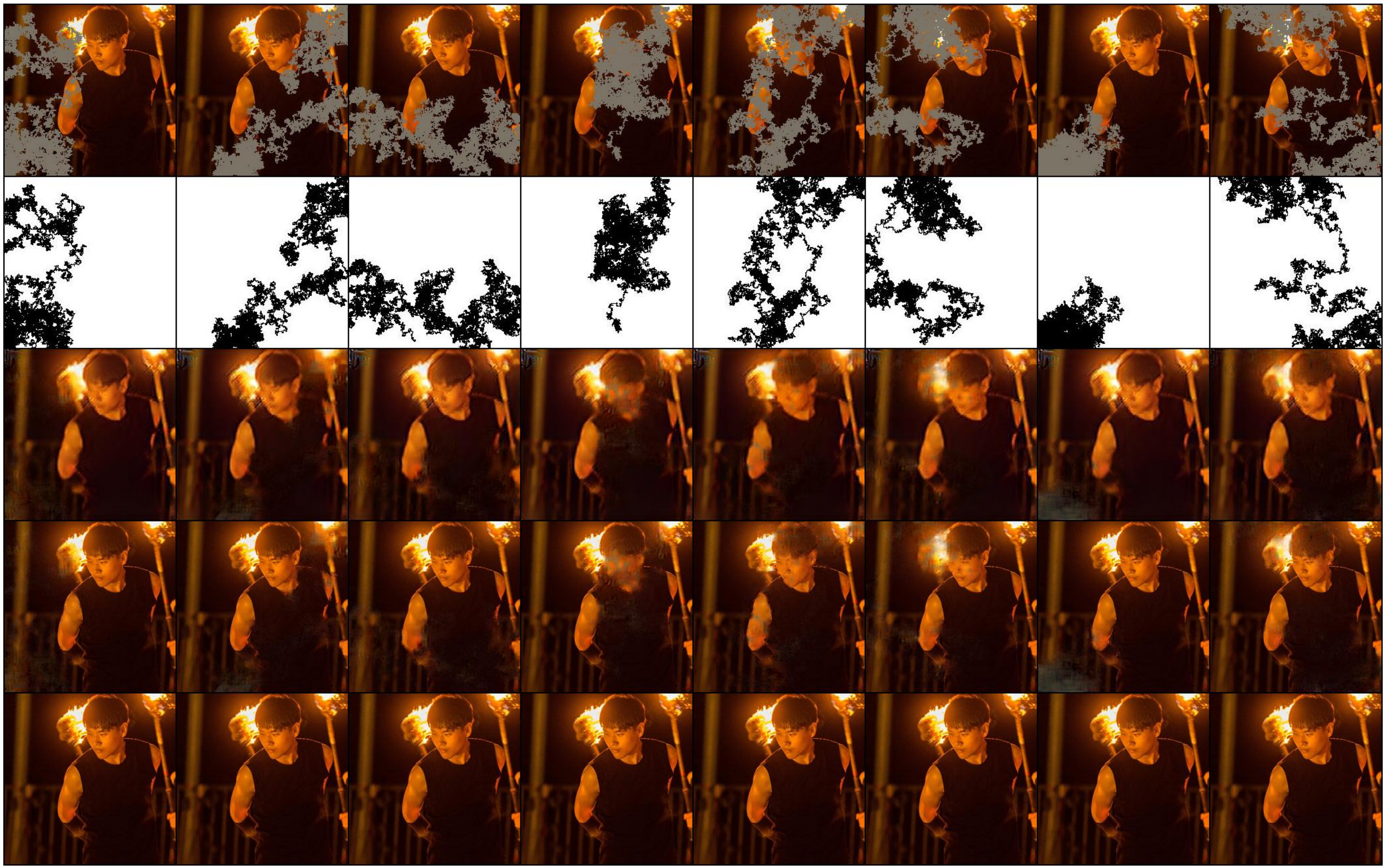
$$\mathcal{L}_{style_{comp}} = \sum_{n=0}^{N-1}\left\|K_n\left(\left(\Psi_n(\mathbf{I}_{comp})\right)^{\intercal}\left(\Psi_n(\mathbf{I}_{comp})\right) - \left(\Psi_n(\mathbf{I}_{gt})\right)^{\intercal}\left(\Psi_n(\mathbf{I}_{gt})\right)\right)\right\|_1$$

APPROACH I — PATCH MATCH

- Implementation

```python
from places2 import Places2
from evaluation import evaluate
from net import PConvUNet
from util.io import load_ckpt

parser = argparse.ArgumentParser()
# training options
parser.add_argument('--root', type=str, default='./data')
parser.add_argument('--snapshot', type=str, default='')
parser.add_argument('--image_size', type=int, default=256)
args = parser.parse_args()

device = torch.device('cuda')

size = (args.image_size, args.image_size)
img_transform = transforms.Compose(
    [transforms.Resize(size=size), transforms.ToTensor(),
     transforms.Normalize(mean=opt.MEAN, std=opt.STD)])
mask_transform = transforms.Compose(
    [transforms.Resize(size=size), transforms.ToTensor()])

dataset_val = Places2(args.root, img_transform, mask_transform, 'val')

model = PConvUNet().to(device)
load_ckpt(args.snapshot, [('model', model)])
```

APPROACH II — PARTIAL CONVOLUTION



APPROACH II — PARTIAL CONVOLUTION

REFERENCE

Image Inprinting for Irregular Holes Using Partial Convolutions

— NVIDIA Corporation, Decenber 2018

Globally and Locally Consistent Image Completion

— Waseda University, July 2017

Context-Encoders: Feature Learning by Inpainting

— Berkeley CVDR, April 2016