2A-L6 Edge detection - 2D

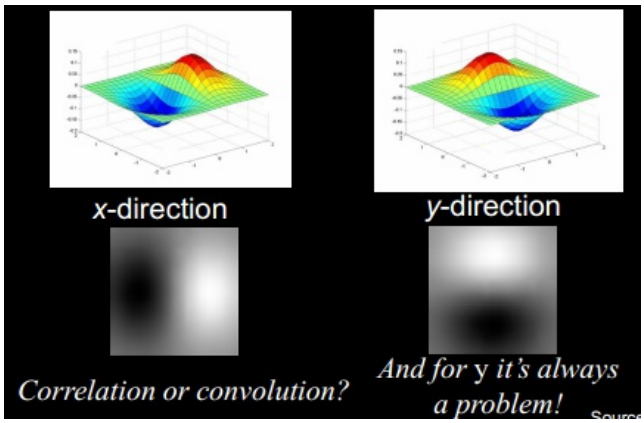
2017/11/11 13:41

1. SUM
    a. the procedure to find the edge
        i. smooth the filter
            1. sigma effects the final egde, the bigger, the more significant
        ii. gradient the filter
            1. Laplacian for 2D
        iii. apply the filter to the image to get the mag and dir of gradient
        iv. threshold
        v. thin
        vi. connect
    b. filters
        i. canny
            1. key ideas
                a. thinning
                b. connecting
            2. edge(img, 'canny')
            3. works better than sobel and others
    c. edge detector matlab
        i. edge(img, 'type', threshold, direction, thinning)
2. Inro
    a. as mentioned before, we can compute the derivative of the filter first and then apply it to the images
    b. reasons
        i. Since h is typically smaller we take fewer derivatives so it's faster.
        ii. The smoothed derivative operator is computed once and you have it to use repeatedly.

c. why can we do this

$$(I \otimes g) \otimes h_x = I \otimes (g \otimes h_x)$$

3. How to choose the size of Gaussian Filter 2D



a. it's the correlation, ATT to the positive direction

b. the bigger size of Gaussian filter, the stronger the smoother and thus the sharper edage reamins and the shallow edges are just smooth out.

i. Smaller values: finer features detected

ii. Larger values: larger scale edges detected

1. How to find the edges

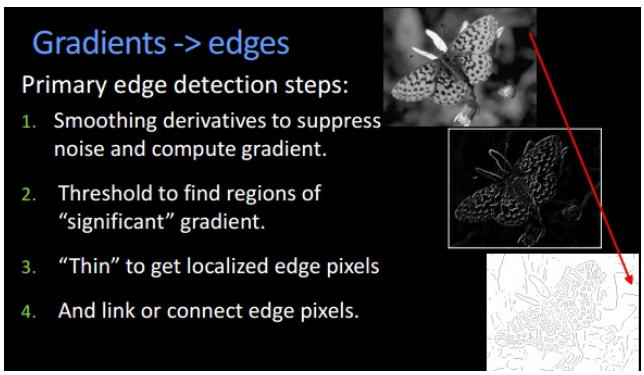
a. Primary edge detection steps:

i. smoothing the derivatives of filters to suppress noise and compute gradient

ii. threshold to find regions of "significant" gradients

iii. "thin" to get localized edge pixels

iv. link or connect edge pixels



b. MATLAB: edge(image, 'canny')

1. https://de.mathworks.com/help/images/ref/edge.html?searchHighlight=edge&s_tid=doc_srchtitle
2. `BW = edge(I)`
    a. returns a binary image `BW` containing **`1`s** where the function finds edges and **`0`s elsewhere**.
    b. The input image `I` is an **intensity or a binary image**.
    c. `BW` is the same size as `I`.
3. `BW = edge(I,'canny')`
    a. By default, `edge` uses the Sobel edge detection method, but we can specify Canny (or a Canny approximation), Laplacian of Gaussian (log), Prewitt, Roberts, or Zero-crossings.
    b. canny is better than sobel
4. `BW = edge(I,'canny',threshold)`
    a. return all edges that are stronger than `threshold`. if not specified, `edge` chooses the value automatically.
5. `BW = edge(I,'canny',threshold,direction)`
    a. specify the direction in which the function looks for edges in the image: `'horizontal'`, `'vertical'`, or `'both'`.
6. `BW = edge(I,'canny',threshold,direction,'nothinning')`

a. 'nothinning' specify to skip the additional edge-thinning stage, The default value is 'thinning'.

7. [BW,threshOut] = edge(I,'canny',__)

a. eturns the threshold value.

1. Canny Edge Detector

a. **filter derivative:** Filter image with derivative of Gaussian

b. **gradient:** Find magnitude and orientation of gradient

c. **thresholding:** to keep the 'significant' edges

d. **Non-maximum suppression: -- thinning**

i. Thin multi-pixel wide "ridges" down to single pixel width

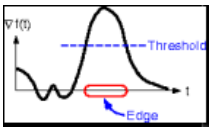
ii. Check and only keep the pixel with local maximum along gradient direction

e. Linking and thresholding (hysteresis):

i. Core idea: Define two thresholds: low and high. Use the high threshold to start edge curves and the low threshold to continue them

ii. procedure

1. Apply a high threshold to detect strong edge pixels.



a. problem: some pixel along the edge can't survive the thresholding

1. Link those strong edge pixels to form strong edges.

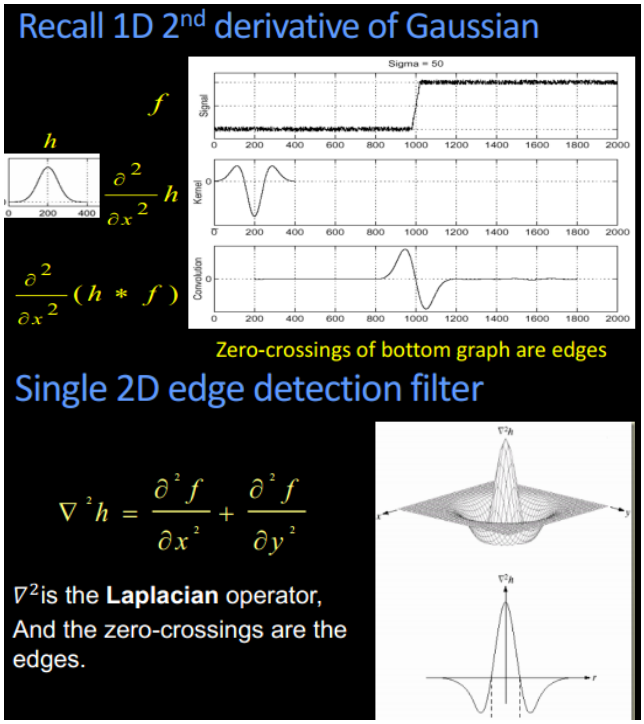
2. Apply a low threshold to find weak but plausible edge pixels.

3. Extend the strong edges to follow weak edge pixels

a. the assumption here: the important edges have some strong

gradient pixels. thus we only keep the edges with strong-edge pixels

1. 10. Canny Results
   a. with canny, we get better results than with others
   b. but actually, it's hard to know when an edge image is good since it has the meaning when it is put into the specific application domain
   c. size of sigma influence the granurity of the edges
2. 11. Quiz: Canny Edge Quiz
   a. The Canny edge operator is probably quite sensitive to noise
      i. Mostly false - it depends upon the σ chose
3. 12. Single 2D Edge Detection Filter
   a. we use the Laplacian operator for the 2nd derivative for 2D



9. Side
   a. image types
      i. A binary image is a digital image that has only two possible values for each pixel.
      ii. An **intensity image** is a data matrix, I, whose values represent intensities within some range, e.g. [0, 255]. with each element of the matrix corresponding to one image pixel
         1. grayscaled image
      iii. **Indexed Images**

1. An indexed image consists of a data matrix, `X`, and a colormap matrix, `map`. `map` is an *m*-by-3 array of class `double` containing floating-point values in the range [0, 1].
2. Each row of `map` specifies the red, green, and blue components of a single color

iv. **RGB (Truecolor) Images**

1. stored as an *m*-by-*n*-by-3 data array that defines red, green, and blue color components for each individual pixel.
2. Graphics file formats store RGB images as 24-bit images, where the red, green, and blue components are 8 bits each.
3. can be of class `double`, `uint8`, or `uint16`.