HW8
r10944013 網媒所碩一 馮啟倫

HW8:

目標：產生24組，noise跟filter排列組合的照片

Code segment

#Gaussian_noise_generator

```
def Gaussian_noise_10(img):
    img_noise = copy.deepcopy(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            noisepixel = int(img[i][j] + 10*random.gauss(0,1))
            if noisepixel > 255:
                noisepixel = 255
            img_noise[i][j] = noisepixel
    return img_noise

def Gaussian_noise_30(img):
    img_noise = copy.deepcopy(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            noisepixel = int(img[i][j] + 30*random.gauss(0,1))
            if noisepixel > 255:
                noisepixel = 255
            img_noise[i][j] = noisepixel
    return img_noise
```

#Salt and pepper generator

```
def Saltandpepper_05(img):
    img_noise = copy.deepcopy(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            random_value = random.uniform(0,1)
            if random_value <=0.05:
                img_noise[i][j] = 0
            elif random_value >= (1-0.05):
                img_noise[i][j] = 255
            else:
                pass
    return img_noise

def Saltandpepper_10(img):
    img_noise = copy.deepcopy(img)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            random_value = random.uniform(0,1)
            if random_value <=0.1:
                img_noise[i][j] = 0
            elif random_value >= (1-0.1):
                img_noise[i][j] = 255
            else:
                pass
    return img_noise
```

#box filter

```
def box_filter_3(img):
    img_pad = cv2.copyMakeBorder(img,1,1,1,1,cv2.BORDER_REPLICATE)
    img_clean = np.zeros((512,512))
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            center_row = row+1
            center_col = col+1
            score = 0
            for i in range(-1,2): #-1,0,1
                for j in range(-1,2):
                    score += img_pad[center_row+i][center_col+j]
            score /= 9
            img_clean[row][col] = score
    return img_clean

def box_filter_5(img):
    img_pad = cv2.copyMakeBorder(img,2,2,2,2,cv2.BORDER_REPLICATE)
    img_clean = np.zeros((512,512))
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            center_row = row+2
            center_col = col+2
            score = 0
            for i in range(-2,3): #-2,-1,0,1,2
                for j in range(-2,3):
                    score += img_pad[center_row+i][center_col+j]
            score /= 25
            img_clean[row][col] = score
    return img_clean
```

#median filter

```
def median_filter_3(img):
    img_pad = cv2.copyMakeBorder(img,1,1,1,1,cv2.BORDER_REPLICATE)
    img_clean = np.zeros((512,512))
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            center_row = row+1
            center_col = col+1
            score = []
            for i in range(-1,2): #-1,0,1
                for j in range(-1,2):
                    score.append(img_pad[center_row+i][center_col+j])
            img_clean[row][col] = sorted(score)[4] #median
    return img_clean

def median_filter_5(img):
    img_pad = cv2.copyMakeBorder(img,2,2,2,2,cv2.BORDER_REPLICATE)
    img_clean = np.zeros((512,512))
    for row in range(img.shape[0]):
        for col in range(img.shape[1]):
            center_row = row+2
            center_col = col+2
            score = []
            for i in range(-2,3): #-2,-1,0,1,2
                for j in range(-2,3):
                    score.append(img_pad[center_row+i][center_col+j])
            img_clean[row][col] = sorted(score)[12]
    return img_clean
```

#open, closing沿用之前HW的，礙於版面就不放了

Result picture:

Salt and pepper - probability = 0.1

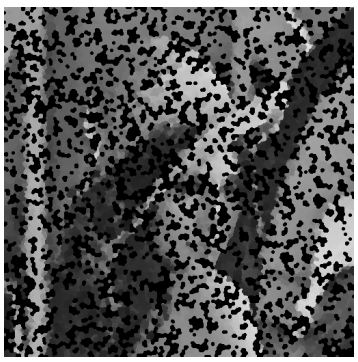
salt and pepper
with prob=0.1
SNR= -2.1207

median_3x3
SNR= 14.566

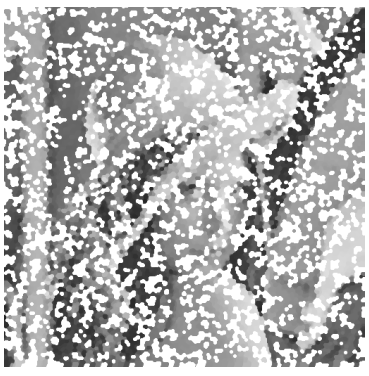
median_5x5
SNR= 15.737

box_3x3
SNR= 6.310

box_5x5
SNR= 8.492



open_close
SNR= -2.416



close_open
SNR= -2.638

Salt and pepper - probability = 0.05

salt and pepper
with prob=0.05
SNR= 1.001

median_3x3
SNR= 19.166

median_5x5
SNR= 16.356

box_3x3
SNR= 9.529

box_5x5
SNR= 11.201

open_close
SNR= 4.405

close_close
SNR= 3.654

Gaussian noise - amplitude = 10

gaussian
amplitude=10
SNR= 13.589

median_3x3
SNR= 17.66

median_5x5
SNR= 16.01

box_3x3
SNR= 17.74

box_5x5
SNR= 14.87

open_close
SNR= 6.95

close_open
SNR= 6.91

Gaussian noise - amplitude = 30

gaussian
amplitude=30
SNR= 2.795

median_3x3
SNR= 10.86

median_5x5
SNR= 12.63

box_3x3
SNR= 10.78

box_5x5
SNR= 11.75

open_close
SNR= 6.77

close_open
SNR= 5.85