1.4.7 Median filtering

Median filtering: image smoothing can eliminate salt pepper noise. The basic idea is to traverse the image through a filter and take the median of the pixel values of each filter area as the new pixel value.

The algorithm idea is as follows:
(1) Input image and convert to grayscale image;
(2) Add salt pepper noise to the grayscale image;
(3) Traverse the pixels and put the pixel values in the filter area into a one-dimensional array;
(4) Selectively sort a one-dimensional array, and assign the median value to the center of the filter, that is, change the pixels of the traversed original image to the median of the filter area;
(5) Output the image after median filtering.

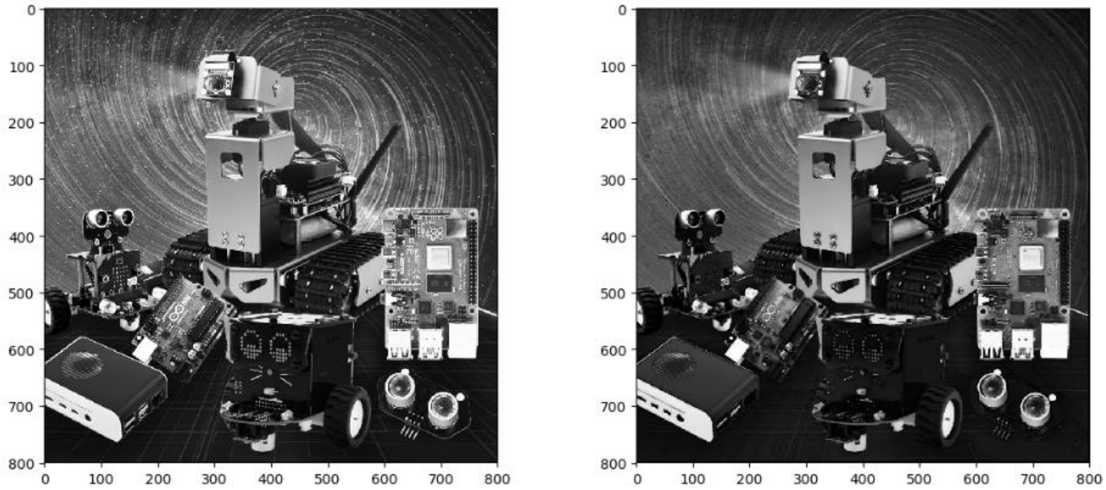
Code path:
/home/pi/Yahboom_Project/Raspbot/1.OpenCV_course/04image_beautification/07_Median_ filtering.ipynb

```
# Median filtering 3*3

import cv2

import numpy as np

import matplotlib.pyplot as plt



img = cv2.imread('yahboom.jpg',1)

imgInfo = img.shape

height = imgInfo[0]

width = imgInfo[1]

img = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)

# cv2.imshow('src',img)

dst = np.zeros((height,width,3),np.uint8)
```

```
collect = np.zeros(9,np.uint8)
for i in range(1,height-1):
    for j in range(1,width-1):
        k = 0
        for m in range(-1,2):
            for n in range(-1,2):
                gray = img[i+m,j+n]
                collect[k] = gray
                k = k+1
        # 0 1 2 3 4 5 6 7 8
        #     1
        for k in range(0,9):
            p1 = collect[k]
            for t in range(k+1,9):
                if p1<collect[t]:
                    mid = collect[t]
                    collect[t] = p1
                    p1 = mid
        dst[i,j] = collect[4]
# cv2.imshow('dst',dst)
# cv2.waitKey(0)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
dst = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(14, 6), dpi=100) #Set the size and pixels of the drawing area

plt.subplot(121)    # The first in a row and two columns

plt.imshow(img)

plt.subplot(122)    # The second in a row and two columns

plt.imshow(dst)

plt.show()
```

After running the above program, two pictures will be displayed in the jupyterLab control interface, as shown below.


0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600
700
800