In [1]:
```
from Image import my_image,Show
import cv2
import numpy as np
```

$\\ \textbf{Average filtering}\\.$ The main idea of the average filter is to run through the signal entry by entry, replacing each entry with the average of neighboring entries. it preserves edges while removing noise,Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection)

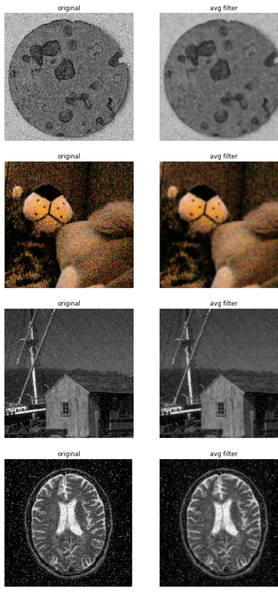
In [2]:
```
order=6
kernel = np.ones((order, order), np.float32)/order**2

img1=my_image.readimage(r'images/test1.png')
img2=my_image.readimage(r'images/test2.png')
img3=my_image.readimage(r'images/test3.png')
img4=my_image.readimage(r'images/test4.png')



avg1 = cv2.filter2D(img1, -1, kernel)
avg2 = cv2.filter2D(img2, -1, kernel)
avg3 = cv2.filter2D(img3, -1, kernel)
avg4 = cv2.filter2D(img4, -1, kernel)


Show.compareim(img1,avg1,'original','avg filter')
Show.compareim(img2,avg2,'original','avg filter')
Show.compareim(img3,avg3,'original','avg filter')
Show.compareim(img4,avg4,'original','avg filter')
```



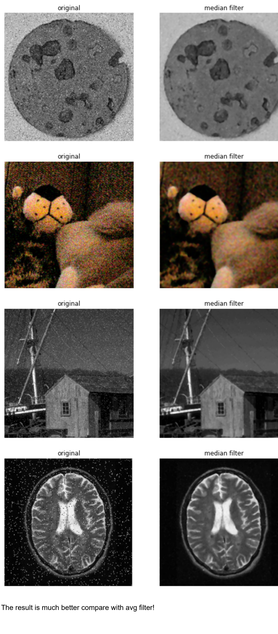
$\\ \textbf{Median filtering}\\.$ The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the average of neighboring entries. it preserves edges while removing noise,Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection)

In [3]:
```
img1=my_image.readimage(r'images/test1.png')
img2=my_image.readimage(r'images/test2.png')
img3=my_image.readimage(r'images/test3.png')
img4=my_image.readimage(r'images/test4.png')


med1 = cv2.medianBlur(img1, ksize=5)
med2 = cv2.medianBlur(img2, ksize=7)
med3 = cv2.medianBlur(img3, ksize=7)
med4 = cv2.medianBlur(img4, ksize=7)



Show.compareim(img1,med1,'original','median filter')
Show.compareim(img2,med2,'original','median filter')
Show.compareim(img3,med3,'original','median filter')
Show.compareim(img4,med4,'original','median filter')
```



The result is much better compare with avg filter!

$\\ \textbf{Gaussian filtering}\\.$

it is best to take advantage of the Gaussian blur's separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass, but requires fewer calculations.

In [4]:
```
img1=my_image.readimage('images/test6.png')
img2=my_image.readimage('images/gauss.png')
img3=my_image.readimage('images/gausstest2.png')
img4=my_image.readimage('images/gausstest.png')

img1=my_image.make_gauss_noise(img1,var=.5,mean=0)
img2=my_image.make_gauss_noise(img2,var=300,mean=0)



gsn1=my_image.gauss_filter(img1.copy(),ksize=7,sigma=3)
gsn2=my_image.gauss_filter(img2.copy(),ksize=7,sigma=600)
gsn3=my_image.gauss_filter(img3.copy(),ksize=7,sigma=3)
gsn4=my_image.gauss_filter(img4.copy(),ksize=7,sigma=1000)


Show.compareim(img1,gsn1,'original','bluring sigma=3')
Show.compareim(img2,gsn2,'original','bluring sigma=600')
Show.compareim(img3,gsn3,'original','bluring sigma=3')
Show.compareim(img4,gsn4,'original','bluring sigma=1000')
```

هر چه سیگما بزرگتر شود میزان بلورشدگی و ناهمی نویزی که به صورت گوسی بوده کم میشود.