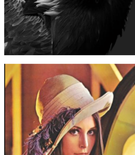
In [1]:
```
from image import my_image,Show
import numpy as np
import cv2
from mlimage import color
import matplotlib.pyplot as plt
```

Edge detection:

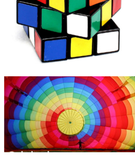
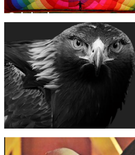
a) Sobel :
-Detects edges are where the gradient magnitude is high.This makes the Sobel edge detector more sensitive to diagonal edge than horizontal and vertical edges.
-The operator uses two kernels which are convolved with the original image to calculate approximations of the derivatives (one for 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 [2]:
```
img1=my_image.readimage('images/POR1.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.comparein(img1,edge1,'','',size=3)
Show.comparein(img2,edge2,'','',size=1.5)
Show.comparein(img3,edge3,'','',size=1.5)
Show.comparein(img4,edge4,'','',size=1.5)
Show.comparein(img5,edge5,'','',size=1.5)
```



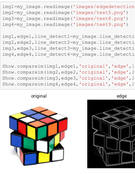
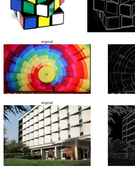



- after edge detection with sobel, i use thresholding. So it takes a little more time to compute.

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 [3]:
```
img1=my_image.readimage('images/POR1.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=1,floor=25)
img3_,Edge3=my_image.edge_detection(img3,mod='laplace',blur_sigma=1.5,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.comparein(img1,Edge1,'','',size=3)
Show.comparein(img2,Edge2,'','',size=1.5)
Show.comparein(img3,Edge3,'','',size=1.5)
Show.comparein(img4,Edge4,'','',size=1.5)
Show.comparein(img5,Edge5,'','',size=1.5)
```


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

In [4]:
```
Show.comparein(edge5,Edge5,'Sobel method','Laplace method (LOG)',2)
```

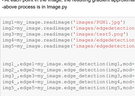
Sobel method

Laplace method (LOG)

2.Hough transform:

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

In [5]:
```
img1=my_image.readimage('images/edgedetection/test1.png')
img2=my_image.readimage('images/test5.png')
img3=my_image.readimage('images/test2.png')

img1,edge1,line_detect1=my_image.line_detection(img1)
img2,edge2,line_detect2=my_image.line_detection(img2)
img3,edge3,line_detect3=my_image.line_detection(img3)
img4,edge4,line_detect4=my_image.line_detection(img4)

Show.comparein(img1,edge1,'original','edge',1.2,line_detect1,'line detection')
Show.comparein(img2,edge2,'original','edge',1.2,line_detect2,'line detection')
Show.comparein(img3,edge3,'original','edge',1.2,line_detect3,'line detection')
Show.comparein(img4,edge4,'original','edge',1.2,line_detect4,'line detection')
```

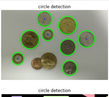


b)Circle detection :

In [6]:
```
origin1=my_image.readimage('images/t2.jpg')
origin2=my_image.readimage('images/t3.png')
origin3=my_image.readimage('images/e4.png')
origin4=my_image.readimage('images/em3.png')

cimg1=my_image.circle_detection(origin1)
cimg2=my_image.circle_detection(origin2,5,7,14)
cimg3=my_image.circle_detection(origin3,5,7,14)
cimg4=my_image.circle_detection(origin4,1,7,13)

Show.comparein(origin1,cimg1,'original','circle detection',1.2)
Show.comparein(origin2,cimg2,'original','circle detection',1.2)
Show.comparein(origin3,cimg3,'original','circle detection',1.2)
Show.comparein(origin4,cimg4,'original','circle detection',1.2)
```

original

circle detection



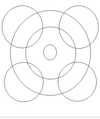
original

circle detection

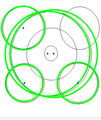
original

circle detection

original



circle detection



In [ ]: