Computer Vision Homework 8

R08922079 資工所一 洪浩翔

Part 0

```python
def readImg(filename='lena.bmp'):
    #read img
    image = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    return image

def snr(origImg, TargImg):
    origImg = origImg / 255.0
    TargImg = TargImg / 255.0
    meanOrig = np.mean(origImg)
    VS = np.mean(np.power(origImg - meanOrig, 2))
    meanTarg = np.mean(TargImg - origImg)
    VN = np.mean(np.power(TargImg - origImg - meanTarg, 2))
    return 20.0*np.log10(np.sqrt(VS)/np.sqrt(VN))
```

```python
def ImgPreProcess():
    #kernel
    Dilkernel = np.array([[0,1,1,1,0],[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1],[0,1,1,1,0]])
    Erokernel = np.array([[255,1,1,1,255],[1,1,1,1,1],[1,1,1,1,1],[1,1,1,1,1],[255,1,1,1,255]])

    return Dilkernel, Erokernel
```

Part 1

Gaussian noise with amp = 10

Snr = 13.939527234532092

3x3 box filter

Snr = 16.44829523717587

5x5 box filter

Snr = 13.610520190975684

3x3 median filter

Snr = 17.8451064589486

5x5 median filter

Snr = 15.926188281486

Close then open

Snr = 13.629515221968312

Open then close

Snr = 13.276470198884061

Gaussian noise with amp = 30

Snr = 2.2587622315879003

3x3 box filter

Snr = 9.678335626876832

5x5 box filter

Snr = 10.341085646093909

3x3 median filter

Snr = 10.87033698053484

5x5 median filter

Snr = 12.328056376705522

Close then open

Snr = 7.83100571664224

Open then close

Snr = 7.924666447669075

Salt and pepper noise with prob = 0.05

Snr = 0.8566596160998019

3x3 box filter

Snr = 9.187921902215123

5x5 box filter

Snr = 10.520613442040286

3x3 median filter

Snr = 18.18777513278135

5x5 median filter

Snr = 15.7546116437436

Close then open

Snr = 5.2034486799958195

Open then close

Snr = 1.2508035238354342

Salt and pepper noise with prob = 0.1

Snr = -2.1076175845166483

3x3 box filter

Snr = 6.196176652696673

5x5 box filter

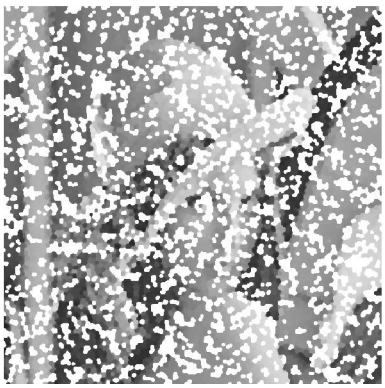
Snr = 8.149110532880508

3x3 median filter

Snr = 14.172461603058151

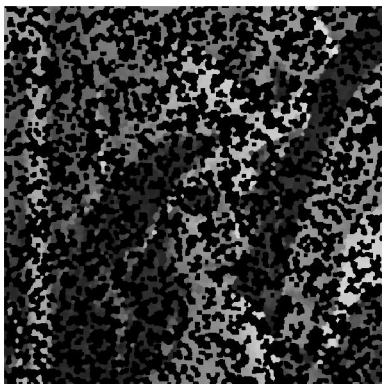
5x5 median filter

Snr = 14.229670526769135



Close then open

Snr = -2.4451848886114664



Open then close

Snr = -2.984293812855861

Code

```python
def gaussianNoise(img, amp):
    result = img.copy()
    w,h = img.shape
    for i in range(w):
        for j in range(h):
            result[i,j] = img[i,j] + int(amp * random.gauss(0,1))
    result = np.clip(result, 0, 255)
    cv2.imwrite("gaussNoise_"+str(amp)+".jpg", result)
    return result


def saltNoise(img, prob):
    result = img.copy()
    w,h = img.shape
    for i in range(w):
        for j in range(h):
            noise = random.uniform(0,1)
            if noise < prob:
                result[i,j] = 0
            elif noise > (1-prob):
                result[i,j] = 255
            else:
                result[i,j] = img[i,j]
    cv2.imwrite("saltnpepperNoise_"+str(prob).split('.')[1]+".jpg", result)
    return result
```

```python
def boxFilter(img, kernel, index):
    result = img.copy()
    w,h = result.shape
    border = int(kernel/2)
    img = np.pad(img, ((border,border),(border,border)), 'constant', constant_values=0)
    for i in range(w):
        for j in range(h):
            result[i,j] = int(np.sum(img[i:i+1+2*border,j:j+1+2*border])/kernel**2)
    cv2.imwrite("boxF_"+str(kernel)+"_"+str(index)+".jpg", result)
    return result

def medianFilter(img, kernel, index):
    result = img.copy()
    w,h = result.shape
    border = int(kernel/2)
    img = np.pad(img, ((border,border),(border,border)), 'constant', constant_values=0)
    for i in range(w):
        for j in range(h):
            tmp = img[i:i+1+2*border,j:j+1+2*border].flatten()
            tmp = np.sort(tmp).tolist()
            place = int(len(tmp)/2)
            result[i,j] = tmp[place]
    cv2.imwrite("medianF_"+str(kernel)+"_"+str(index)+".jpg", result)
    return result
```

```python
def dilation(image, kernel, shape):
    #dilation on gray image
    dilImg = np.zeros(shape)
    w,h = image.shape
    for i in range(2,w-2):
        for j in range(2,h-2):
            dilImg[i-2, j-2] = np.amax(image[i-2:i+3, j-2:j+3]*kernel)
    return dilImg

def erosion(image, kernel, shape):
    #erosion on gray image
    eroImg = np.zeros(shape)
    w,h = image.shape
    for i in range(2,w-2):
        for j in range(2,h-2):
            eroImg[i-2, j-2] = np.amin(image[i-2:i+3, j-2:j+3]*kernel)
    return eroImg

def opening(image, Dilkernel, Erokernel, shape):
    #erosion than dilation
    image = np.pad(image, ((2,2),(2,2)), 'constant', constant_values=255)
    eroImg = erosion(image, Erokernel, shape)
    eroImg = np.pad(eroImg, ((2,2),(2,2)), 'constant', constant_values=0)
    openImg = dilation(eroImg, Dilkernel, shape)
    return openImg

def closing(image, Dilkernel, Erokernel, shape):
    #dilation than erosion
    image = np.pad(image, ((2,2),(2,2)), 'constant', constant_values=0)
    dilImg = dilation(image, Dilkernel, shape)
    dilImg = np.pad(dilImg, ((2,2),(2,2)), 'constant', constant_values=255)
    closImg = erosion(dilImg, Erokernel, shape)
    return closImg
```

```python
def openThenClose(img, Dilkernel, Erokernel, shape, index):
    openImg = opening(img, Dilkernel, Erokernel, shape)
    closImg = closing(openImg, Dilkernel, Erokernel, shape)
    cv2.imwrite("openThenClose_"+str(index)+".jpg", closImg)
    return closImg

def closeThenOpen(img, Dilkernel, Erokernel, shape, index):
    closImg = closing(img, Dilkernel, Erokernel, shape)
    openImg = opening(closImg, Dilkernel, Erokernel, shape)
    cv2.imwrite("closeThenOpen_"+str(index)+".jpg", openImg)
    return openImg
```

```python
def main():
    img = readImg()
    shape = img.shape
    amp = [10, 30]
    gauss = []
    for i in amp:
        gauss.append(gaussianNoise(img.copy(), i))
        print("gauss {} snr: {}".format(i, snr(img, gauss[-1])))

    prob = [0.1, 0.05]
    salt = []
    for i in prob:
        salt.append(saltNoise(img.copy(), i))
        print("saltPepper {} snr: {}".format(i, snr(img, salt[-1])))
    kernel = [3,5]

    box = []
    for i in kernel:
        for j,k in zip(gauss, amp):
            box.append(boxFilter(j.copy(), i, k))
            print("box {} gauss {} snr: {}".format(i, k, snr(img, box[-1])))
    for i in kernel:
        for j,k in zip(salt, prob):
            box.append(boxFilter(j.copy(), i, k))
            print("box {} salt {} snr: {}".format(i, k, snr(img, box[-1])))

    median = []
    for i in kernel:
        for j,k in zip(gauss, amp):
            median.append(medianFilter(j.copy(), i, k))
            print("median {} gauss {} snr: {}".format(i, k, snr(img, median[-1])))
    for i in kernel:
        for j,k in zip(salt, prob):
            median.append(medianFilter(j.copy(), i, k))
            print("median {} salt {} snr: {}".format(i, k, snr(img, median[-1])))
```

```python
    openThenCloseImgs = []
    closeThenOpenImgs = []
    Dilkernel, Erokernel = ImgPreProcess()
    for j,k in zip(gauss, amp):
        openThenCloseImgs.append(openThenClose(j.copy(), Dilkernel, Erokernel, shape, k))
        print("openThenClose gauss {} snr: {}".format(k, snr(img, openThenCloseImgs[-1])))
        closeThenOpenImgs.append(closeThenOpen(j.copy(), Dilkernel, Erokernel, shape, k))
        print("closeThenOpen gauss {} snr: {}".format(k, snr(img, closeThenOpenImgs[-1])))

    for j,k in zip(salt, prob):
        openThenCloseImgs.append(openThenClose(j.copy(), Dilkernel, Erokernel, shape, k))
        print("openThenClose salt {} snr: {}".format(k, snr(img, openThenCloseImgs[-1])))
        closeThenOpenImgs.append(closeThenOpen(j.copy(), Dilkernel, Erokernel, shape, k))
        print("closeThenOpen salt {} snr: {}".format(k, snr(img, closeThenOpenImgs[-1])))


if __name__ == "__main__":
    main()
```