Final Project

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

Jun 14, 2019

Preface

We have learnt the concept of hole filling during first half of the semester. However, within this topic, we only consider the image consisting of binary color, black and white. We wonder whether it is possible to fill the whole of a gray image and even the colorful image. After googling for some related information, we choose the topic, "Image Inpainting". It is the task the of filling holes on the images.

Basic Execution

- Platform: Linux (Workstation with GPU)
- Language: Python

Original Image

We consider both gray scale images and color scale images, and the images come from the profile photos of our members, You-Chien, HSU and Yu-Ying, TSENG. The following images is the mentioned photos.

Figure 1: You-Chien, HSU (Gray)

Figure 2: You-Chien, HSU (Color)

Figure 3: Yu-Ting, TSENG (Gray)

Figure 4: Yu-Ting, TSENG (Color)

Patch Match

This is a traditional method of image inpainting. A "patch" is a part of image with size determined by user. It is a window consists of several pixels; in other words, it contains more information than just a pixel. There are three steps in the algorithm.

1. **Initialization**
   (a) Decide which to be the target pixel.
   (b) Let the pixel with offset to be patch p.
   (c) Compare to another patch in the image.
   (d) Compute how similar they are.
   (e) If two patches are similar enough, we can continue on Step #2.
2. **Propagation**
   (a) Check the neighboring patches to find whether we can get better similarity.
   (b) If successfully get better similarity, updates the match of patches.
   (c) Continue Step #3.
3. **Searching**
   (a) Check the different offsets to find whether we can get better similarity.
   (b) Search the patch with radius starts with the size of the image.
   (c) Half the radius each time until it's 1.
   (d) Repeatedly execute Step #2 and Step #3 until all defected pixels are filled.
4. **Note** We calculate the similarity by MSE (minimum square error).

The images below are the "Original Image", "Mask", "Defected Image" and "Resultant Image" in order. The first 4 images display the results of gray images, and the 4 images following are the resultant images of color.


Figure 5: Original

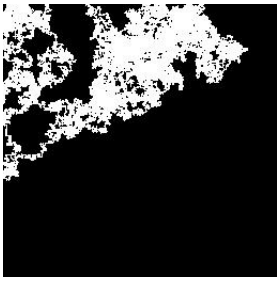
Figure 6: Mask

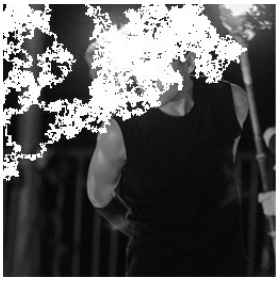
Figure 7: Defected


Figure 8: Resultant

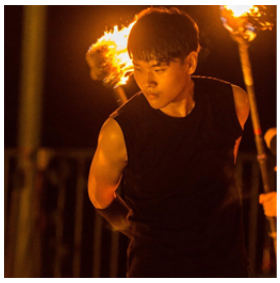
Figure 9: Original

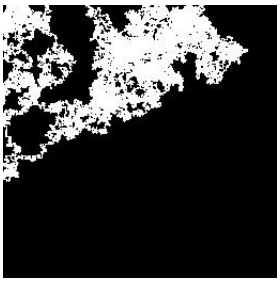
Figure 10: Mask


Figure 11: Defected


Figure 12: Resultant

Furthermore, we adopt 2 different algorithms of Patch Match, NS algorithm and TELEA algorithm, which originate from "Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting" and "An Image Inpainting Technique Based on the Fast Marching Method" respectively. We can find that TELEA algorithm has the texture windows with smaller size; on the other hands, it makes the images more discontinuous. As a whole, NS seems to have better performance than TELEA does.


Figure 13: Original

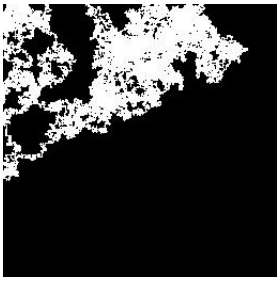
Figure 14: Mask


Figure 15: Defected

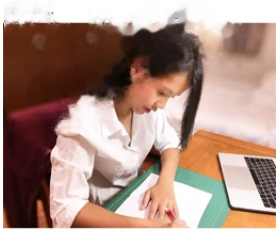
Figure 16: NS


Figure 17: Original

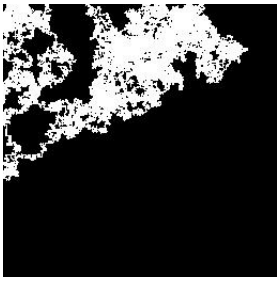
Figure 18: Mask


Figure 19: Defected

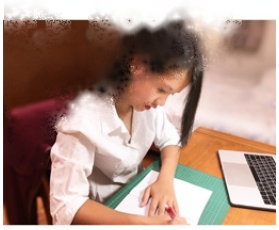
Figure 20: TELEA

It seems that the algorithm perform well, however, the fact is that, if the defected part

contains face, it works badly. We take a try via other ways – a novel idea, "Partial Convolution".

Partial Convolution

It is a algorithm based on convolution neural network (CNN). It also extracts convolved features, but the main difference is that it implement distinct degree of CNN under distinct condition.

- If the window is consists of the pixels all 1 (complete without defect), do the standard convolution.
- If the window is composed of some pixel(s) 0 (some pixels are defected), learn from neighbor pixel.
- If the window contains all 0 (all of the pixels are covered by mask), do not do anything during this iteration.
- Repeatedly do the iteration until all pixels are filled.
- Evaluate the loss function and keep updating the results.

We have try to iterate for 10, 100, 1000, 10000 and 50000 times. It appears that we successfully get the resultant images much more clearer gradually. In addition, the results are better than the traditional methods, Patch Match as we mentioned above. The images are displayed below:

Figure 21: Itr 100

Figure 22: Itr 1000

Figure 23: Itr 10000

Figure 24: Itr 50000

Figure 25: Itr 100

Figure 26: Itr 1000

Figure 27: Itr 10000

Figure 28: Itr 50000

Results

The rows of the first two images represent defected image, TELEA algorithm resultant image and NS resultant image of gray scale and color scale in order.

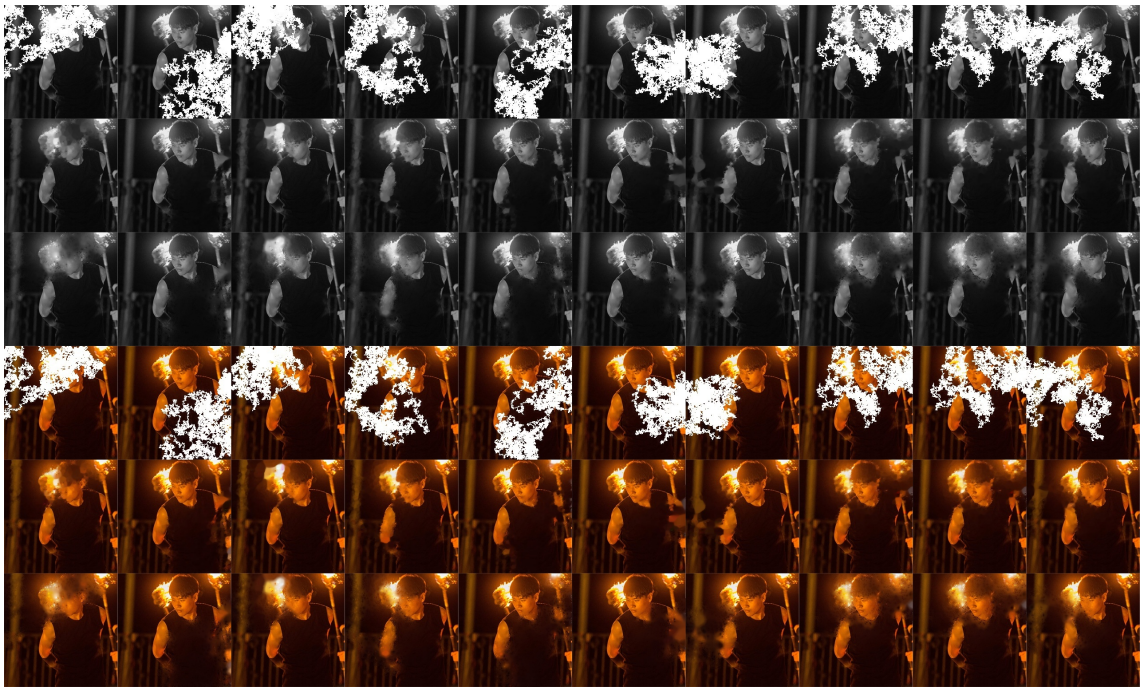

Figure 29: Chien's Results of PatchMatch

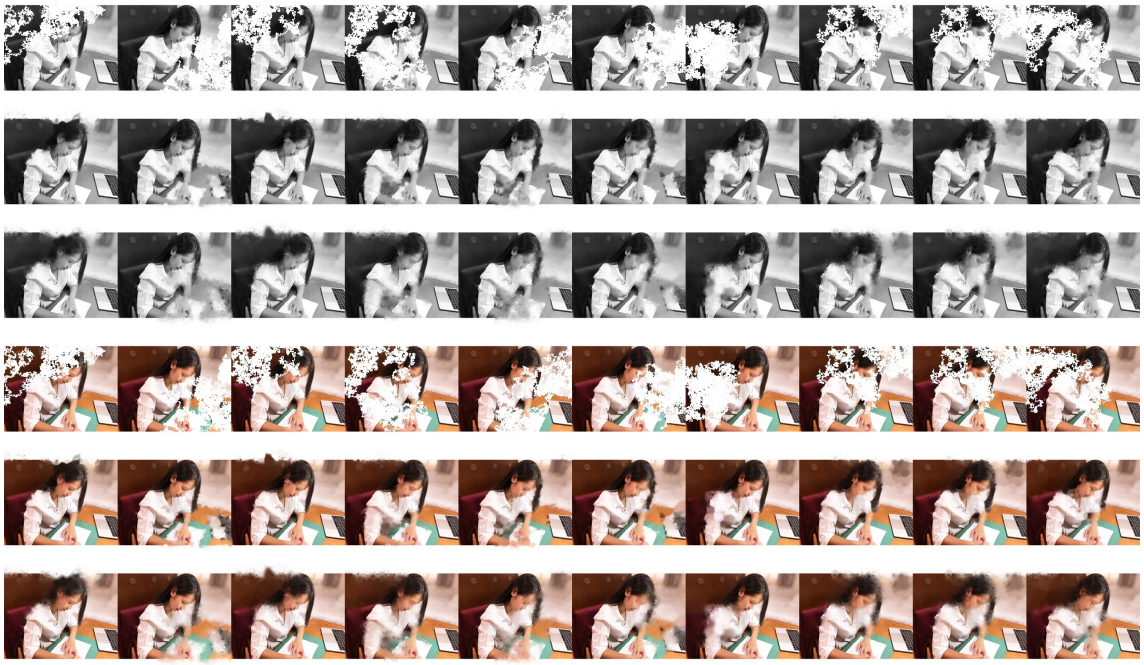

Figure 30: Ting's Results of PatchMatch

The rest of the image is the results of Partial Convolution with iterating 10, 100, 1000, 10000 and 50000 times.

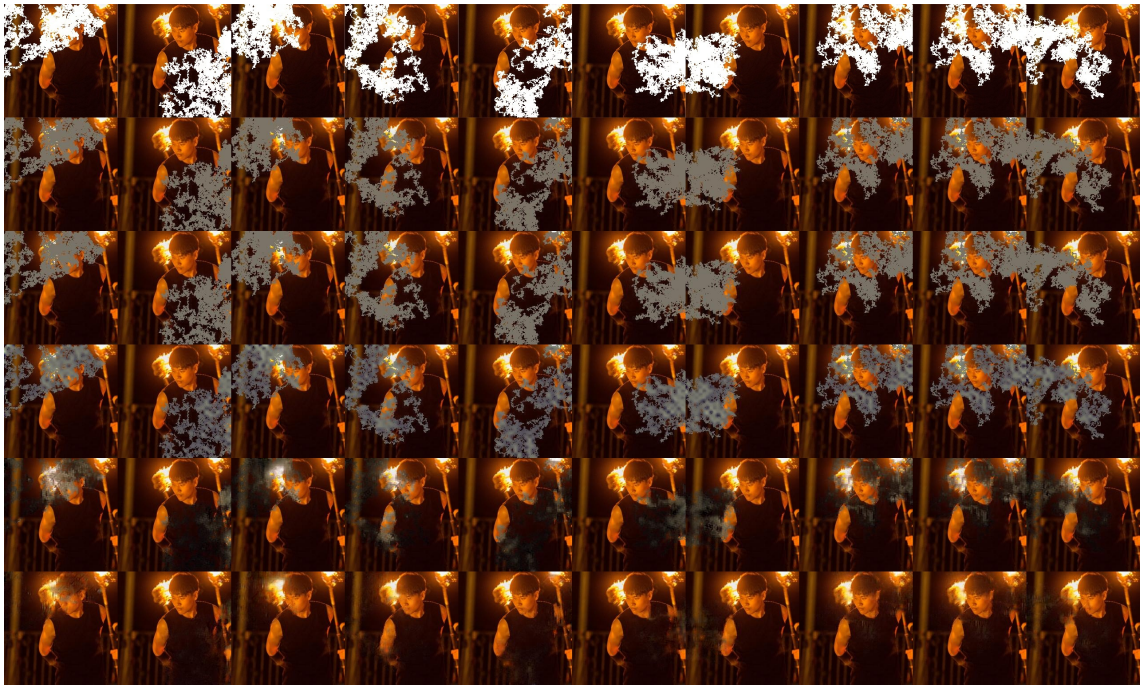

Figure 31: Chien's Results of PartialConvolution

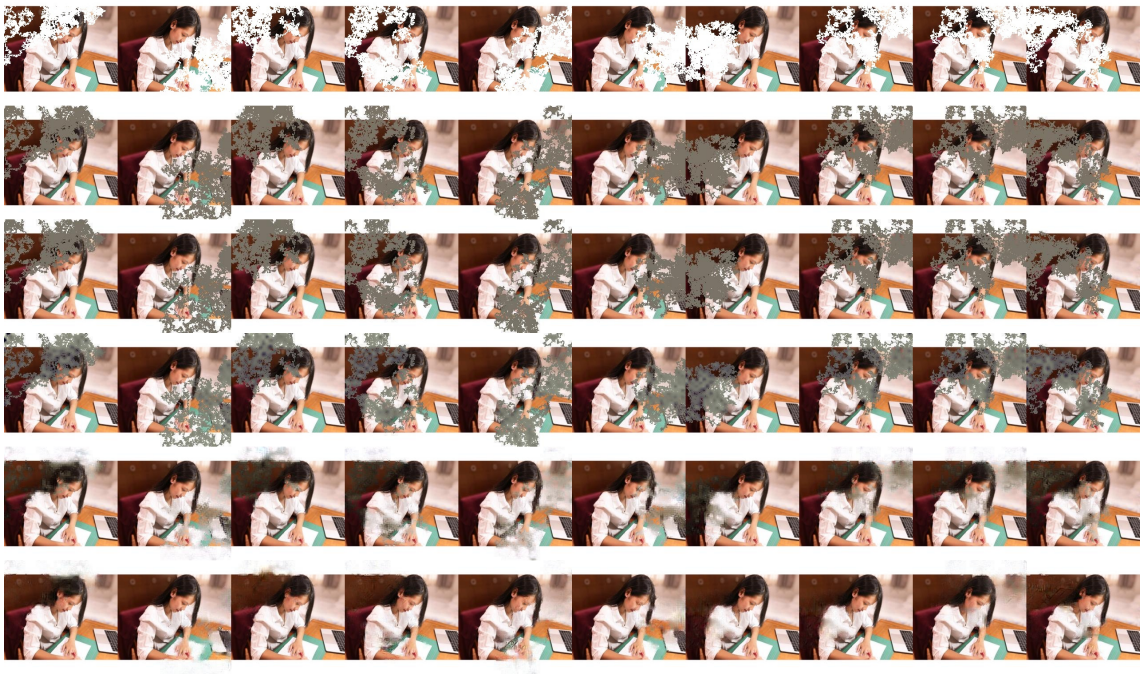

Figure 32: Ting's Results of PartialConvolution

Execution

- Package used: Numpy, OpenCV, Skimage, Pillow, Protobuf, TensorBoard, and Pytorch.
- `PatchMatch_OpenCV.py` to execute Patch Match Method (both NS and TELEA algorithm).

Execute by `python3 PatchMatch_OpenCV.py ${filename}` and get the output images `PatchMatch_${algorithm}_${filename}${k}.jpg`.

- `PatchMatch_Self.py` to execute Patch Match Method (self constructed).
  Execute by `python3 PatchMatch_Self.py ${filename}` and get the output images `PatchMatch_${filename}_${k}.jpg`.

- `Train.py` and `Test.py` to execute Partial Convolution Method.
  Get models by `CUDA_VISIBLE_DEVICES=${GPUid} python3 Train.py`;
  Execute by `python3 Testing.py --snapshot ${model_path}` and get the output images `PartialConvolution_${filename}_${k}.jpg`.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein and Dan B Goldman. *PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing.*

[2] M. Bertalmio, A. L. Bertozzi and G. Sapiro. *Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting.*

[3] Alexandru Telea. *An Image Inpainting Technique Based on the Fast Marching Method.*

[4] Guilin Liu, Fitsum A. Reda, Kevin J., Shih Ting-Chun Wang, Andrew Tao and Bryan Catanzaro. *Image Inpainting for Irregular Holes Using Partial Convolutions.*