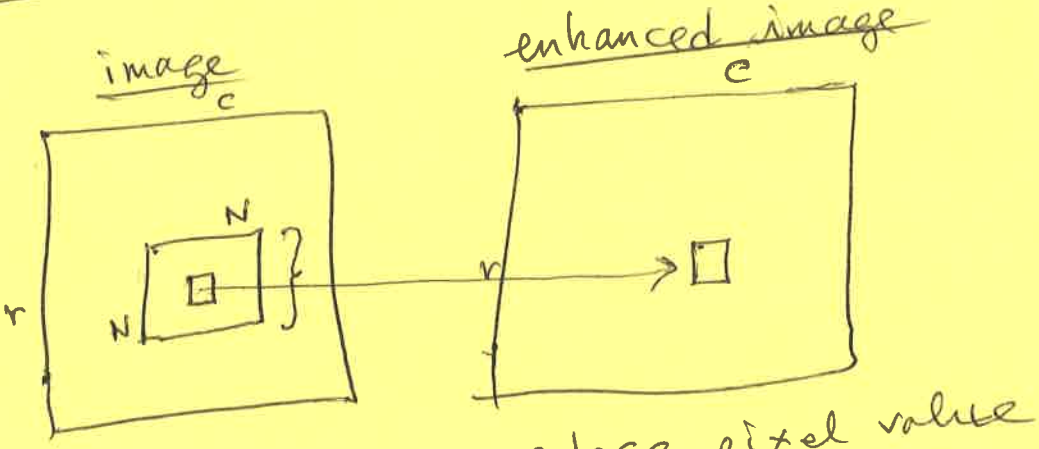
Chapter 4

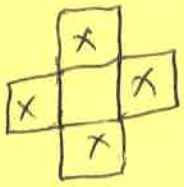
Enhancement: remove noise
sharpen image

Spatial filtering:

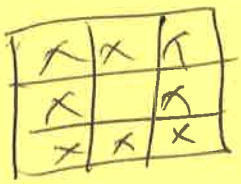


replace pixel value as a function of its neighborhood

Pixel neighbors



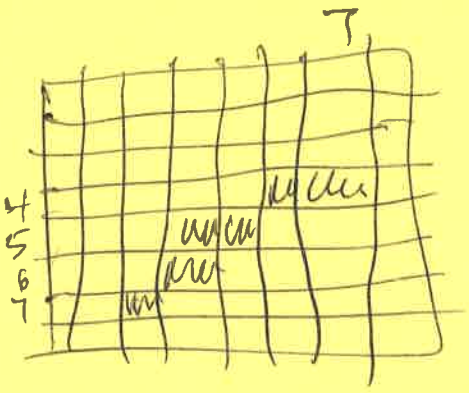
4-neighbors
center-to-center
distance = 1



8-neighbors
center-to-center
distance < 2

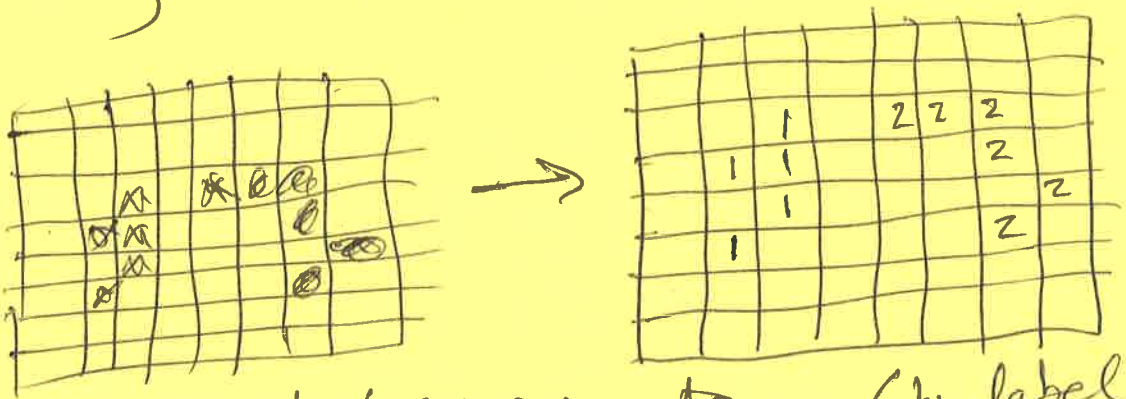
Connectivity

pixels (r_1, c_1) and (r_2, c_2) are <u>connected</u> if there exists a sequence of pixels from (r_1, c_1) to (r_2, c_2) such that each adjacent pair in the sequence are neighbors. (can be one of 4- or 8-neighbors, i.e., all the same)



(7,3) & (4,7) are connected

usually consider this in a binary image.



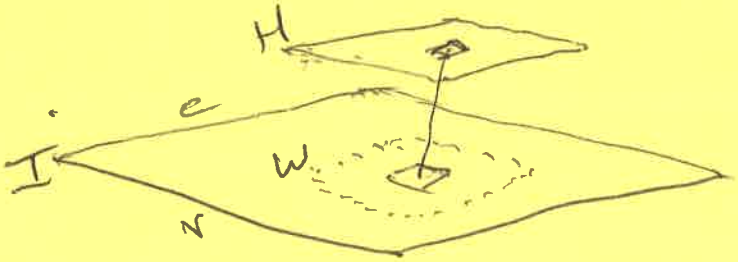
2 connected components (bw label)

Linear Filtering

H: an N x N array: filter, kernel, mask

I: an M x K array: image

* usually want H to have N an odd number
* to get an output value for (r, c):
  + position center of H over r, c
  + pointwise multiply corresponding pixels
  + sum the products



i.e., if W is N x N sub-image of I centered at (r, c):

$$\text{output}(r,c) = \sum_{k=1}^{N}\sum_{l=1}^{N} W(k,l) * H(k,l)$$

Matlab function fspecial creates filters

" " imfilter performs linear filtering

Nonlinear Filtering

compute nonlinear function on neighborhood

- * max
- * min
- * median
- * mode

} statistics of neighborhood

Some useful filters

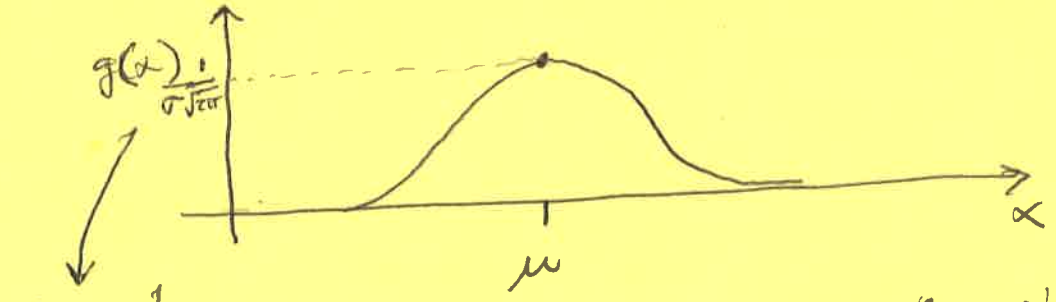
mean filter: ones(N, N) / N^2

\+ smooths image
− blurs image

median filter: values ← from N×N M
sort values
return middle value

Gaussian filtering: weighted average

1D $$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



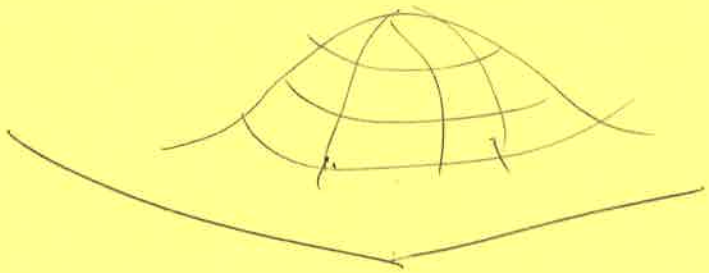
if $\sigma = \frac{1}{\sqrt{2\pi}}$
then $g(\mu) = 1$

see CS4640_Gaussian

2D Gaussian

$$G(x,y) = A \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \qquad \left(\text{book: } A = \frac{1}{2\pi\sigma^2}\right)$$

$$G(x,y) = A \exp\left(-\left(\frac{x-\mu(1)}{2\sigma_x^2} + \frac{y-\mu(2)}{2\sigma_y^2}\right)\right)$$

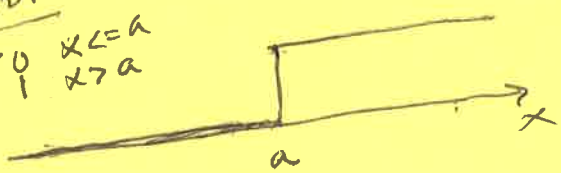


Used to smooth image (viewed as a surface)
CS4640 - week 4

Edge detection : discontinuities

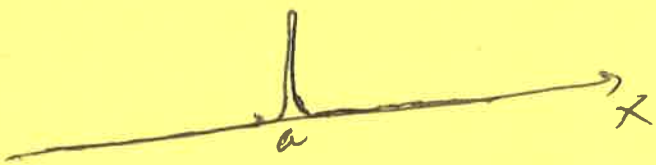
Derivative operator

$$f(x) = \begin{cases} 0 & x <= a \\ 1 & x > a \end{cases}$$



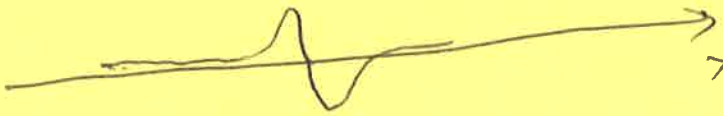
edge is at max value

$f'(x)$



edge is at zero crossing

$f''(x)$



Think about why smoothing is necessary

Derivatives in an image

$$\frac{\partial f}{\partial x} \quad \lim_{\Delta x \to} \quad \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

discrete

$$f(x+1,y) - f(x,y)$$

$$\Delta x = 1$$

$$\frac{\partial f}{\partial y} \quad \lim_{\Delta y \to 0} \quad \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$$

$$f(x, y+1) - f(x,y)$$

$\nabla f(x,y)$ — 2×1 vector

$$\begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

$$\begin{bmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{bmatrix}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) = \frac{\partial}{\partial x}\left[f(x+1,y) - f(x,y)\right]$$

? means

Consider $f \equiv$

$x-1$	x	$x+1$	
v_1	v_2	v_3	

$v_1 = f(x-1, y)$
$v_2 = f(x, y)$
$v_3 = f(x+1, y)$

$f' \Rightarrow$

$x-1$	x
$v_2 - v_1$	$v_3 - v_2$

$f'' \Rightarrow (v_3 - v_2) - (v_2 - v_1)$

$= v_3 - 2v_2 + v_1$

---

f $\quad f(x-1) \quad f(x) \quad f(x+1)$

$f(x) - f(x-1) \quad f(x+1) - f(x)$

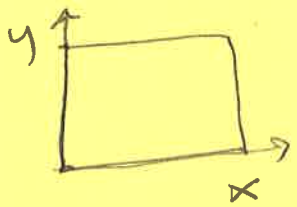
$f(x+1) - f(x) = (f(x) - f(x-1))$

$= f(x+1) - 2f(x) + f(x-1)$

$$\frac{\partial^2 f}{\partial y^2} \Rightarrow \quad f(x,y+1) - 2f(x,y) + f(x,y-1)$$

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f(x+1,y) - 2(f(x,y)) + f(x-1,y) + f(x,y+1) - 2f(x,y) + f(x,y-1)$$

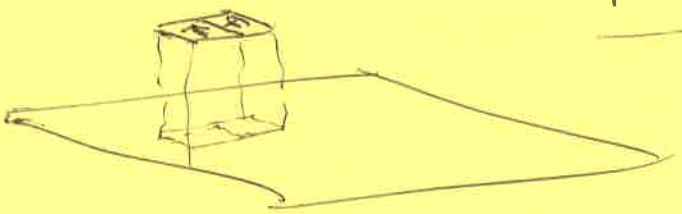
Rethink derivatives as filters



$$\frac{\partial f}{\partial x} = f(x+1,y) - f(x,y) \Rightarrow$$

-1	+1

point wise multiply and add



$$\frac{\partial f}{\partial y} =$$

1
-1

$$\frac{\partial^2 f}{\partial x^2} =$$

0	0	0
1	-2	1
0	0	0

$$\frac{\partial^2 f}{\partial y^2}$$

0	1	0
0	-2	0
0	1	0

$$\nabla^2 f =$$

0	1	0
1	-4	1
0	1	0

is sum of +

Note sum of values in filters is 0

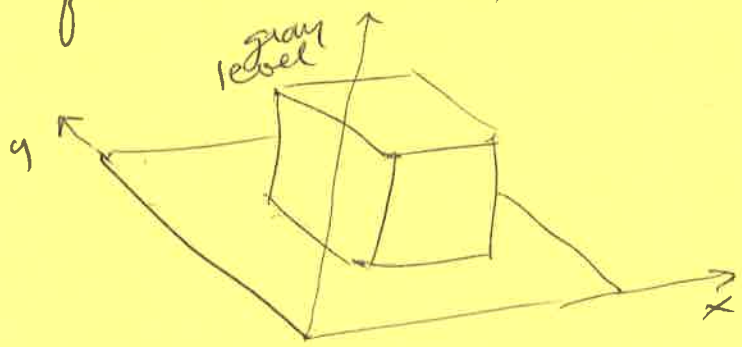
why?

Can use:

```
[dx, dy] = gradient(im);
[dxx, dxy] = gradient(dx);
[dyx, dyy] = gradient(dy);

mag = sqrt(dx.^2 + dy.^2);
ori = posori(atan2(dy, dx));

quiver(dx, dy);   shows gradient directions
```



gradient points in direction of function increase

Edge Detectors in the book

Roberts	X deriv	Y deriv
	0 -1	-1 0
	1 0	0 1

$mag = sqrt(G_x^2 + G_y^2)$

$ori = atan2(G_y, G_x) + \frac{\pi}{4}$

why?

Prewitt		
	1 0 -1	1 1 1
	1 0 -1	0 0 0
	1 0 -1	-1 -1 -1

Sobel		
	1 0 -1	1 2 1
	2 0 -2	0 0 0
	1 0 -1	-1 -2 -1

```
H = fspecial('Prewitt');
```

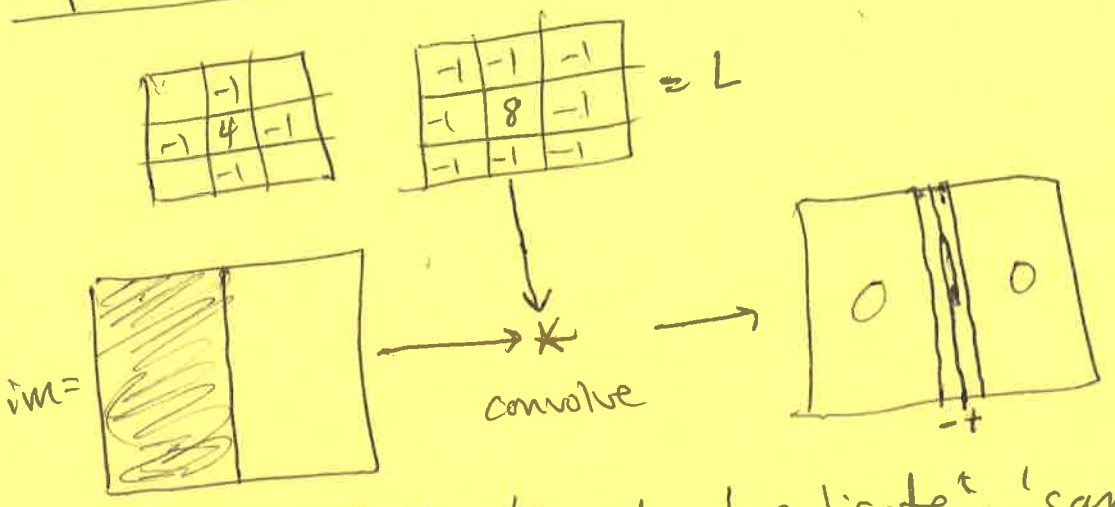
Call edge

```
edge(im, 'Prewitt');
```

4/10

2nd Order Edge detection

Laplacian edge detection



imL = imfilter(im, L, 'replicate', 'same');

as for all derivative operators, it is better to smooth image first:

Laplacian of Gaussian

fspecial

```
L = fspecial('log', 19, 2);
G = fspecial('gaussian', 19, 3) - fspecial('gausm', -
                    19, 1);
```

Compare effects of L vs. G.

surf L + G

Zero-Crossing

depends on type of data:
* noisy: smooth more
* crisp: don't smooth

$-x_{i+1}$

edge(im, 'zerocross', 0, L)
↑ closed contours

* to overcome noise, look at gradient at zero crossing (magnitude of gradient)

Edge Enhancement

Edge sharpening

subtract Laplacian from image $I - \text{lap}(I)$

or $I + k(I - I_{smoothed})$

```
G = fspecial('gaussian', 19, 3);
camb = imfilter(cam, G);
camen = cam + 0.5 * camb;
```

look at row 60