Image Gradients (Edge visually detector)

① Sober and Scharr Derivatives

Sober operations ~~are~~ is a joint Gaussian smoothing plus differentiation operation, so its more resistant to noise.

One of the arguments is the ksize. If ksize is (-1), then a 3x3 Scharr filter is used, which gives better results than a 3x3 Sober filter.

② Laplacian Derivatives

If ksize=1, then this kernel is used: kernel = $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

#lets see now an example of how these functions help us see the edges:

```
img = cv2.imread(...,...)
laplacian = cv2.Laplacian(img, cv2.CV_8U)
```

... 16S
... 16U
... 8S
... 32S
... 32F
... 64F

MAT TYPE.

```
sobelx = cv2.Sobel(img, cv2.MAT_TYPE, x, y, ksize=3)
sobely = cv2.Sobel(img, cv2.MAT_TYPE, x, y, ksize=3)
```

↳ since we want to see the "x" corners, cv2.sobel(img, cv2.MAT_TYPE, 1, 0, ksize=3)

↳ for the y corners: cv2.sobel(img, cv2.MAT_TYPE, x, y, ksize=3)

the bigger the ksize, the smaller the accuracy..

Best result:

```
sobel = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5) #High value format as
                                                  CV_16S, CV_64F
sobel_abs = np.absolute(sobel) # TAke the absolute of the sobel
sobel_8U = np.uint8(sobel_abs) # create its 8 CV_8U format.
```