

```
In [19]: from Image import my_image, Show, segmentation
import numpy as np
import cv2
from skimage import color
import matplotlib.pyplot as plt
```

Edge detection:

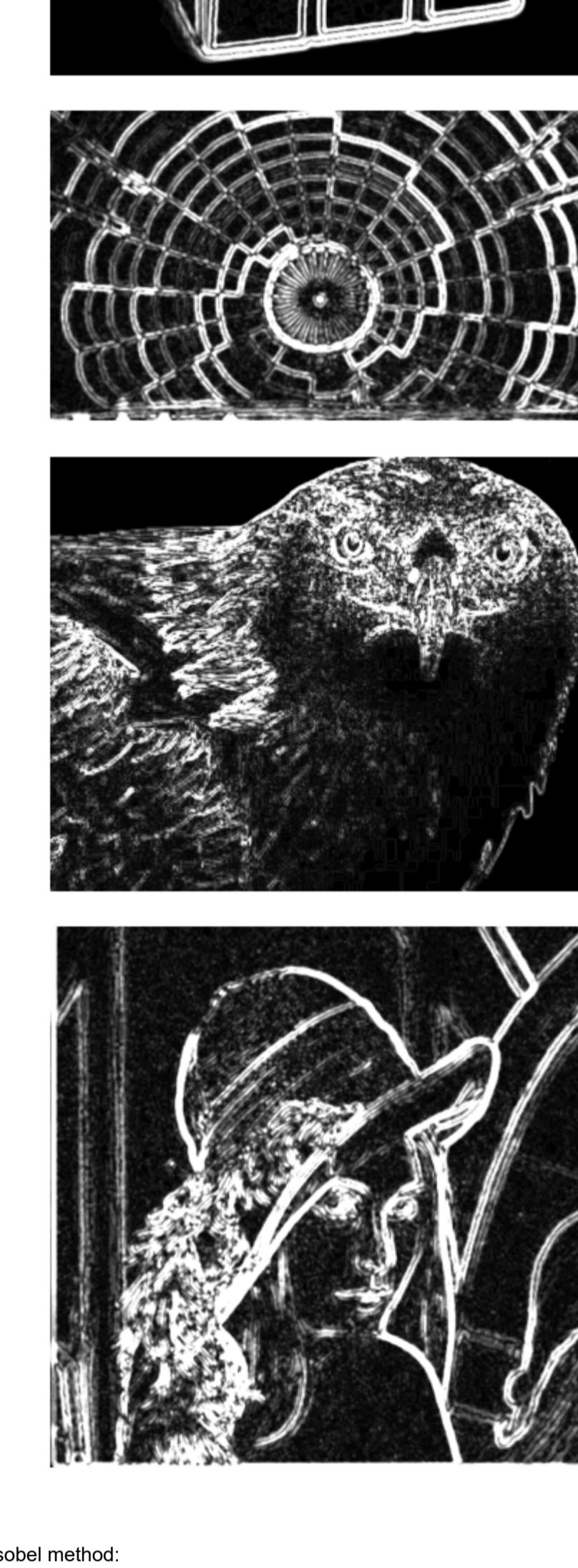
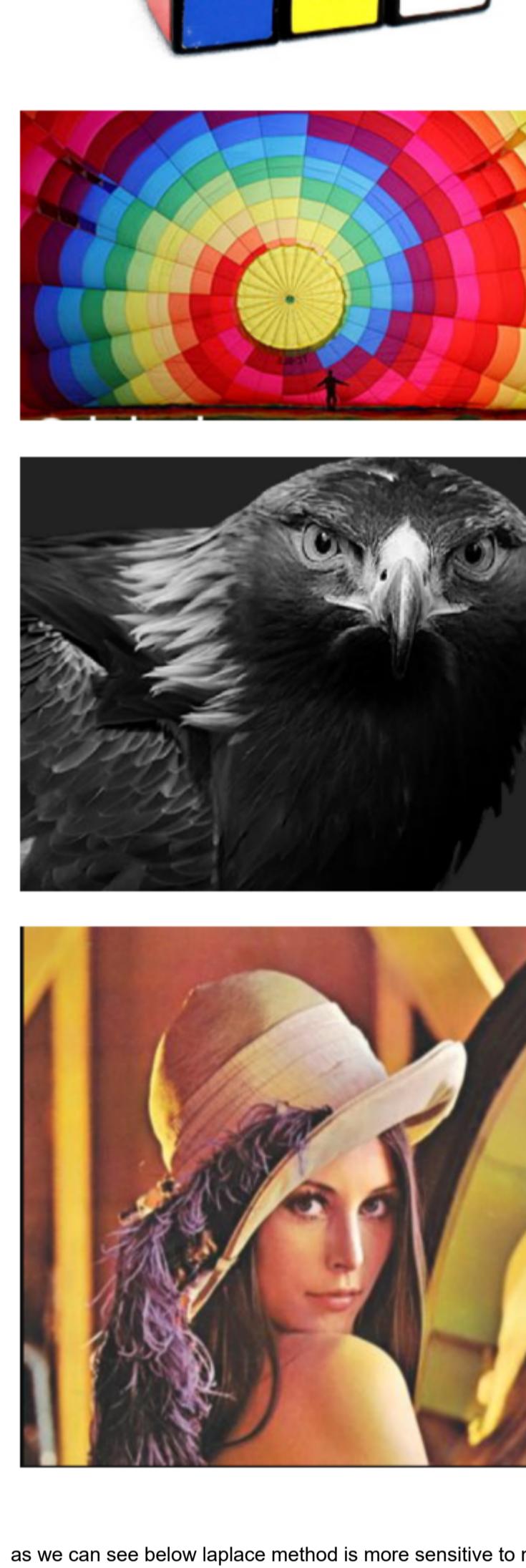
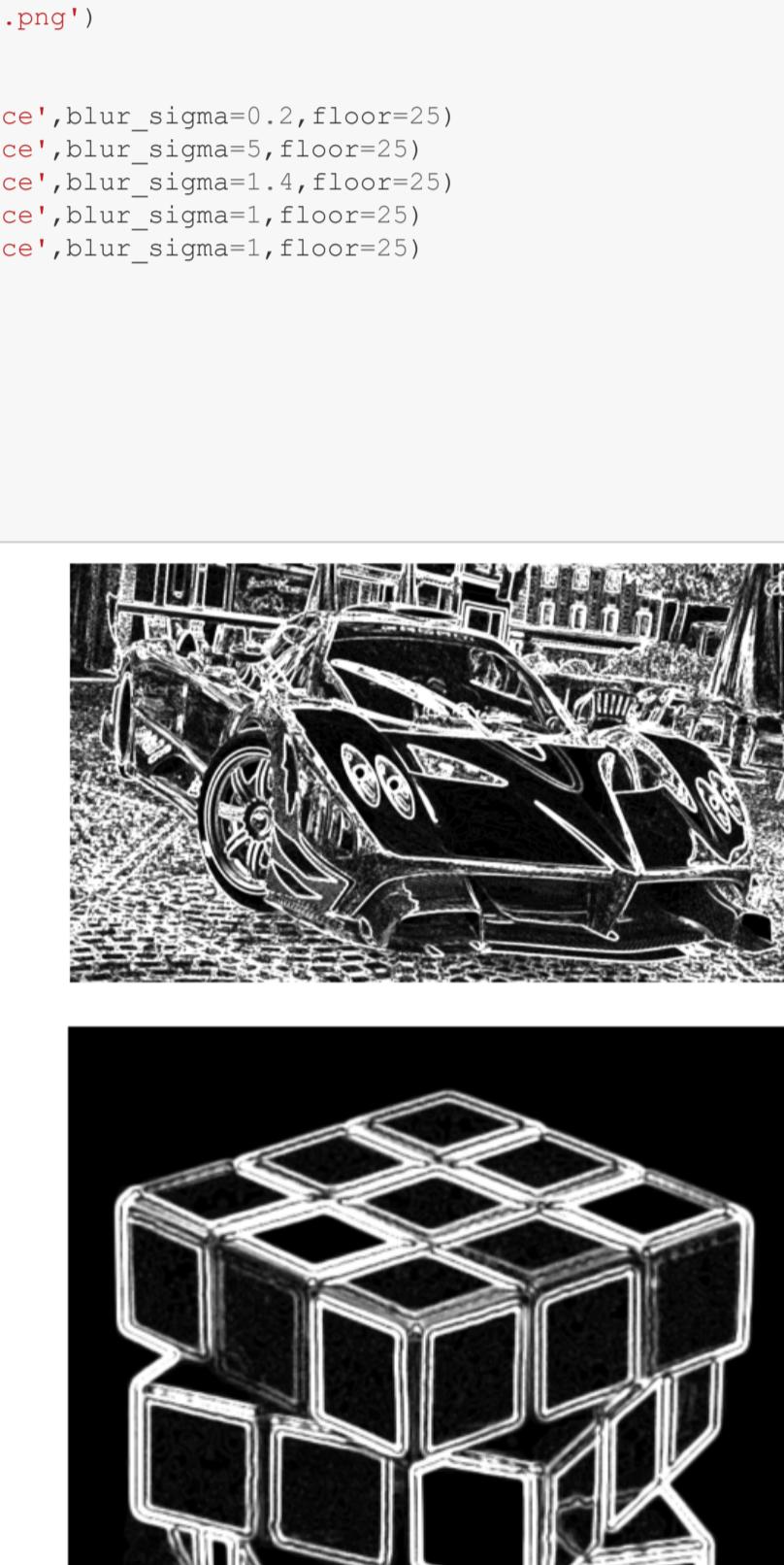
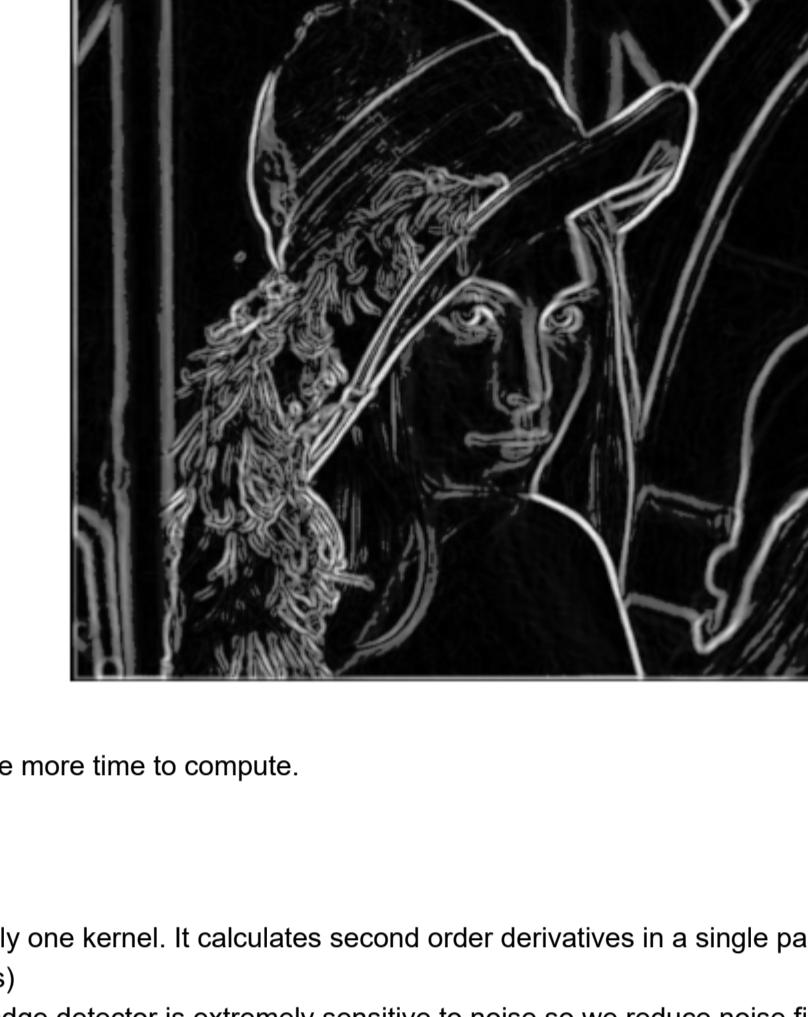
a) Sobel :

- Detects edges are where the gradient magnitude is high. This makes the Sobel edge detector more sensitive to diagonal edge than horizontal and vertical edges.
- The operator uses two kernels which are convolved with the original image to calculate approximations of the derivatives (one for horizontal changes, and one for vertical). If we define A as the source image, and Gx and Gy are two images which at each point contain the vertical and horizontal derivative approximations.
- At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude.
- above process is in Image.py

```
In [20]: img1=my_image.readimage('images/PNG2.jpg')
img2=my_image.readimage('images/edgedetection/test1.png')
img3=my_image.readimage('images/test5.png')
img4=my_image.readimage('images/edgedetection/test2.png')
img5=my_image.readimage('images/edgedetection/test3.png')
```

```
img1 ,edge1 =my_image.edge_detection(img1,mod='sobel',blur_sigma=0.2)
img2 ,edge2 =my_image.edge_detection(img2,mod='sobel',blur_sigma=0.2)
img3 ,edge3 =my_image.edge_detection(img3,mod='sobel',blur_sigma=0.2)
img4 ,edge4 =my_image.edge_detection(img4,mod='sobel',blur_sigma=0.2)
img5 ,edge5 =my_image.edge_detection(img5,mod='sobel',blur_sigma=0.2)

Show.compareim(img1,edge1,'','','size=2)
Show.compareim(img2,edge2,'','','size=1.5')
Show.compareim(img3,edge3,'','','size=1.5')
Show.compareim(img4,edge4,'','','size=1.5')
Show.compareim(img5,edge5,'','','size=1.5')
```



* after edge detection with sobel; i use thresholding. So it takes a little more time to compute.

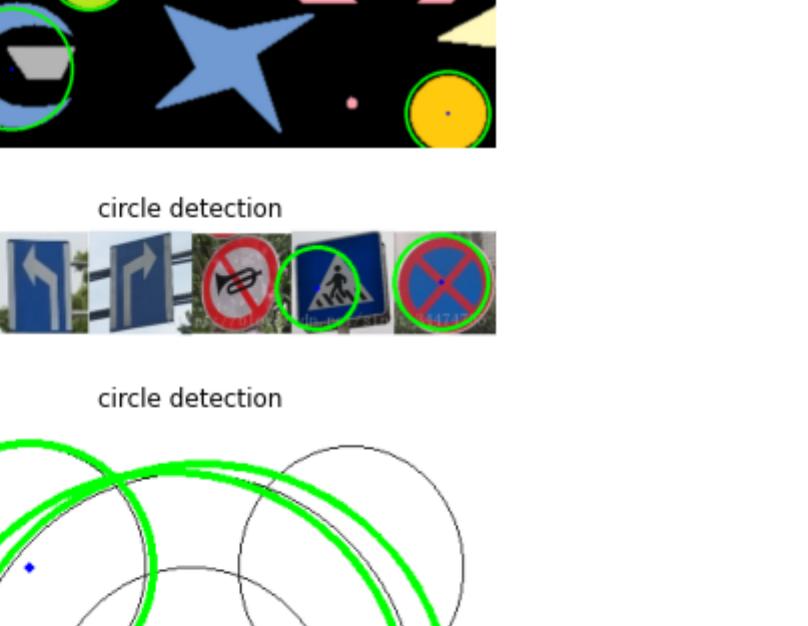
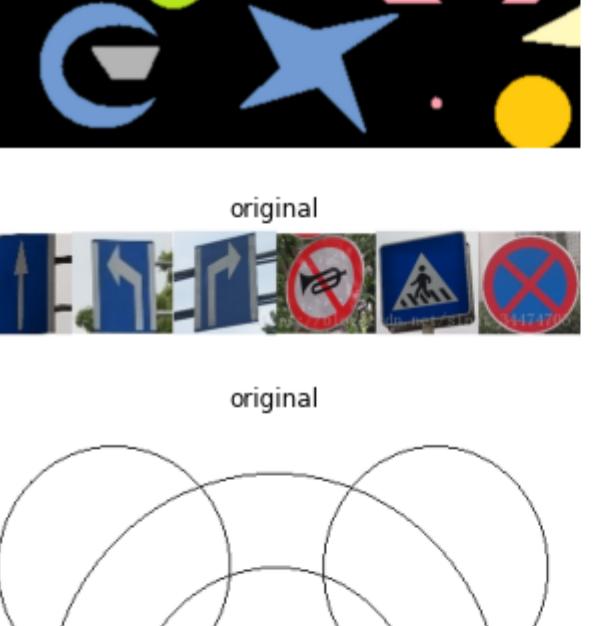
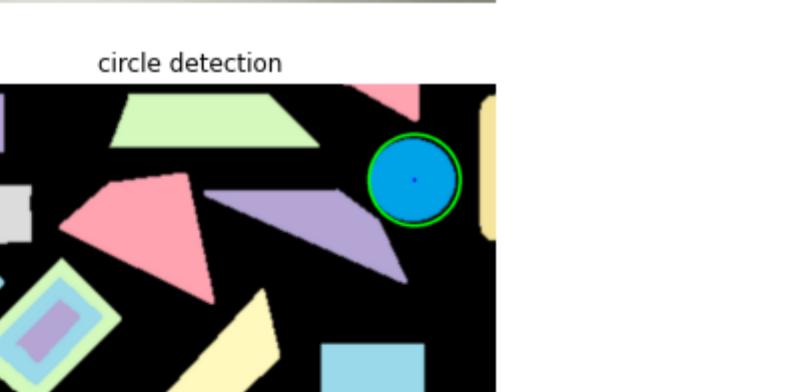
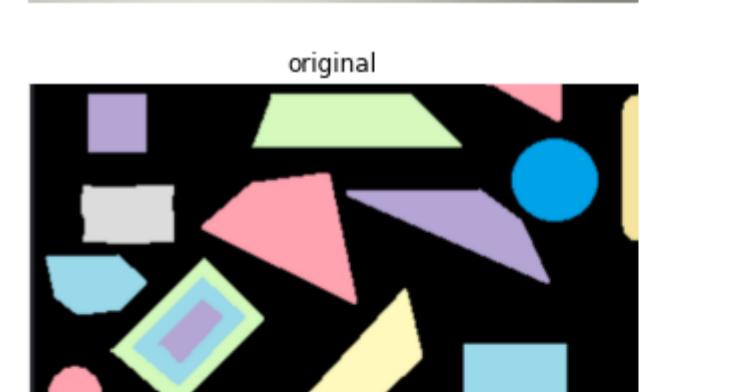
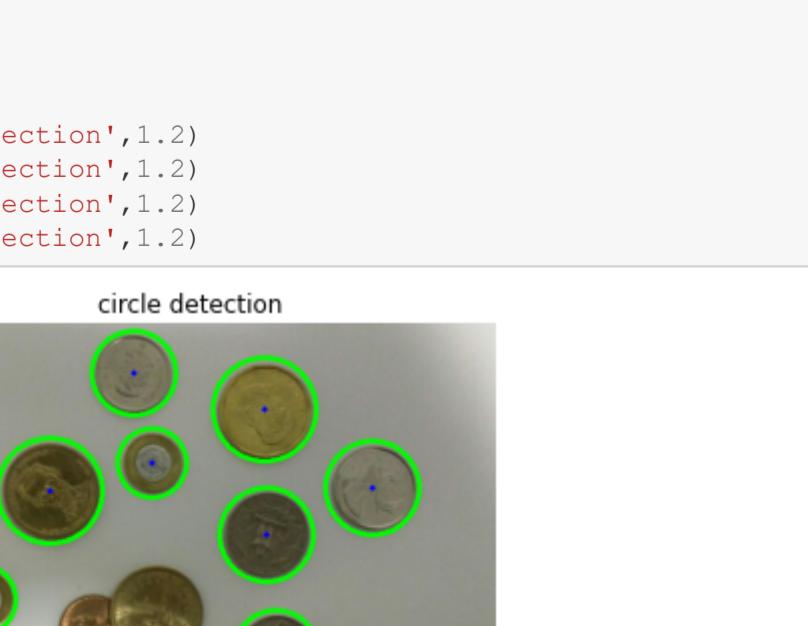
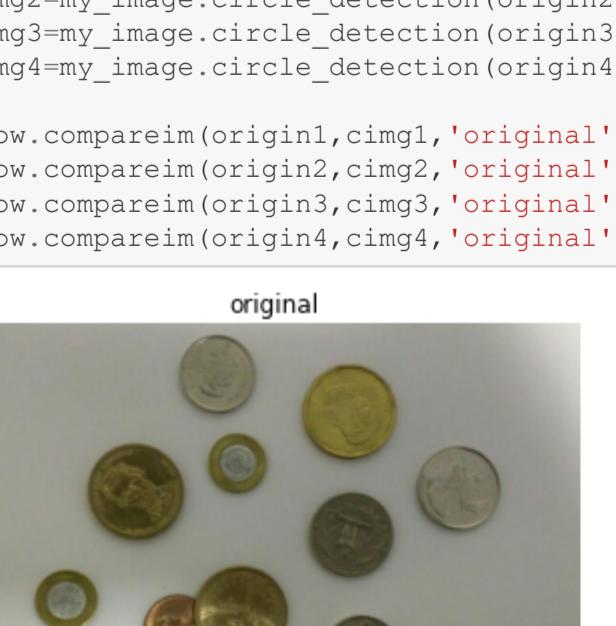
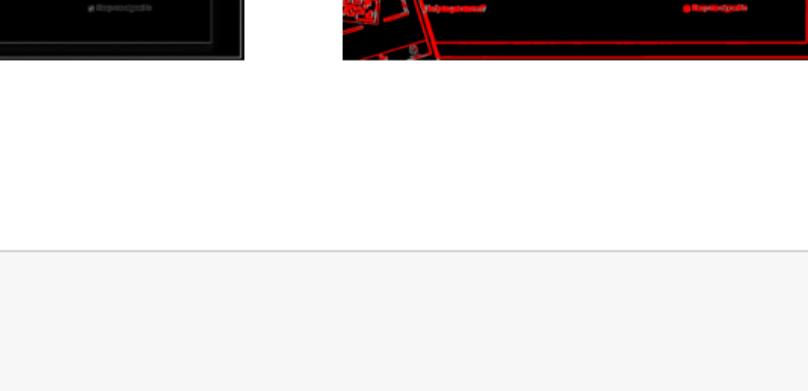
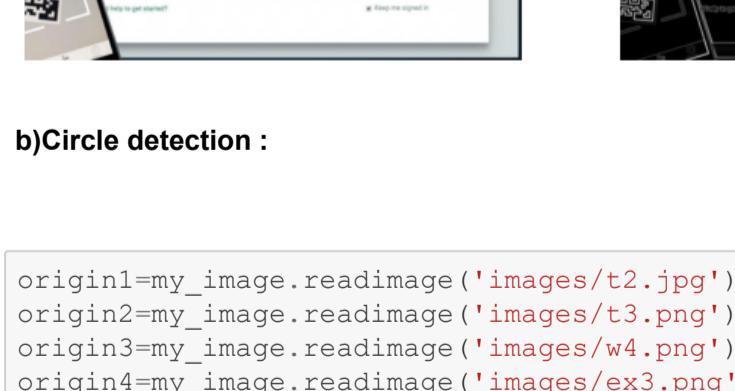
b) Laplace :

- Unlike the Sobel edge detector, the Laplacian edge detector uses only one kernel. It calculates second order derivatives in a single pass so it is computationally faster to calculate (only one kernel vs two kernels)
- because we're working with second order derivatives, the laplacian edge detector is extremely sensitive to noise.so we reduce noise first by median filter and second with Gauss filter

```
In [21]: img1=my_image.readimage('images/PNG2.jpg')
img2=my_image.readimage('images/edgedetection/test1.png')
img3=my_image.readimage('images/test5.png')
img4=my_image.readimage('images/edgedetection/test2.png')
img5=my_image.readimage('images/edgedetection/test3.png')
```

```
img1 ,Edge1 =my_image.edge_detection(img1,mod='laplace',blur_sigma=0.2,floor=25)
img2 ,Edge2 =my_image.edge_detection(img2,mod='laplace',blur_sigma=5,floor=25)
img3 ,Edge3 =my_image.edge_detection(img3,mod='laplace',blur_sigma=1.4,floor=25)
img4 ,Edge4 =my_image.edge_detection(img4,mod='laplace',blur_sigma=1,floor=25)
img5 ,Edge5 =my_image.edge_detection(img5,mod='laplace',blur_sigma=1,floor=25)

Show.compareim(img1,Edge1,'','','size=2)
Show.compareim(img2,Edge2,'','','size=1.5')
Show.compareim(img3,Edge3,'','','size=1.5')
Show.compareim(img4,Edge4,'','','size=1.5')
Show.compareim(img5,Edge5,'','','size=1.5')
```



as we can see below laplace method is more sensitive to noise than sobel method:

```
In [4]: Show.compareim(Edge5,Edge1,'Sobel method','Laplace method (LOG)',2)
```



Sobel method



Laplace method (LOG)

2.Hough transforms:

a)Line detection :

- first i do edge detection after importing the image
- second i threshold it
- third i do line detection with HoughLinesP

```
In [23]: img1=my_image.readimage('images/edgedetection/test1.png')
img2=my_image.readimage('images/test5.png')
img3=my_image.readimage('images/test8.jpg')
img4=my_image.readimage('images/test9.png')
```

```
img1 ,edge1 =my_image.line_detection(img1)
img2 ,edge2 =my_image.line_detection(img2)
img3 ,edge3 =my_image.line_detection(img3)
img4 ,edge4 =my_image.line_detection(img4)

Show.compareim(img1,edge1,'original','edge',1,2,1,line_detect1,'line detection')
Show.compareim(img2,edge2,'original','edge',2,1,line_detect2,'line detection')
Show.compareim(img3,edge3,'original','edge',2,1,line_detect3,'line detection')
Show.compareim(img4,edge4,'original','edge',2,1,line_detect4,'line detection')
```

original

edge

line detection

original

edge

line detection

original

edge

line detection

original

edge

line detection

original

edge

line detection

original

```
In [22]: img1=my_image.readimage('images/mm.jpg')
line=my_image.line_detection(img1)
```

```
Show.show_me(line[2])
```

