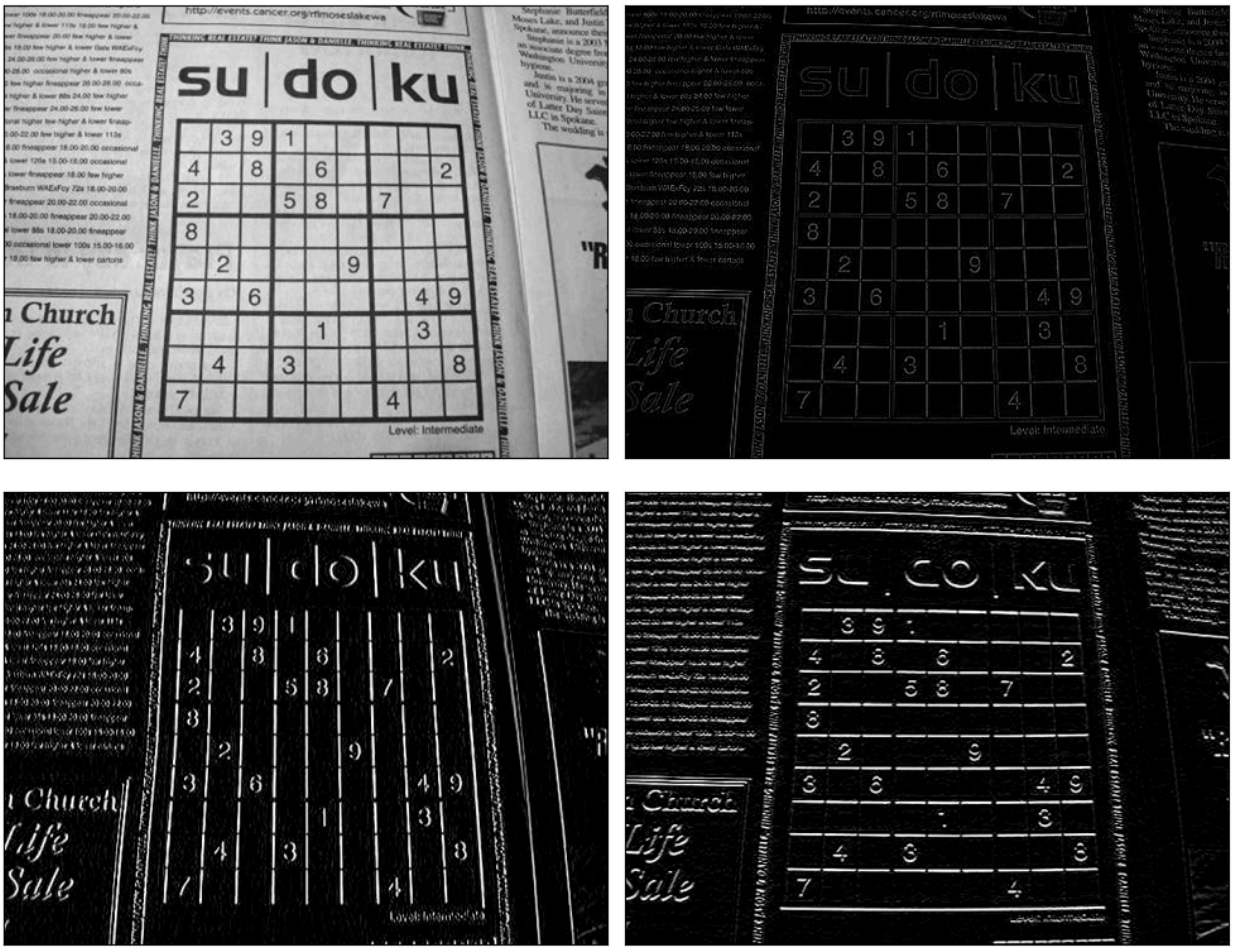
Lecture 7 (High-pass Filters & Edge Detection)



Topics:

1. Edge Detection
2. Sobel filter
3. Laplacian filter
4. Canny edge detection

What is Edge Detection in Image Processing?

Edge: The edges can be defined as curves that join different regions of an image, and they are crucial for understanding the structure and content of the image.

Edge detection is a technique used in image processing to identify points in a digital image where the brightness changes sharply or has discontinuities.
These points are typically associated with significant changes in the image, such as the boundaries of objects, textures, and other features.

High-pass Filters

- emphasize edges and fine details
- suppress the low-frequency components of an image
- preserve or enhance high-frequency components like edges, texture, or noise.

Purpose: Edge detection, enhancing fine details, sharpening.

Common High-pass Filters:

1. **Sobel Filter:**
   - Detects edges by calculating the gradient of image intensity in both horizontal and vertical directions.
   - uses two 3x3 kernels (one for detecting horizontal changes, another for vertical changes)
   - Sobel operators is a joint Gaussian smoothing plus differentiation operation
   - more resistant to noise
   - The term "Sobel" is derived from the names of its inventors, Gilberto Sobel and Gary Feldman.

Horizontal Sobel kernel:

```
[[-1, 0, 1],
[-2, 0, 2],
[-1, 0, 1]]
```

Vertical Sobel kernel:

```
[[-1, -2, -1],
[ 0,  0,  0],
[ 1,  2,  1]]
```

cv.Sobel(src_img, ddepth, dx, dy, ksize=kernel_size)

ddepth: data type of the output image, when ddepth=-1, the output image will have the same depth as the source.
dx: order of the derivative in the x-direction.
dy: order of the derivative in the y-direction.

2. **Laplacian Filter:**
   - A second-order derivative filter that detects edges based on changes in intensity.
   - sensitive to noise
   - provides detailed edge information

$$\texttt{dst} = \Delta \texttt{src} = \frac{\partial^2 \texttt{src}}{\partial x^2} + \frac{\partial^2 \texttt{src}}{\partial y^2}$$

cv.Laplacian(src_img, ddepth, ksize)

If ksize = 1, it is 3x3 kernel

[[0, 1, 0],
[1, -4, 1],
[0, 1, 0]]

If ksize>1 then the function calculates the Laplacian of the source image by adding up the second x and y derivatives calculated using the above equation.

Canny edge detection

- popular edge detection algorithm. It was developed by John F. Canny
- The algorithm is a multi-stage algorithm
- It can handle noises
- Very precise
- The quality of the output depends on the choice of threshold values
- Computationally intensive

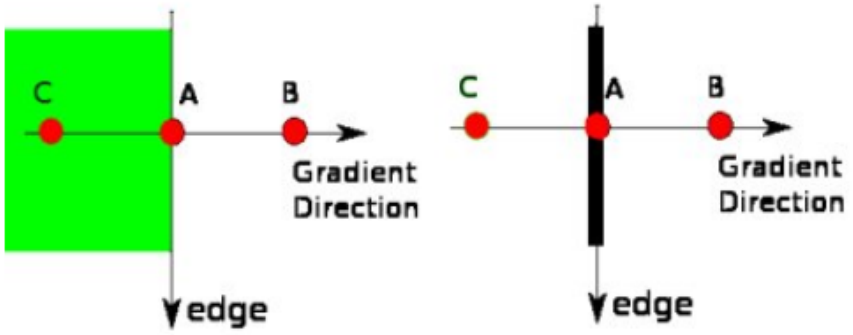
Stages of the algorithm:

1. **Noise Reduction:** uses a 5x5 gaussian filter to remove noise

2. **Finding Intensity Gradient of the Image:**
   Smoothened image is then filtered with a Sobel kernel to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2} Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3. **Non-maximum Suppression:** a full scan of image is done to remove any unwanted pixels which may not constitute the edge.

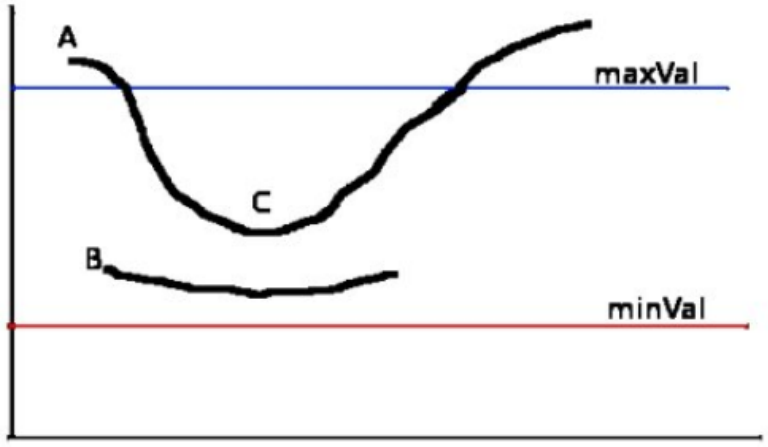


Image

Opencv

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).
In short, the result you get is a binary image with "thin edges".

4. **Hysteresis Thresholding:**
   Finds actual edges using two threshold values, minVal and maxVal.



Opencv

Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.
This stage also removes small pixels noises on the assumption that edges are long lines.

cv.Canny(src_img, lower_thresh, upper_thresh)

Don’t stop here
Learn More: **Prewitt Operator**, **Scharr Operator …**