3.6.ImageGradient & Edge Detect

6.1derivatives or gradient

- 高通滤波器，检测边缘
- cv2.Scharr()/cv2.Sobel(): Scharr(3*3 matrix) is a optimized version of Sobel, for calculating 1st or 2nd derivatives
- cv2.Laplacian(): for 2nd derivatives, and: $kernel = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('data/sudoku.jpg', 0)

# cv2.CV_64F输出图像的深度（若使用 -1，与原图像保持一致，i.e.:np.uint8）
#若使用-1（而非cv2.cv_64F）,会丢失导数为负数的边界（从白到黑）
laplacian = cv2.Laplacian(img, cv2.CV_64F)
# 参数1，0表示x方向（最大可以求2nd导数）
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)
# 参数0,1表示y方向（最大可以求2nd导数）
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)

plt.subplot(2, 2, 1), plt.imshow(img, cmap='gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 2), plt.imshow(laplacian, cmap='gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 3), plt.imshow(sobelx, cmap='gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 2, 4), plt.imshow(sobely, cmap='gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
plt.show()
```

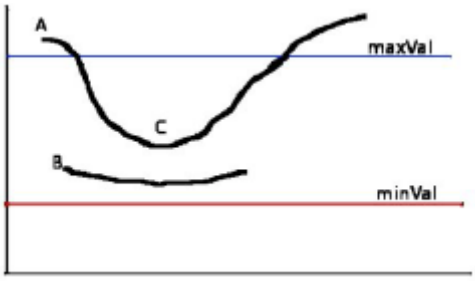
ps：depth of image--bits

6.2 Edge detection--Canny

6.2.1 原理

1. Noise reduction: using 5 by 5 Gaussian filter
2. With Sobel(), get Gx andGy--the 1st derivatives on x and y axes. Using them to caculate the graident and angle of detected edge, and $Angle(\theta) = \tan^{-1}(\frac{Gx}{Gy})$, $(G) = \sqrt{(Gx^2 + Gy^2)}$
3. Scan the whole image, and ingore the points that are not on edges, by the following idea: if one pixel is on the edge, **the gradient of this point should be the largest** among its neighboors **with the same angle.**
4. Find the 'true' edge: derivatives>maxVal→ture

- derivatives>maxVal→ture
- derivatives<minVal→false
- derivatives∈(minVal,maxVal): depend on weather it is connected w/ true edge



6.2.2 cv.canny()

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('data/messi5.jpg',-1)

edges = cv2.Canny(img, 230, 240)
#Canny(image, minVal, maxVal,kenerl=3_by_3,LaGradient=False)
#L2Gradient=False: Edge-Gradient(G)=|Gx^2|+|Gy^2|
#L2Gradient=True: Edge-Gradient(G)=(Gx^2+G2y^2)
cv2.imshow('Edges',edges)
cv2.waitKey(0)
cv2.destroyAllWindows
```

ps:

L2Gradient=False$\rightarrow$ Edge−Gradient(G)=$|Gx^2| + |Gy^2|$

L2Gradient=True$\rightarrow$ Edge−Gradient(G)=$\sqrt{(Gx^2 + Gy^2)}$