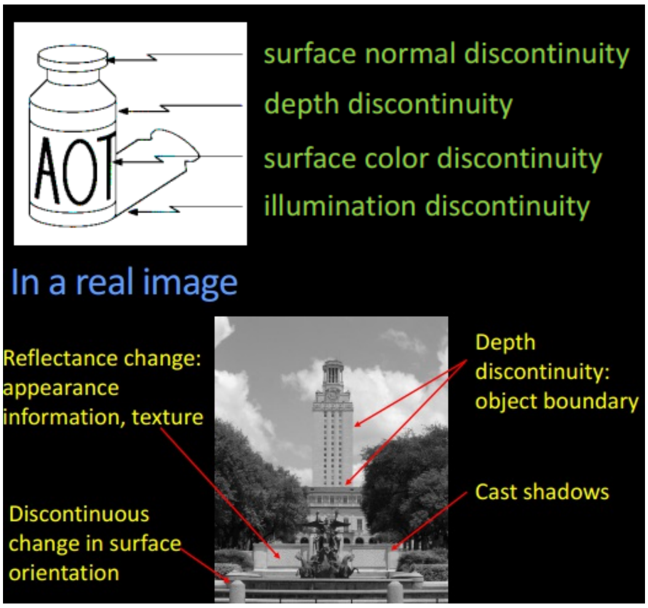
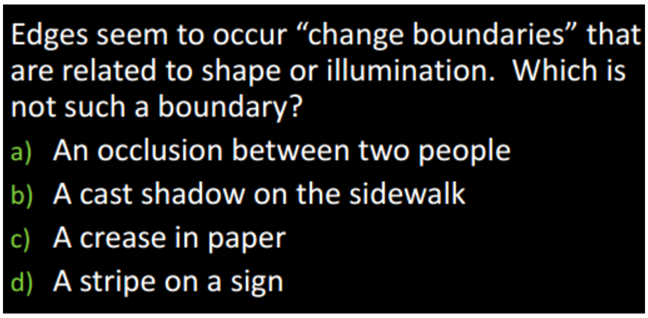
2A-L5 Edge detection - Gradients

2017/11/11 00:41

1. Sum
   a. to find/recognize the obj, the obj boundary helps.
   b. edge: where discontinuity happens: color, surface, depth, lumination
   c. finit difference
   d. gradient filter: sobel
   e. use gradient to help find the edge
      i. smooth
      ii. gradient
         1. derivative of the filter first to save the computation
      iii. 2nd gradient to find the max
2. Intro
   a. working on the problem, to find the obj on generic images which is not suitable through a template
3. Reduced Images
   a. Edges are important and they give a lot of info. sometimes
4. Edges
   a. origin
      i. surface normal discontinuity
      ii. depth discontinuity
      iii. color discontinuity
      iv. illumination discontinuity

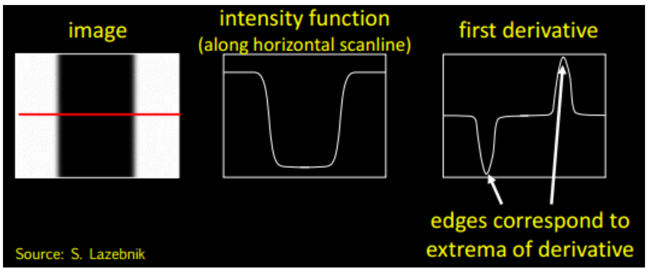


5. Quiz: Change Boundaries Quiz



i. analysis
1. An occlusion occurs when a physical object (or part of one) is in front of another, with respect to the viewing angle. This is part of the geometry or overall shape of the scene.
2. A shadow is caused due to difference in illumination falling on a surface.
3. A crease alters the shape of the paper, which is observable from certain viewing angles.
4. A stripe is essentially a color discontinuity, which is not caused by shape or illumination.

1. Edge Detection
    a. how to define a edge, or how can we determine certain pixel is on the edge?
        i. on an image, edges look like steep cliffs, pixels with a strong change in a neighborhood
    b. so the Basic idea: look for a neighborhood with strong signs of change.
        i. Problems
            1. neighborhood size
            2. how to define a change is a strong change which can be regarded as boundary
2. Derivatives and Edges
    a. when talking about changes in math, we rely on derivatives so
        i. An edge is a place of rapid change in the image intensity function, and this rapid change corresponds to the maximum of the derivative, see e.g.



ii. so the problem turns out to be find the peaks of the derivatives

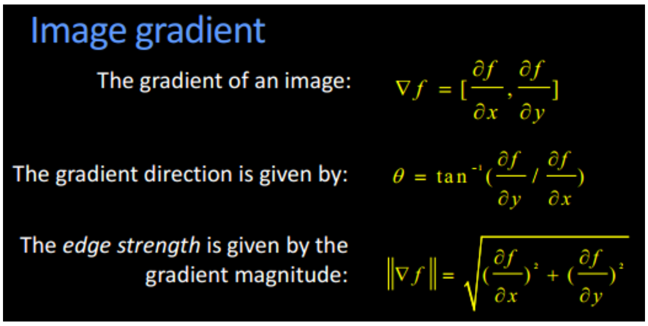
a. how to compute the derivative

i. differential operator, just like other filters

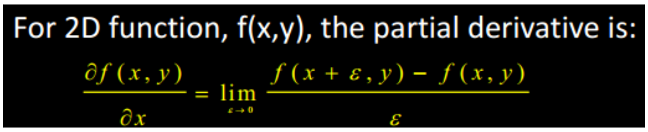
1. Differential operators －when applied to the image returns some derivatives.
2. • Model these “operators” as masks/kernels that compute the image gradient function.
3. • Threshold the this gradient function to select the edge pixels.

1. What is a Gradient

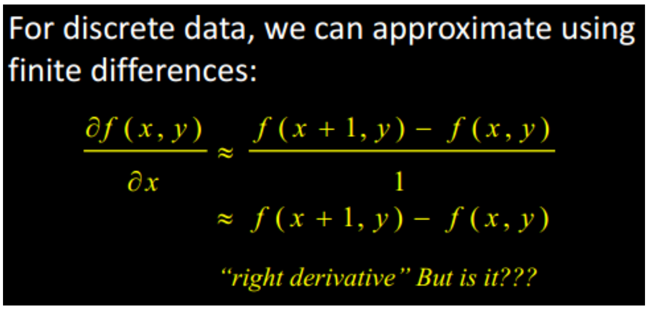
a. The gradient points in the direction of most rapid increase in intensity



9. Finite Differences

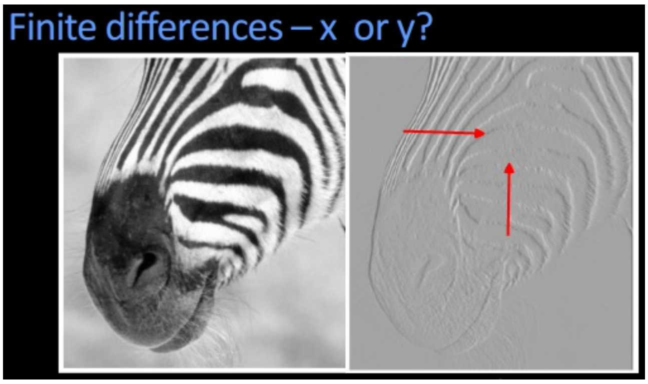


a. finite difference



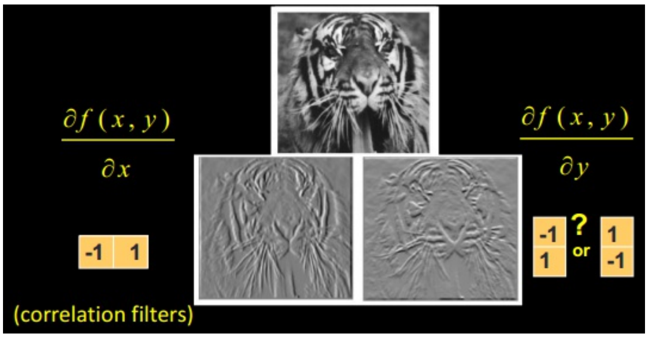
i. this is also called right derivative

a. e.g.



1. it's a finite difference on x, since horizontally, the edges are stronger than the vertical ones

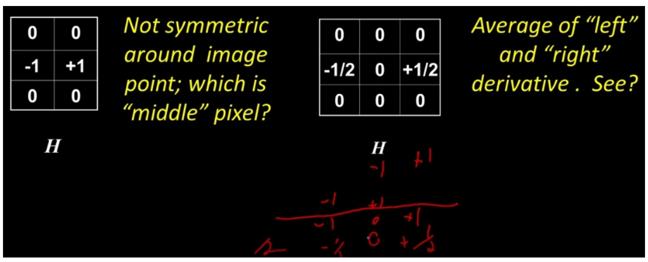
1. Partial Derivatives of an Image
    a. filters along the x, is [-1, 1]; along the y depends on your origin, if on the top-left corner, positive downside[CS choice]; if on the bottom left corner, positive upside[math choice].



    b. here correlation and convolution are different, be careful
    c. how does this [-1, 1] filter work?

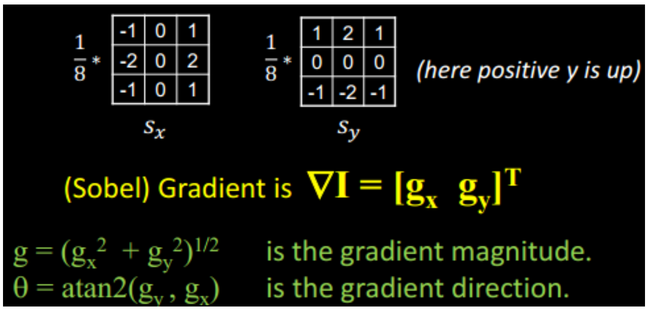
        i. it is an awful image for edge detection

1. The Discrete Gradient Filter
    a. what kind of filter should we use

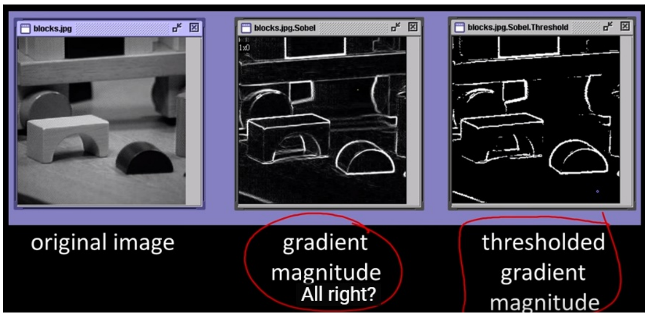


        i. the first is not suitable to find the middle pixel; so the left one, and the 1/2 is the average of the left and right derivatives

1. Sobel Operator



   a. the idea of adding the -1 & 1 is that we assume that images are sort of locally smooth so when compute the derivatives, it'e better to look around the neighborhood and then normalize the filter.
   b. imgradient in matlab applies sobel operator by default but it doesn't normalize it by dividing 8
   c. how does it work

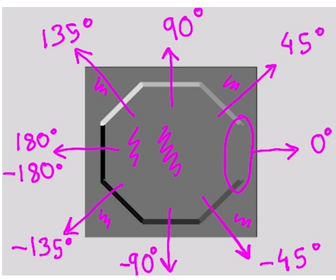


      i. it's not a great edge detection, but also not an awful one. So it's better than the previous one with [-1, 1]

1. Well Known Gradients
   a. matlad code
      i. filt = fspecial('sobel')
      ii. outim = imfilter(double(im), filt);
      iii. imagesc(outim);
      iv. colormap gray
   b. functions
      i. h = fspecial(type) creates a two-dimensional filter h of the specified type.
         1. sobel, prewitt, roberts
      ii. B = imfilter(A,h) filters the multidimensional array A with the multidimensional filter h
         1. The result B has the same size and class as A.
         2. B, using double-precision floating point. so we need the conversion
         3. by default, it uses correlation rather than convolution, but it can be changed.
      iii. imagesc(C) displays the data in array C as an image that uses the full range of colors in the colormap.Each element of C specifies the color for 1 pixel of the image.

iv. colormap maptype sets the colormap for the current figure to one of the predefined colormaps.
1. maptype: gray, spring, summer, .., copper , colorcube, ...

2. Quiz: Gradient Direction Quiz
   a. to compute, it's better to use correlation since it's easy to know the left and right ???
   b. matlab code
      i. %% Load and convert image to double type, range [0, 1] for convenience
         1. img = double(rgb2gray((imread('moon.jpg')))) / 255.;
            a. % assumes [0, 1] range for double images. since imgshow img range [0,1] or [min, max]
            b. or imshow(img, [0, 255])
            c. but if imshow(img), the range is [min, max] may cause some problem, just like imagsc()
         2. imshow(img);
      ii. %% Compute x, y gradients
         1. [gx gy] = imgradientxy(img, 'sobel'); % Note: gx, gy are not normalized
         2. figure(5);imshow((gx + 4)/8);
      iii. %% Obtain gradient magnitude and direction
         1. [gmag gdir] = imgradient(gx, gy);
         2. figure(2);imshow(gmag / (4 * sqrt(2))); % mag = sqrt(gx^2 + gy^2), so [0, (4 * sqrt(2))]
         3. figure(3);imshow((gdir + 180.0) / 360.0); % angle in degrees [-180, 180]



         4. %% Find pixels with desired gradient direction -- set the desired pixels 1, the

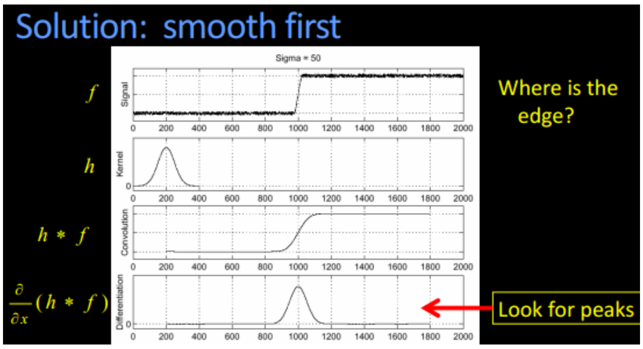
others 0

a. result = gmag > mag_min & gdir > angle_low & gdir < angle_high;

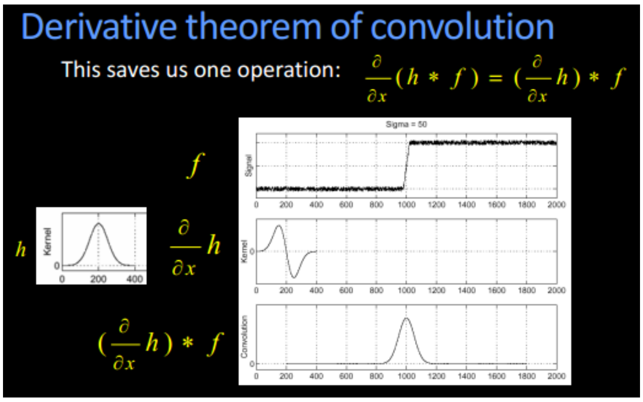
i. & is the element-wise logic AND when applied with matrix

ii. expr1 && expr2, && is for expression

1. But in the Real World
   a. Since in the real world, there're noise in the images, so we can't compute the gradient directly
   b. filter to smooth the images first and then gradient



i. to save the computation, do the conv first



c. edge is corresponds to the max of the gradient, so compute the second order derivative