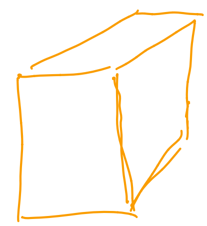
CSci 4270 and 6270
Spring 2021
Lecture 7: Feature Extraction — Edges
Thursday, February 19, 2021

Overview

- Goal: extract image locations of sharpest intensity change.
  - Intuitively, edges are “significant events” in images.
  - This is a “boundary first” approach to image analysis, whereas the problem area known as *segmentation* is a “region first” approach.



- A single pixel edge is called an *edgel*, short for “edge element”.
- Steps of edge(l) detection
  - Smoothing
  - Differentiation, gradient magnitude and direction
    - Some methods use second order derivatives, but we will not.
  - Non-maximum suppression, sub-pixel localization, thresholding.
  - Linking into edgel chains
- We will discuss each in turn.

Smoothing for Edge Detection

Choice is the Gaussian, for reasons we have already outlined

- Isotropic and separable
- Fast implementations
- Good spatial and frequency properties

Choosing the Smoothing Scale: Scale Space

- Smaller values of σ lead to more detailed edges and better edge localization.
- Larger values of σ lead to fewer and perhaps more significant edges
- Different scales are needed when images have different sizes or magnifications
- We can combine these by working with multiple scales, treating σ as an additional variable, producing what is called *scale space*.
  - We will look at scale space in much more detail in future lectures.
- For the purposes of these notes, we will work with the results of smoothing with a single value of σ.

Differential Operations

- Two common choices
  - 1st derivative based on the directional derivatives, the gradient, and finding peak.
  - 2nd derivative based on the Laplacian and finding "zero-crossings".
- While our emphasis will be on first derivative operations, during lecture we will take a brief look at the formal structure of the Laplacian and why it is used. In particular, we will discuss the equation

$$\nabla^2(I * G)$$

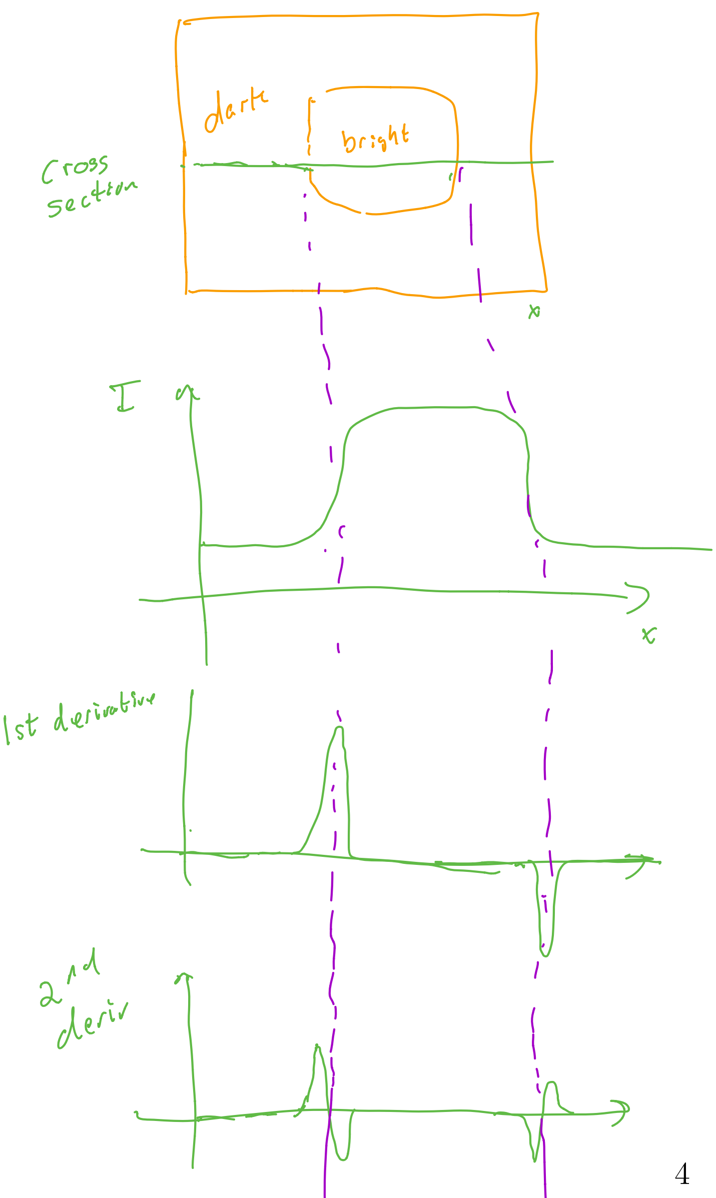
img gaussian smooth

where

$$\nabla^2(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

compute 2nd derivative and add result

We will return to the use of the Laplacian during "interest point" extraction.



Computing the Gradient

- Let I_s be the smoothed image.
- Compute $I_x = \partial I_s / \partial x$ and $I_y = \partial I_s / \partial y$ using discrete differentiation.
- Then we can compute the gradient direction and magnitude as

$$I_m(x,y) = \| I_x(x,y)^2 + I_y(x,y)^2 \|^{1/2}$$
$$I_\theta(x,y) = \text{atan2}(I_y(x,y), I_x(x,y))$$

  – Notice with atan2 the y derivative is first.

- The direction is “normal” to the direction of the edge contour and points in the direction of most rapid intensity increase.

range $[-\pi, \pi]$

$I_y = -1$

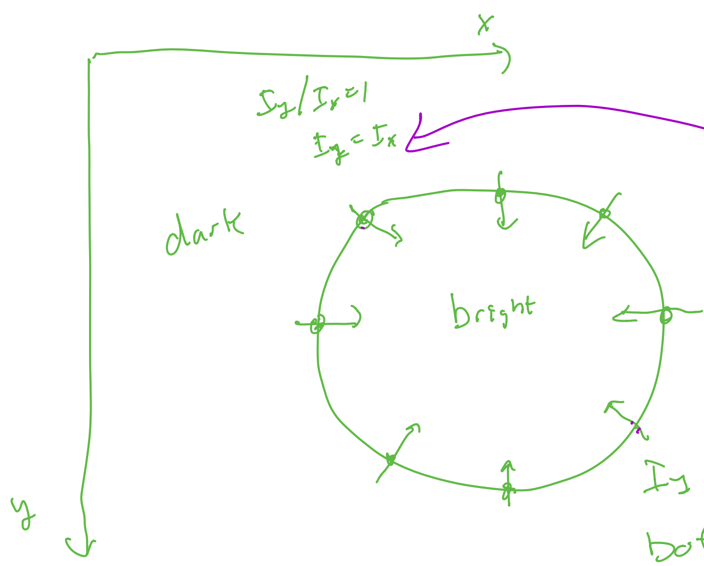
$I_x = -1$

$I_y / I_x = 1$

$I_y = 1 \quad I_x = 1$

$\tan^{-1}(I_y / I_x) = 45^\circ$

$\pi/4$



$\text{atan2}(-1, -1) = -3\pi/4$

$\sim 135^\circ$

$I_y = I_x$

both negative

$\Rightarrow -3\pi/4$

-135°

Other Operators

- As we've discussed, other partial derivative operators have been used. For example, the Sobel and Prewitt $\partial f/\partial x$ kernels are

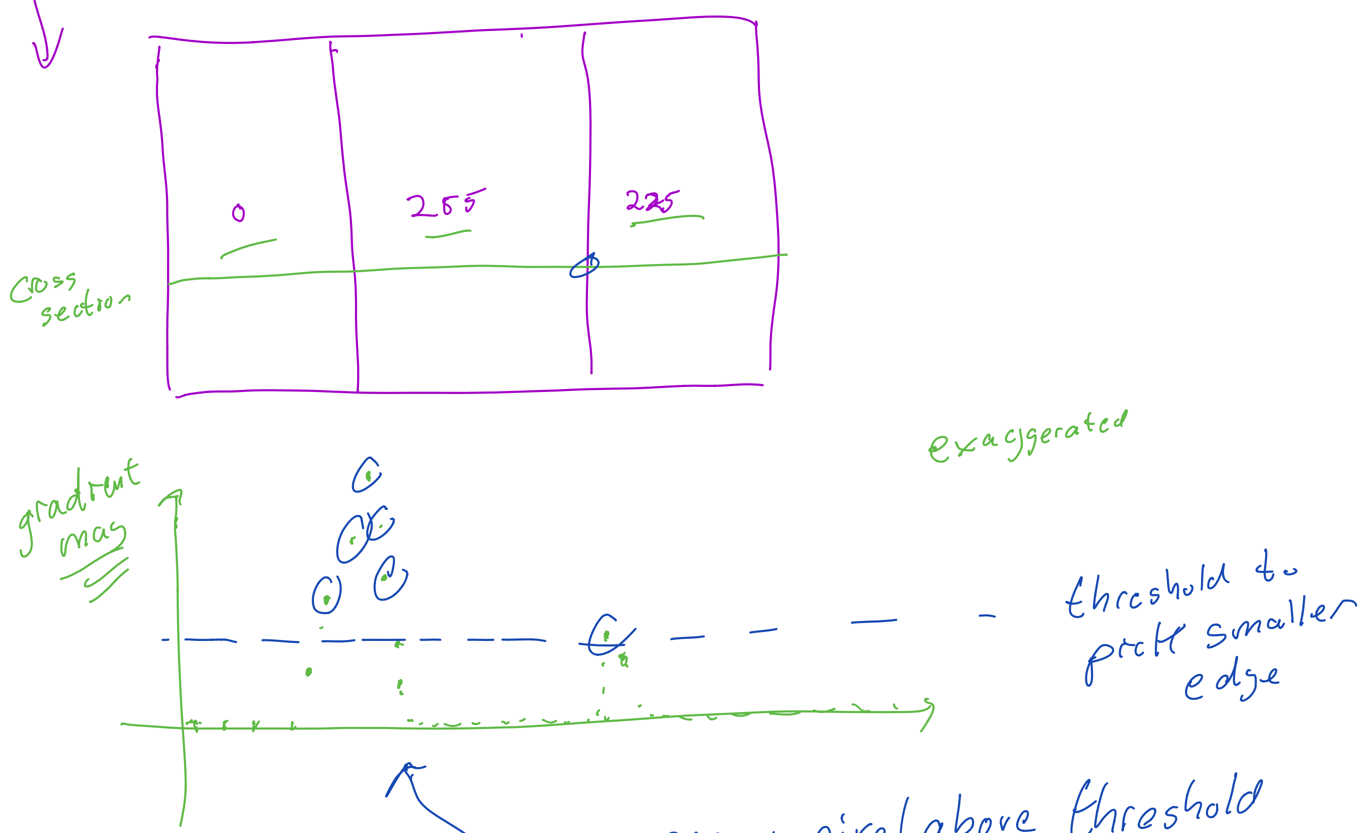
$$\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{pmatrix}$$

respectively.

- Notice that there is implicit smoothing in these operators, and the magnitudes of the derivatives are scaled up.

Additional Steps are Needed Following Smoothing and Gradient Computation

- After gradient computation, we still just have an image
- Pure thresholding leaves us with pixel locations of strong gradients, not edges (edge elements).
- We want to "extract" the edges, both individually and in contours.
- Decisions about what is a true edge and what is not may be made in groups of edgels.
- We'll sketch a picture in class to illustrate this.



Many pixel above threshold
⇒ "thick" edge

solve w non-maximum suppression!

Non-Maximum Suppression

- For each pixel, suppress it as a "non-edge" if a neighboring pixel has a stronger gradient.
- We'll start with a 1-d image where the gradients are shown in the following (with the column index above the boxes and the gradient magnitude in the boxes):

gradient may x-section

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
~~4~~	~~6~~	~~7~~	~~8~~	~~11~~	~~15~~	18	~~17~~	~~8~~	~~6~~	~~4~~	~~8~~	10	~~8~~	~~4~~

and demonstrate the importance of non-maximum suppression and not recording the values "in-place" in the array.

must compare to "suppressed possible edge locs

if threshold is 8 we get two edges/peaks

"thinning"

- In 2d it is important to think about what should should remain after non-maximum suppression.

- We will focus our discussion on the following example where the gradient magnitudes are shown as integers (assume blanks boxes have gradient 0):

gradient mag image!

	0	1	2	3	4	5	6	7
0						85	94	71
1					78	100	96	
2				55	82	98	80	
3			49	80	102	99	65	
4			71	100	104	71		
5		72	90	89	68			
6	73	91	92	76				
7	80	99	80	45				

smaller

smaller

What if 8 connected nbrs and do NMS ?

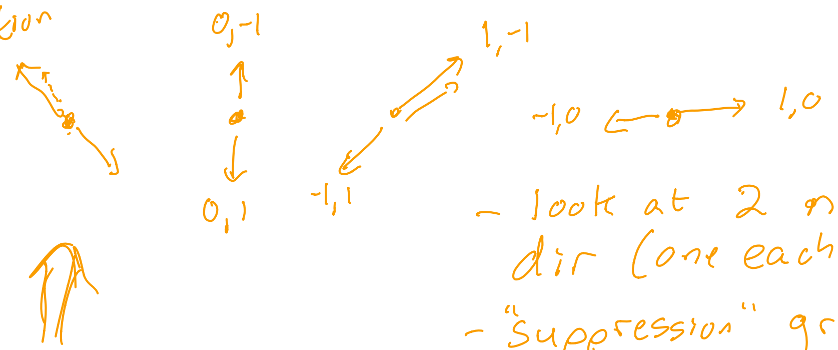
8-connected get 3 peaks

4-connected get 4

- Discussion:
  - What happens if we don't consider gradient direction during non-maximum suppression? get blue
  - How to use gradient direction? add orange
  - Discretization effects must be considered here, since the determination of who is a "neighbor" may not be symmetric.

- Finally, candidate edgels are the pixels that are left after non-maximum suppression.
  - This can include locations with very low gradient magnitudes!

trick to use grad dir to do NMS → grad dir to choose one of 4 pairs of direction -1,-1 and 1,1



- look at 2 nbrs along dir (one each way)

- "suppression" grad (as non max) if one of these nbrs has larger gradient

Sub-Pixel Localization of Edges

Still only know pixel locations of peaks, so...

- At each surviving peak, fit a parabola to the gradient magnitudes along the gradient direction.
- Offset the edge location along the normal based on the location of the peak.
- Yields the subpixel edge location
- While fitting a parabola sounds like an expensive computation, we will see in class that it reduces to straightforward arithmetic.
- Per usual we will start with 1d and then consider 2d.

Edge Linking

Group edgels into chains:

- Tangent direction is 90° rotated from the gradient direction.
- Record "next" and "previous" edgels at each edge (think of a doubly-linked list) — the edges "ahead" and "behind" the current one.
- Resolving ambiguities:
  - Find candidates for linking based on consistency of position **and** orientation.
  - If there are two candidates ahead (or behind) and they are consistent with each other, choose the four-connected neighbor first
  - If they aren't consistent, then this is a branching location. We can either
    * Terminate the chain or
    * Attempt to find the stronger link and continue it.
- Result is a series of edgel chains.

Edge Thresholding and Hysteresis

We still haven't decided which points are really "edgels", so we need to apply a threshold:

- Could use a single global threshold, perhaps computed from the image gradient statistics.
  - For example, assume the edgels occupy no more than a certain percentage of the image and use the histogram to compute a threshold.
- Could compute different thresholds in different (but overlapping) regions, perhaps interpolating between them to generate (potentially) different thresholds at each pixel.
- Double thresholding — also called "hysteresis" thresholding:
  - Two thresholds: $\theta_1 < \theta_2$.
  - All edgels with gradient magnitude below θ_1 are eliminated
  - Those above θ_2 are kept.
  - Those in between must be connected by a chain of edgels with strengths above θ_1 to an edgel with strength above θ_2 in order to survive.

Examples

Using Python and OpenCV, we will examine edge detection results on a number of images and consider how well it works.

Summary: How Well Does it Work?

- When thinking about how we want a vision system to work, we think "boundaries" instead of "edges".
- Therefore, we want "edges" where there are no significant intensities changes, and we want to ignore many prominent edges.
- Some of this can be seen in the following examples from Martin, Fowlkes, Malik, IEEE T-PAMI 2004:

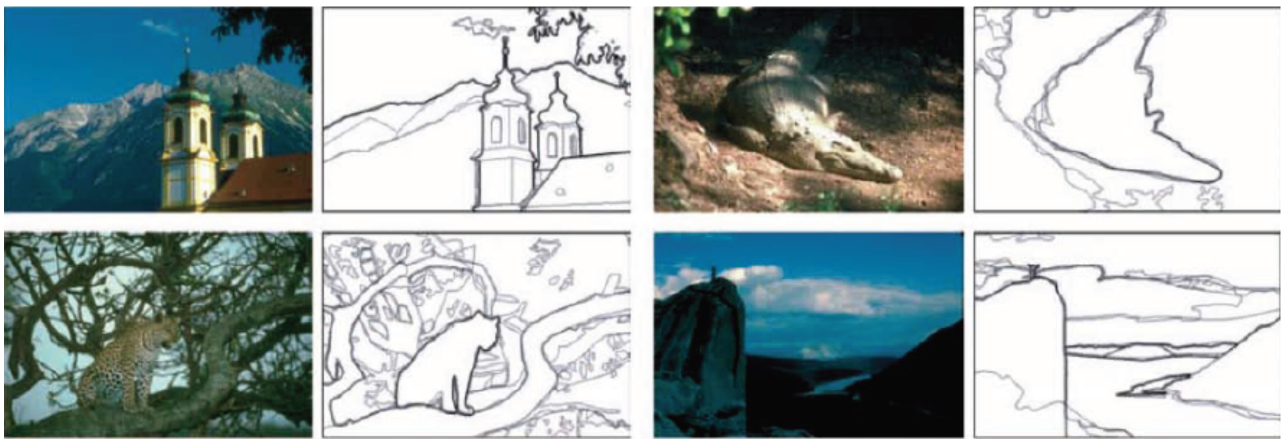


Figure 1: How well does edge detection work?