

Image Gradients (Edge visually detector).

① Sobel and Scharr Derivatives

Sobel 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 Sobel filter.

② Laplacian Derivatives

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

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

`img = cv2.imread(.....)`
`Laplacian = cv2.Laplacian(img, cv2.CV_8U)` → MAT TYPE.

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

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

△ since we want to see the "x" corners: `cv2.sobel(img, cv2.MATTYPE, 1, 0, ksize=3)`

△ for the y corners: `cv2.sobel(img, cv2.MATTYPE, x, y, ksize=3)`

The bigger the ksize, the smaller the accuracy..

Best result:

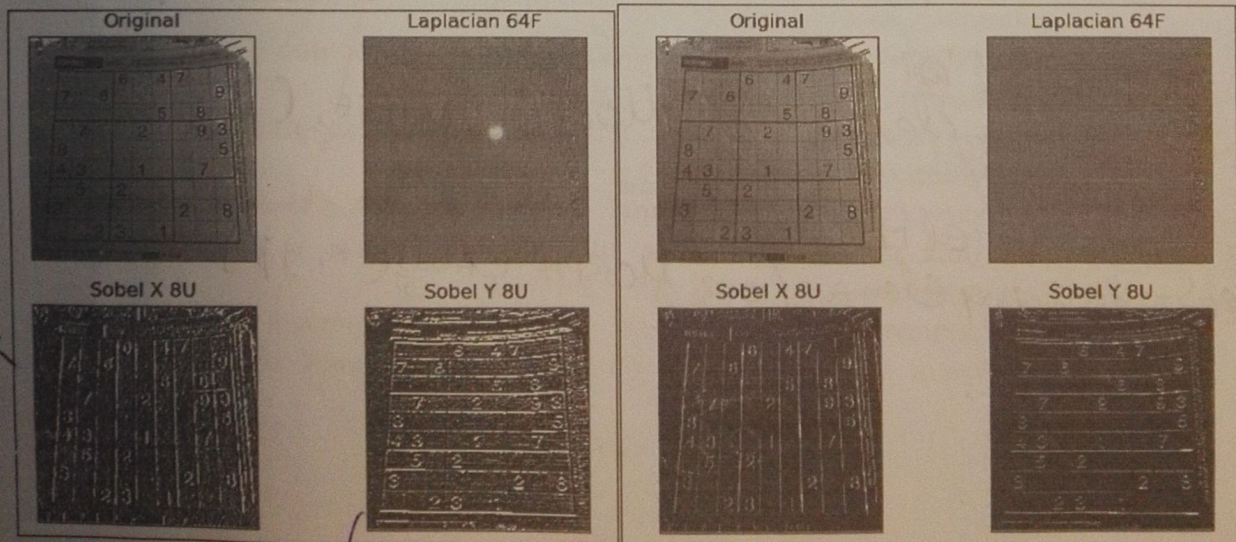
`Sobel = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)` # High value format as CV_16S, CV_64F

`Sobel-abs = np.absolute(sobel)` # Take the absolute of the sobel

`Sobel-8U = np.uint8(sobel-abs)` # create its CV_8U format.

ksize= 5

ksize= 3



x=1
y=0

x=0
y=1

23