

# Image Filtering

## Salt and Pepper noise on Python

ผลการทดลองการ Denoise รูปภาพที่มี Salt & Pepper noise

โดย นายชนกันต์ ชุมทัพ 6410301022

## ผลการทดลอง

1. ภาพต้นฉบับในรูปแบบของ Gray scale



2. ภาพหลังจากการเพิ่ม Noise ด้วยวิธี Salt & Pepper noise และใช้ density = 0.1



### 3. ภาพหลังจากกำจัด Noise ออกไป



#### สรุปผลการทดลอง

จะเห็นได้ว่าในภาพหลังจากการ Denoise ไปแล้ว ยังมี Noise ให้เห็นอยู่เป็นจุด ๆ อยู่ แต่ มีจำนวนน้อยลง และความคมชัดยังเท่าเดิม ซึ่งเกิดจากการทำกระบวนการ Median Blur และใช้ Filter ขนาด  $3 \times 3$  ไป 1 รอบ แต่ไม่ได้เพิ่มขนาด Filter ใน การ Denoise เพิ่ม เพราะ การทดลองการใช้ Filter ขนาดตั้งแต่  $5 \times 5$  ขึ้นไป พบว่า ภาพที่ได้จะมีความเบลอขึ้น จนความคมชัด ของภาพเมื่อเทียบกับภาพต้นฉบับมีความแตกต่างกัน จะเห็นได้ตามตัวอย่างด้านล่างนี้



Filter ขนาด  $5 \times 5$



Filter ขนาด  $7 \times 7$

จากการนี้ข้างต้นทำให้เราต้องสร้างเงื่อนไขในการเช็คความเหมือนกันระหว่างภาพต้นฉบับ และภาพที่กำลังถูก Denoise เป็น Step by Step เพื่อเช็คว่าเราต้องทำการ Denoise ภาพนั้น เพิ่มให้มีความใกล้เคียงกับภาพต้นฉบับที่สุดก็รอด

ซึ่งทำให้เราต้องสร้างเงื่อนไขในการ Denoise ดังนี้

```
def find_max_ssim(noise_free_image, noisy_image):
    quality_range = []
    ssim_value_og = compare_ssim(noise_free_image, noisy_image)

    for i in range(3, 10, 2):
        denoise = cv2.medianBlur(noisy_image, i)
        ssim_value = compare_ssim(noise_free_image, denoise)
        quality_range.append(ssim_value)

    return max(quality_range), ssim_value_og

def denoiser(noise_free_image, noisy_image):

    filter_size = 1
    max_quality, ssim_value = find_max_ssim(noise_free_image, noisy_image)

    print(f"\nMax quality : {max_quality}")
    print(f"SSIM : {ssim_value}")

    filter_size = 1

    while ssim_value != max_quality:

        denoise = cv2.medianBlur(noisy_image, filter_size)
        ssim_value = compare_ssim(noise_free_image, denoise)

        print(f"SSIM after denoise by {filter_size} : {ssim_value}")

        compare_pics = np.concatenate((noise_free_image, denoise), axis=1)
        cv2.imshow(f'Original and Denoise {filter_size}', compare_pics)
        cv2.waitKey(1250)
        cv2.destroyAllWindows()

        filter_size += 2

    cv2.imwrite('pics/denoise.jpg', denoise)

    return 0
```

พังก์ชัน find\_max\_ssim() และ denoiser() ใช้สำหรับการทำ Denoise ของรูปภาพ

จาก Source code ข้างต้น ผู้ได้เลือกใช้ค่าของ SSIM ในการเปรียบเทียบความเหมือนและความต่างของรูปภาพ 2 รูป (วัดค่า Noise) โดยค่า SSIM สามารถระบุ range ในความเหมือนและความต่างของรูปภาพได้ ซึ่งจะมีค่าอยู่ที่ -1 ถึง 1 ซึ่งยิ่งถ้าหากค่าอยู่น้อย จะแสดงถึงรูปภาพทั้ง 2 รูปนั้นมีความแตกต่างกันมาก แต่ถ้าหากค่าอยู่มาก จะแสดงถึงรูปภาพทั้ง 2 รูปนั้นมีความเหมือนหรือใกล้เคียงกันมาก ตามลำดับ

ผู้เลือกใช้ฟังก์ชัน compare\_ssim() ในการดูค่า ซึ่งค่าที่ออกมานั้นในการเทียบครั้งแรก จากการเทียบรูปต้นฉบับและรูปที่มี Noise พบร่วมค่า SSIM มีค่าน้อย ซึ่งหมายถึงภาพมีความต่างกันแต่พอหลังจากการ Denoise ไปเรื่อย ๆ พบร่วมค่า SSIM สูงขึ้น เนื่องจากภาพเริ่มมีความใกล้เคียงกัน ตัวอย่างรูปด้านล่าง

```
● k. @Chonakans-MacBook-Air Classworks % /usr/local/bin/python3 /Users/k./Documents/Junior/Image-processing/Classworks/classwork3.py
Enter salt density: 0.1
Enter paper density: 0.1
SSIM : 0.24401541043449756
SSIM after denoise by 1 : 0.24401541043449756
SSIM after denoise by 3 : 0.6171606586397382
● k. @Chonakans-MacBook-Air Classworks % /usr/local/bin/python3 /Users/k./Documents/Junior/Image-processing/Classworks/classwork3.py
Enter salt density: 0.3
Enter paper density: 0.3
SSIM : 0.10138638851994773
SSIM after denoise by 1 : 0.10138638851994773
SSIM after denoise by 3 : 0.28746632898929847
SSIM after denoise by 5 : 0.4533406428350322
○ k. @Chonakans-MacBook-Air Classworks %
```

โดยค่า SSIM ที่ได้สามารถนำมาเช็คในการหาได้ว่า แต่ละรูปภาพที่มี Noise แตกต่างกัน ต้องใช้การ Denoise (Median Blur) กี่รอบถึงจะใกล้เคียงกับภาพต้นฉบับมากที่สุด

ซึ่งผู้ได้สร้างฟังก์ชัน find\_max\_ssim() มาใช้ในการหา max ในการ denoise ของแต่ละภาพ โดยภายในได้เลือกใช้ for loop รัน 4 รอบ (assume ว่ามีภาพเบลอค่า SSIM ยิ่งน้อย เลยเบลอแค่ Filter size 3, 5, 7, 9) เก็บค่า SSIM ของแต่ละการ Filter size และนำมาเช็คว่าในช่วง Filter size ไหนที่ได้ค่า SSIM มากที่สุด ซึ่งหมายถึงใกล้เคียงรูปภาพมากที่สุดนั่นเอง

```
def find_max_ssim(noise_free_image, noisy_image):
    quality_range = []
    ssim_value_og = compare_ssim(noise_free_image, noisy_image)

    for i in range(3, 10):
        denoise = cv2.medianBlur(noisy_image, i)
        ssim_value = compare_ssim(noise_free_image, denoise)
        quality_range.append(ssim_value)

    return max(quality_range), ssim_value_og
```

ทำให้ยิ่งถ้าหากค่า Density ของ Noise มาก รอบในการ Denoise ก็จะยิ่งมากตามลำดับ



```
● k. @Chonakans-MacBook-Air Classworks % /usr/local/bin/python3 /Users/k./Documents/Junior/Image-processing/Classworks/classwork3.py
Enter salt density: 0.1
Enter paper density: 0.1

Max quality : 0.6145390660200549
SSIM : 0.24611351849866517
SSIM after denoise by 1 : 0.24611351849866517
SSIM after denoise by 3 : 0.6145390660200549
○ k. @Chonakans-MacBook-Air Classworks %
```

ซึ่งจากตัวอย่างนี้ทำการใช้แค่ Filter 3x3 ในการ denoise เพราะถ้าหากใช้ Filter ใหญ่ไป  
มากกว่านี้จะทำให้ภาพเสียความคมชัดและไม่ใกล้เคียงกับภาพต้นฉบับไป

## Source Code:

<https://github.com/Kariusdi/Image-processing/blob/main/Classworks/classwork3.py>

```
Classworks > 📸 classwork3.py > ⚡ denoiser
  1 import cv2
  2 import numpy as np
  3 import random
  4 from skimage.metrics import structural_similarity as compare_ssim
  5
  6 def salt_and_pepper_noisel(img, dstSalt, dstPepper):
  7
  8     density_salt = dstSalt
  9     density_pepper = dstPepper
 10
 11     number_of_white_pixel = int(density_salt * (img.shape[0] * img.shape[1]))
 12
 13     for i in range(number_of_white_pixel):
 14         y_coord = random.randint(0, img.shape[0]-1)
 15         x_coord = random.randint(0, img.shape[1]-1)
 16         img[y_coord][x_coord] = 255
 17
 18
 19     number_of_black_pixel = int(density_pepper * (img.shape[0] * img.shape[1]))
 20
 21     for i in range(number_of_black_pixel):
 22         y_coord = random.randint(0, img.shape[0]-1)
 23         x_coord = random.randint(0, img.shape[1]-1)
 24         img[y_coord][x_coord] = 0
 25
 26
 27     return img
 28
 29 def find_max_ssim(noise_free_image, noisy_image):
 30     quality_range = []
 31     ssim_value_og = compare_ssim(noise_free_image, noisy_image)
 32
 33     for i in range(3, 10, 2):
 34         denoise = cv2.medianBlur(noisy_image, i)
 35         ssim_value = compare_ssim(noise_free_image, denoise)
 36         quality_range.append(ssim_value)
 37
 38     return max(quality_range), ssim_value_og
 39
 40 def denoiser(noise_free_image, noisy_image):
 41
 42     filter_size = 1
 43     max_quality, ssim_value = find_max_ssim(noise_free_image, noisy_image)
 44
 45     print(f"\nMax quality : {max_quality}")
 46     print(f"SSIM : {ssim_value}")
 47
 48     filter_size = 1
 49
 50     while ssim_value != max_quality:
 51
 52         denoise = cv2.medianBlur(noisy_image, filter_size)
 53         ssim_value = compare_ssim(noise_free_image, denoise)
 54
 55         print(f"SSIM after denoise by {filter_size} : {ssim_value}")
 56
 57         compare_pics = np.concatenate((noise_free_image, denoise), axis=1)
 58         cv2.imshow(f'Original and Denoise {filter_size}', compare_pics)
 59         cv2.waitKey(1250)
 60         cv2.destroyAllWindows()
 61
 62         filter_size += 2
 63
 64     cv2.imwrite('pics/denoise.jpg', denoise)
 65
 66
 67     return 0
 68
 69 if __name__ == '__main__':
 70
 71     image = cv2.imread('pics/DogCat.jpg', 0)
 72     density_salt = float(input("Enter salt density: "))
 73     density_paper = float(input("Enter paper density: "))
 74
 75     noisy = salt_and_pepper_noisel(image, density_salt, density_paper)
 76     cv2.imwrite('pics/noisy.jpg', noisy)
 77
 78     noise_free_image = cv2.imread('pics/Dogcat.jpg', 0)
 79     cv2.imwrite('pics/DogCatGrayscale.jpg', noise_free_image)
 80     noisy_image = cv2.imread('pics/noisy.jpg', 0)
 81
 82     denoiser(noise_free_image, noisy_image)
```