

Lesson 9 Image Processing - Smoothing

1. Image Noise

During collecting, processing and transferring, the digital image will be disturbed by different noises, which leads to low-quality image, obscure image and disappeared image feature. And image smoothing is to improve the image by removing noise, and salt-and-pepper noise as well as Gauss noise are common.

1.1 Salt-and-pepper Noise

Salt-and-pepper noise is also known as pulse noise which is white dots and black dots appearing randomly, like there are black pixels in bright area and white pixels in dark area.

At left is the original picture and the right is the picture with salt-and-pepper noise.



1.2 Gauss Noise

Gauss noise is a term from signal processing theory denoting a kind of signal noise that has a probability density function (pdf) equal to that of the normal distribution (which is also known as the Gaussian distribution). Commonly, it can be suppressed by mathematical statistics.

At left is the original picture and the right is the picture with Gauss noise.



2. Image Smoothing

From the perspective of signal, image smoothing is to filter the high frequency part of the signal and reserve the low frequency part.

Based on filter, filtering can be divided into mean filtering, Gaussian filtering and median filtering.

2.1 Mean Filtering

The idea of mean filtering is simply to take the mean of all the pixels of the image that is assign the mean of all the pixels in the unit of a square to the center pixel.

Take the picture below as example. In picture (a), the center pixel value is

“226” and the mean of all the pixels is “122” obtained from the equation below, and “122” is the new center pixel value.

$$(40 + 107 + 5 + 198 + 226 + 223 + 37 + 68 + 193) \div 9 = 122$$

Replace the original center pixel value by “122” as the picture (b) shown.

40	107	5
198	226	223
37	68	193

(a)

40	107	5
198	122	223
37	68	193

(b)

The algorithm of the mean filtering is simple and its calculation speed is fast. However, the details of the image are destroyed during removing noise resulting in low definition.

2.2 Gauss Filtering

The weighted mean is calculated by multiplying each value by the corresponding weight, adding the sum, and then dividing the sum by the number of the values.

Gauss filtering is to obtain the weighted mean of all the pixels of the image that is assign the weighted mean in the unit of a square to the center pixel.

Take the picture below as example. In picture (a), the center pixel value is “226” and the weighted mean of all the pixels is “164” obtained from the equation below, and “164” is the new center pixel value.

$$40 \times 0.05 + 107 \times 0.1 + 5 \times 0.05 + 198 \times 0.1 + 226 \times 0.4 + 223 \times 0.1 + 37 \times 0.05 + 68 \times 0.1 + 193 \times 0.05 = 164$$

Replace the original center pixel value by “122” as the picture (c) shown.

<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>40</td><td>107</td><td>5</td></tr> <tr><td>198</td><td style="background-color: #f4a460;">226</td><td>223</td></tr> <tr><td>37</td><td>68</td><td>193</td></tr> </table> <p>(a)</p>	40	107	5	198	226	223	37	68	193	×	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>0.05</td><td>0.1</td><td>0.05</td></tr> <tr><td>0.1</td><td>0.4</td><td>0.1</td></tr> <tr><td>0.05</td><td>0.1</td><td>0.05</td></tr> </table> <p>(b)</p>	0.05	0.1	0.05	0.1	0.4	0.1	0.05	0.1	0.05	=	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>40</td><td>107</td><td>5</td></tr> <tr><td>198</td><td style="background-color: #f4a460;">164</td><td>223</td></tr> <tr><td>37</td><td>68</td><td>193</td></tr> </table> <p>(c)</p>	40	107	5	198	164	223	37	68	193
40	107	5																													
198	226	223																													
37	68	193																													
0.05	0.1	0.05																													
0.1	0.4	0.1																													
0.05	0.1	0.05																													
40	107	5																													
198	164	223																													
37	68	193																													

2.3 Median Filtering

The median is the middle value when a data set is ordered from least to greatest.

Median filtering is to take the median of all the pixels of the image that is assign the median in the unit of square to the center pixel.

Take the picture below as example. In picture (a), the center pixel value is “226” and the medium of the pixels is “**107**”, and “**107**” is the new center pixel value.

Replace the original center pixel value by “107” as the picture (b) shown.

<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>40</td><td>107</td><td>5</td></tr> <tr><td>198</td><td style="background-color: #f4a460;">226</td><td>223</td></tr> <tr><td>37</td><td>68</td><td>193</td></tr> </table> <p>(a)</p>	40	107	5	198	226	223	37	68	193	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>40</td><td>107</td><td>5</td></tr> <tr><td>198</td><td style="background-color: #f4a460;">107</td><td>223</td></tr> <tr><td>37</td><td>68</td><td>193</td></tr> </table> <p>(b)</p>	40	107	5	198	107	223	37	68	193
40	107	5																	
198	226	223																	
37	68	193																	
40	107	5																	
198	107	223																	
37	68	193																	

3.Operation Steps


This routine will execute mean filtering, Gauss filtering and median

filtering separately.

Before operation, please copy the routine “**filtering.py**” in “4.OpenCV->Lesson 9 Image Processing --- Smoothing->Routine Code” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

Note: the input command should be case sensitive and the keywords can be complemented by “Tab” key.

1) Open virtual machine and start the system. Click “

2) Input command “**cd /mnt/hgfs/Share/**” and press Enter to enter the shared folder.

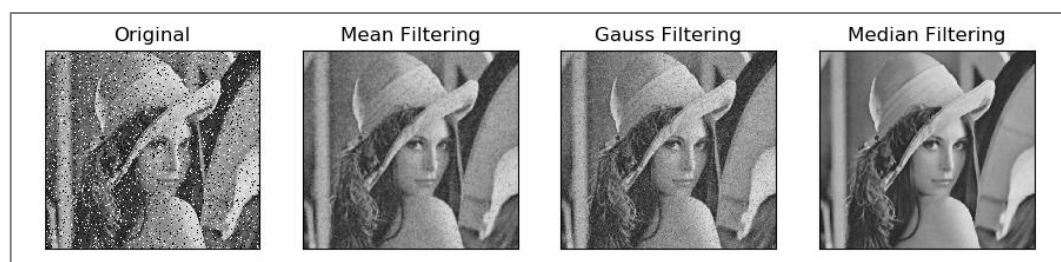
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 filtering.py**” and press Enter to run the routine.

```
hiwonder@ubuntu:/mnt/hgfs/Share$ python3 filtering.py
```

4.Program Outcome

The final output image is as follow.



5.Program Analysis

The routine “**filtering.py**” can be found in “4.OpenCV->Lesson 9 Image Processing --- Smoothing->Routine Code”.

```

1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # image reading
6  img = cv2.imread('noise.jpg')
7
8  # image smoothing
9  blur1 = cv2.blur(img, (5, 5))    # mean filtering
10 blur2 = cv2.GaussianBlur(img, (5, 5), 1)    # Gauss filtering
11 blur3 = cv2.medianBlur(img, 5)    # Median filtering
12
13 # image display
14 plt.figure(figsize=(10, 5), dpi=100)
15 plt.rcParams['axes.unicode_minus'] = False
16 plt.subplot(141), plt.imshow(img), plt.title("Original")
17 plt.xticks([], plt.yticks([]))
18 plt.subplot(142), plt.imshow(blur1), plt.title("Mean Filtering")
19 plt.xticks([], plt.yticks([]))
20 plt.subplot(143), plt.imshow(blur2), plt.title("Gauss Filtering")
21 plt.xticks([], plt.yticks([]))
22 plt.subplot(144), plt.imshow(blur3), plt.title("Median Filtering")
23 plt.xticks([], plt.yticks([]))
24 plt.show()

```

5.1 Image Processing

◆ Import Module

Firstly, import the required module through import statement.

```

1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt

```

◆ Read Images

Then call **imread()** function in cv2 module to read the image that needs to be filtered.

```

6  img = cv2.imread('noise.jpg')

```

In the bracket is the image name.

◆ Mean Filtering

Call blur() function in cv2 module to perform mean filtering on the specific image.

```
9 blur1 = cv2.blur(img, (5, 5)) # mean filtering
```

The format of blur() function is as follow.

`cv2.blur(src, ksize)`

The first parameter “**scr**” is the input image.

The second parameter “**ksize**” is the size of convolution kernel.

◆ Gauss Filtering

Call GaussianBlur() function in cv2 module to perform Gauss filtering on the specific image.

```
10 blur2 = cv2.GaussianBlur(img, (5, 5), 1) # Gauss filtering
```

The format of GaussianBlur() function is as follow.

`cv2.GaussianBlur(src, ksize, sigmaX, sigmaY, borderType)`

The first parameter “**scr**” is the input image.

The second parameter “**ksize**” is the size of Gaussian convolution kernel. Both the height and width of the convolution kernel must be positive number and odd number.

The third parameter “**sigmaX**” is the horizontal standard deviation of Gaussian kernel.

The fourth parameter “**sigmaY**” is the vertical standard deviation of

Gaussian kernel, **0** by default.

The fifth parameter “**borderType**” is the type of border filling.

◆ Median Filtering

Call `medianBlur()` function in `cv2` module to perform median filtering on the specific image.

```
11 blur3 = cv2.medianBlur(img, 5) # Median filtering
```

The format of `medianBlur()` function is as follow

cv2.medianBlur(src, ksize)

The first parameter “**scr**” is the input image.

The second parameter “**ksize**” is the size of the convolution kernel.

5.2 Image Display

◆ Create Custom Image

Call `figure()` function in `matplotlib.pyplot` module to create a custom image for displaying the final output image.

```
14 plt.figure(figsize=(10, 5), dpi=100)
```

The format of the `figure()` function is as follow.

matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kwargs)

The first parameter “**num**” is the only identifier of the image i.e. the serial number of the picture (number) or the name (string).

The second parameter “**figsize**” is the width and height of the image in

inch.

The third parameter “**dpi**” is the resolution of the image i.e. the number of pixels by inch

The fourth parameter “**facecolor**” is the background color.

The fifth parameter “**edgecolor**” is the frame color

The sixth parameter “**frameon**” determines whether to draw the picture, and it is “**True**” by default.

The seventh parameter “**FigureClass**” is used to select the custom figure when generating the image

The eighth parameter “**clear**” determines whether to clear all the original images.

The ninth parameter “****kwargs**” represents other properties of the image.

◆ Modify matplotlib Configuration

matplotlib is plotting library of Python. User can access and modify matplotlib configuration options through parameter dictionary “**rcParams**”.

```
15 plt.rcParams['axes.unicode_minus'] = False
```

The codes above are used to manipulate the display of the normal characters.

◆ Set the Parameter of Image Display

Call subplot(), imshow() and title() functions in matplotlib.pyplot modules to designate the position, color and headline of the subplot in the Figure.

```
16 plt.subplot(141), plt.imshow(img), plt.title("Original")
```

1) subplot() function is used to set the position of the subplot, and the

function format is as follow.

```
matplotlib.pyplot.subplot(nrows, ncols, index, **kwargs)
```

The first parameter “**nrows**” and the second parameter “**ncols**” respectively are the number of row and column of subplot.

The third parameter “**index**” is the index position. Index starts at 1 in the upper left corner and increases to the right.

When both the row and column are less than “**10**”, these two values can be abbreviated to an integer. For example, the meaning of “subplot(1, 4, 1)” and “subplot(141)” are the same, both representing the image is divided into one row and four columns, and the subplot is in the first place i.e. 1st row, 1st column.

2) imshow() function is used to set the color of subplot, and its format is as follow.

```
matplotlib.pyplot.imshow(X, cmap=None)
```

The first parameter “**X**” is the image data.

The second parameter “**cmap**” is the colormap, RGB(A) color space by default.

3) title() function is used to set the title of the subplot. The parameter in the bracket is the name of the subplot and the function format is as follow.

```
matplotlib.pyplot.title(label, fontdict=None, loc=None, pad=None, *,  
y=None, **kwargs)
```

The first parameter “**label**” is the title composed of string.

The second parameter “**fontdict**” is the property of the font, and the current parameter refers to dictionary.

The third parameter "**loc**" is the position of the title. It can be "**left**", "**center**" or "**right**", and "**center**" by default.

The fourth parameter "**pad**" is the padding distance (inside margin) between the tile and the subplot, "**6.0**" by default.

The fifth parameter "**y**" is the vertical distance between the title and the subplot, and the unit is the percentage of the height of the subplot. The default value is "None", that is, the position of the title is automatically determined to avoid overlapping with other elements. "**1.0**" means the title is at the top of the subplot.

The sixth parameter "****kwargs**" is the text object keyword property, which is used to determine the appearance of the text, such as font, text color, etc.

◆ Set Axis Tick

Call `xticks()` and `yticks()` function in `matplotlib.pyplot` module to set the tick and tag of X and Y axis. As the coordinate axis is not required in image display in this routine, the list is set as none that is the coordinate axis will not be displayed.

```
17 plt.xticks([], plt.yticks([]))
```

The format of `xticks()` function is as follow.

```
xticks(ticks=None, labels=None, **kwargs)
```

When the parameter is none, the function will return the current tick and tag of X axis. Otherwise the function is used to set the current tick and label of X axis.

The first parameter "**ticks**" is a list of the positions of the X-axis ticks. If the list is empty, the X-axis ticks will be cleared.

The second parameter “**labels**” is the label of X-axis tick. Only when parameter “ticks” is not none, can this parameter be passed.

The third parameter “****kwargs**” is used to control the appearance of the tick and label.

The format of yticks() is the same as that of xticks(). The difference lies in the controlled object.

◆ Display Image

Call show() function in matplotlib.pyplot module to display the image on the window.

```
24 plt.show()
```

The complete codes of image display part are as follow.

```
14 plt.figure(figsize=(10, 5), dpi=100)
15 plt.rcParams['axes.unicode_minus'] = False
16 plt.subplot(141), plt.imshow(img), plt.title("Original")
17 plt.xticks(), plt.yticks()
18 plt.subplot(142), plt.imshow(blur1), plt.title("Mean Filtering")
19 plt.xticks(), plt.yticks()
20 plt.subplot(143), plt.imshow(blur2), plt.title("Gauss Filtering")
21 plt.xticks(), plt.yticks()
22 plt.subplot(144), plt.imshow(blur3), plt.title("Median Filtering")
23 plt.xticks(), plt.yticks()
24 plt.show()
```