Computer Vision

NTUT 資訊工程所 111598070 賴俊霖

Homework 2

1. Mean filter (save as : output1.png)

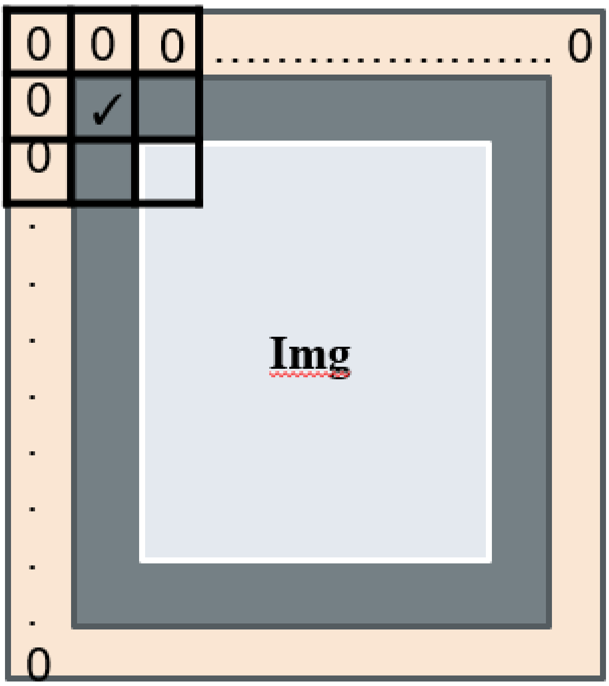
In meanFilter(img),you can see

Using n,m,_ = img.shape to get length,width and channel’s data

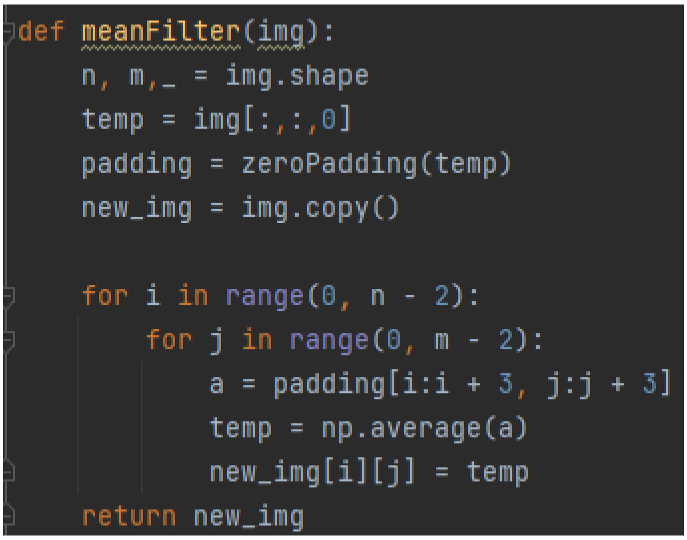
And img[:,:,0] to get a single channel and store in temp.

Then we copy an original img to take as a base.

We using zeroPadding to make the temp like pic below.



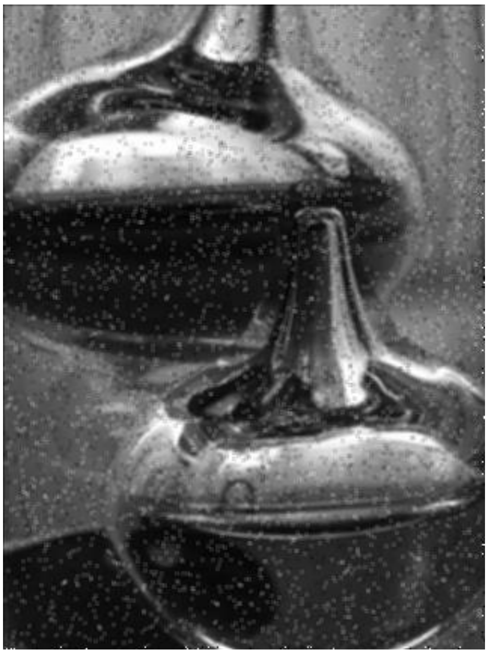
And we get the 3x3 of zeroPadding we created and use np.average(a) to get the Avg and replace the center Value of the position a which exactly the new_img[i][j] we copied.



```
def meanFilter(img):
    n, m,_ = img.shape
    temp = img[:,:,0]
    padding = zeroPadding(temp)
    new_img = img.copy()

    for i in range(0, n - 2):
        for j in range(0, m - 2):
            a = padding[i:i + 3, j:j + 3]
            temp = np.average(a)
            new_img[i][j] = temp
    return new_img
```

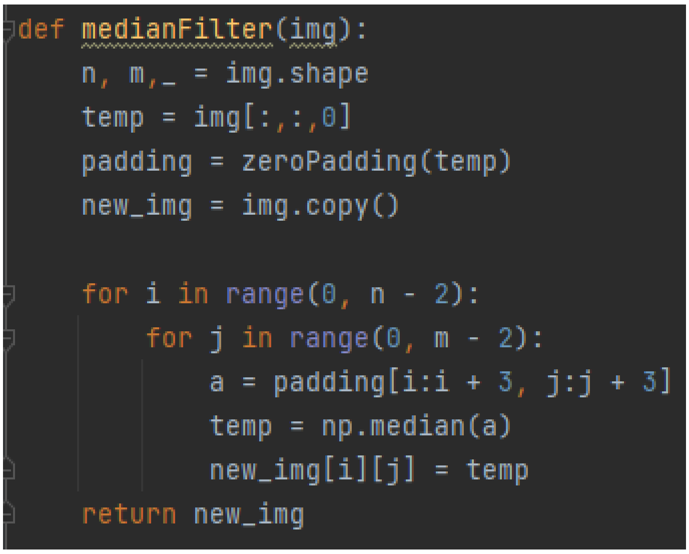
Images conversion as follow



2. **Median filter** (save as : output2.png)

The logic of Median filter is the same as Mean filter.

The only difference is using np.median(a) not np.average.



```python
def medianFilter(img):
    n, m,_ = img.shape
    temp = img[:,:,0]
    padding = zeroPadding(temp)
    new_img = img.copy()

    for i in range(0, n - 2):
        for j in range(0, m - 2):
            a = padding[i:i + 3, j:j + 3]
            temp = np.median(a)
            new_img[i][j] = temp
    return new_img
```

Images conversion as follow

3. **Image histogram** : Count the number of each pixel intensity.

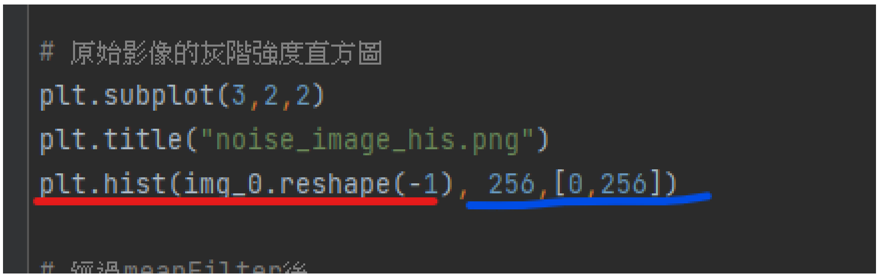
(save as : noise_image_his.png, output1_his.png, output2_his.png)

In Python ,we can easily get the hist using

img = img[:,:,0] to get single channel ,So that the statistic does not triple.

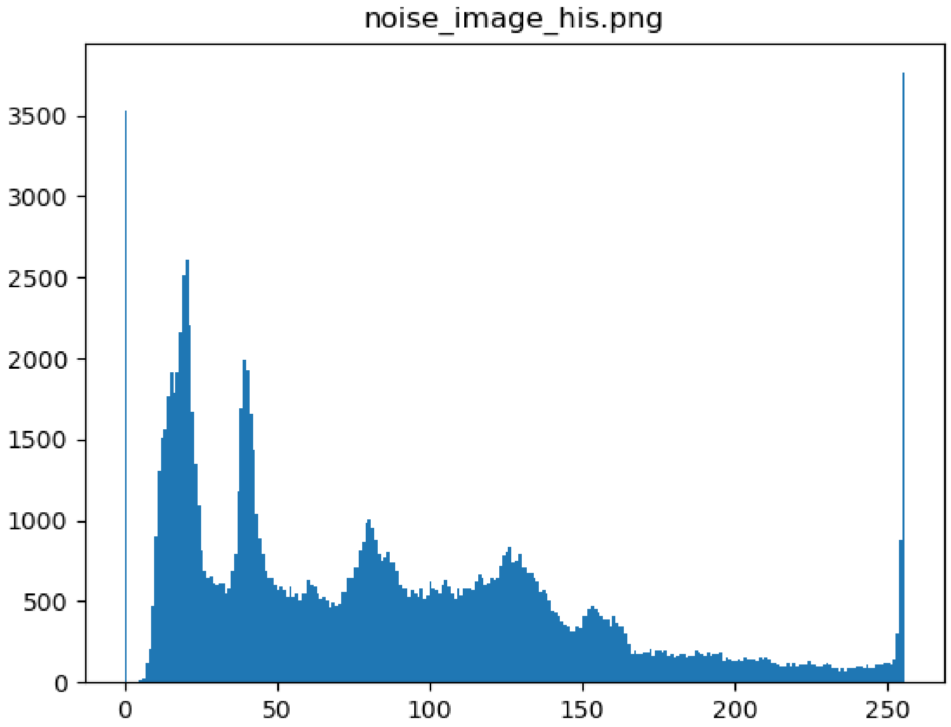
reshape(-1) is for converting 3D matrix to 1D list.

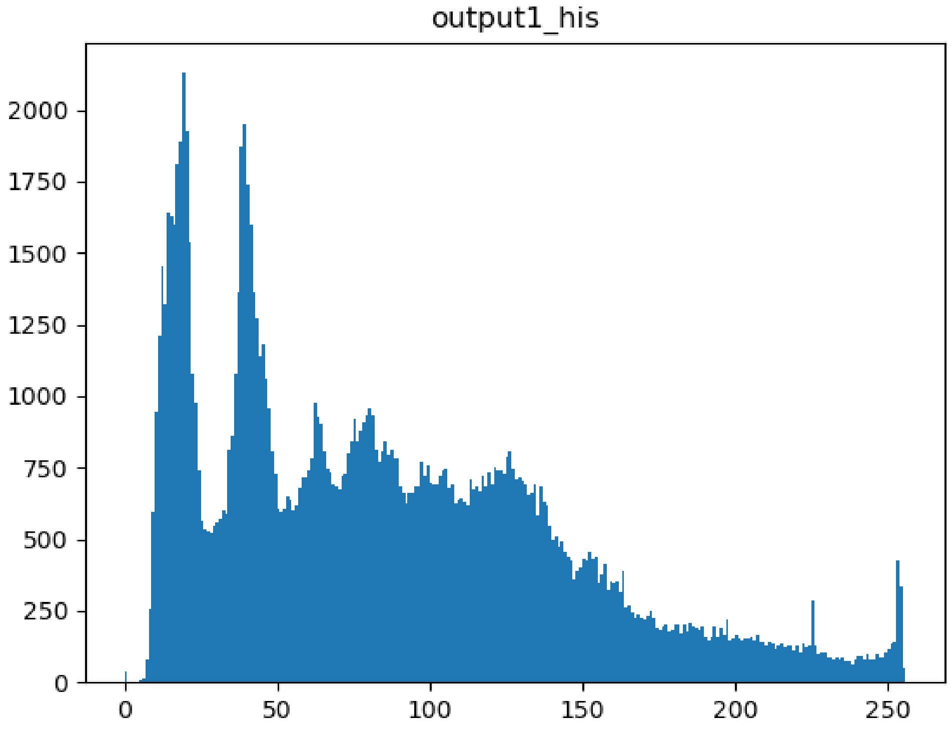
And we fixed the Minimum value and Maximum of coordinates.

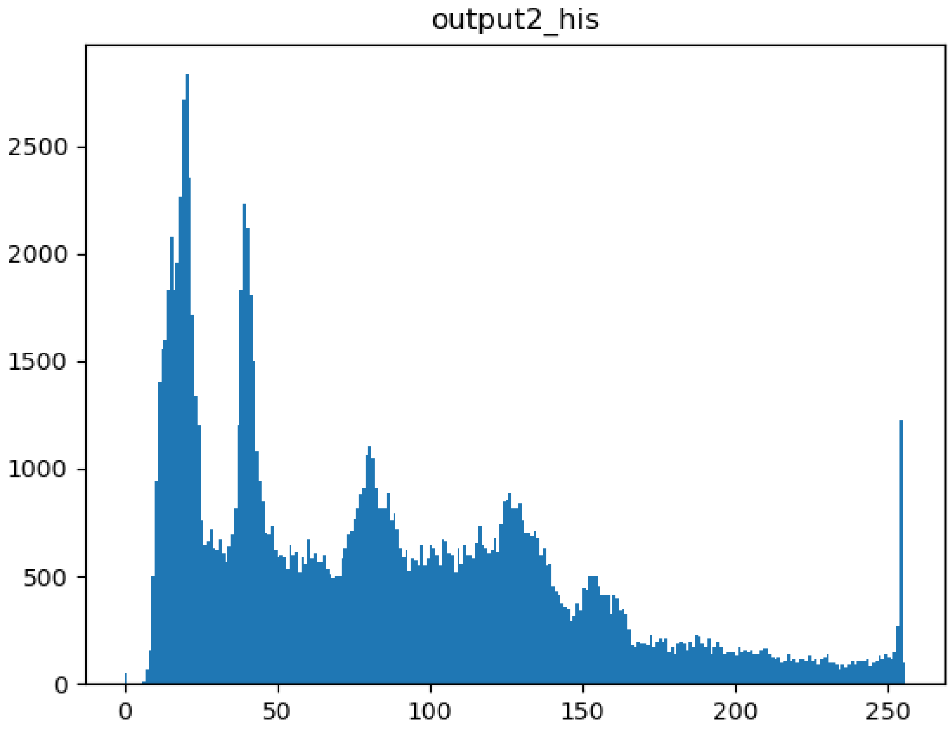


```
# 原始影像的灰階強度直方圖
plt.subplot(3,2,2)
plt.title("noise_image_his.png")
plt.hist(img_0.reshape(-1), 256,[0,256])

# 經過meanFilter後
```


noise_image_his.png
3500
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250


output1_his
2000
1750
1500
1250
1000
750
500
250
0
0
50
100
150
200
250



Instead of relying on the above approach, we can write our own as

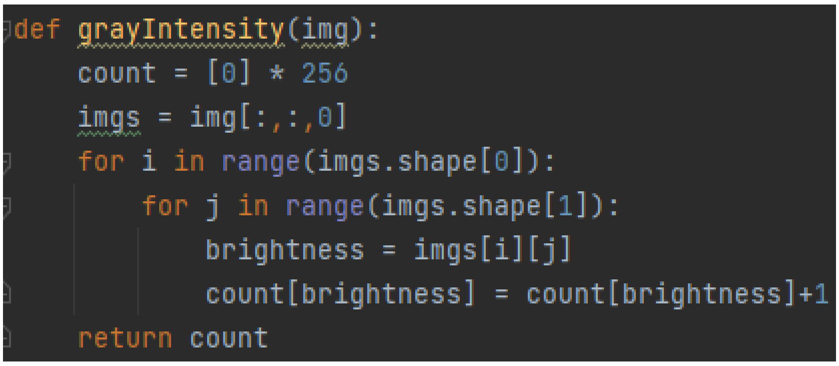
grayIntensity(img)

Count = [0] * 256 to declare a 1-dimensional list with all 0's

Imgs = img[:,:,0] as mentioned before avoiding 3 times the number of counts.

We visited every point of this image and go through 0~255, then go to the count we created , use the number we just got as the index to do incremental.

Then return count to make the histogram.



```
def grayIntensity(img):
    count = [0] * 256
    imgs = img[:,:,0]
    for i in range(imgs.shape[0]):
        for j in range(imgs.shape[1]):
            brightness = imgs[i][j]
            count[brightness] = count[brightness]+1
    return count
```

4. **Compare** : noise_image_his.png, output1_his.png, output2_his.png

(Text description)

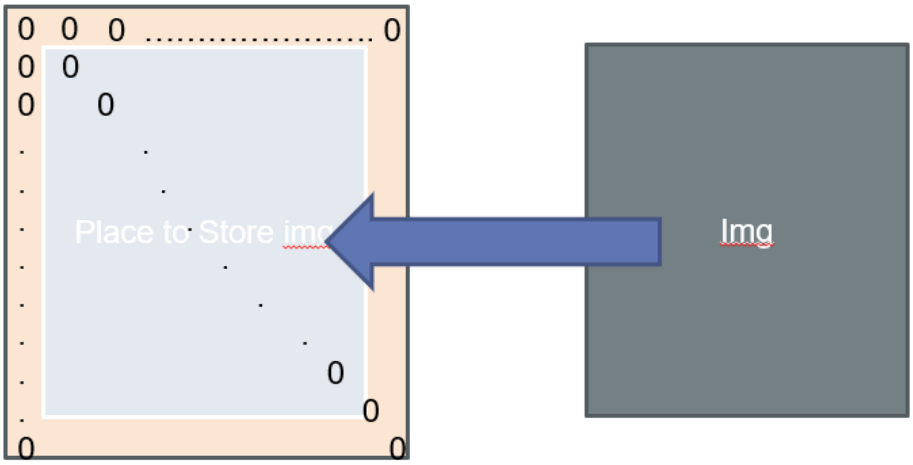
According to the original histogram, we can find that the pepper salt noise problem causes our values at 0 and 255 to be unusually high.

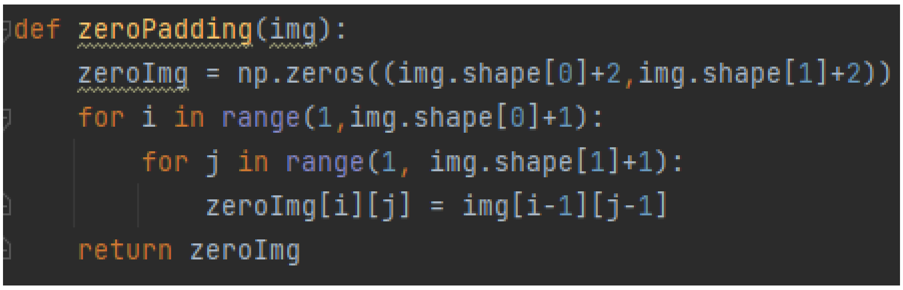
Using both the mean filter and the median filter, we can find that the part at 0 drops significantly and is quite close to 0.

Based on the previous output of Median Filter, we can also know that the image seems to be more suitable for using Median Filter as a denoise filter .

5. 補充 zeroPadding

In zeroPadding, first we created a bigger 2D matrix is full of zero, and stored the original pixel values to the new img from i=1 to (shape[0]-1) or (j=1 to shape[1]-1)





```
def zeroPadding(img):
    zeroImg = np.zeros((img.shape[0]+2,img.shape[1]+2))
    for i in range(1,img.shape[0]+1):
        for j in range(1, img.shape[1]+1):
            zeroImg[i][j] = img[i-1][j-1]
    return zeroImg
```