9-25

3:02

by we saw neighborhood operations early glimpse

HW 2

must

Neighborhood Operations

Prof. George Wolberg

Dept. of Computer Science

City College of New York

Objectives

- This lecture describes various neighborhood operations:
  - Blurring
  - Edge detection
  - Image sharpening
  - Convolution

HW2
lots
of
programming

Neighborhood Operations

no LUT

use kernal weights

neighborhood
↓
* use kernel
↓
add them up
↓
new pxl

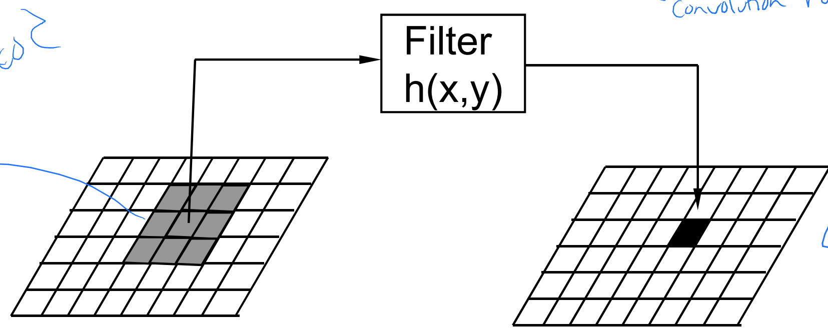
- Output pixels are a function of several input pixels.
- h(x,y) is defined to weigh the contributions of each input pixel to a particular output pixel.
- g(x,y) = T[f(x,y); h(x,y)]

Filter (its putting weights on input pixels)

g(x,y)=T[f(x,y); h(x,y)]=f(x,y)*h(x,y)

convolution not multiplication

neighborhood?



Input: f(x,y)

Output: g(x,y)

Spatial Filtering

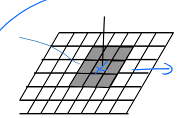
- h(x,y) is known as a *filter kernel, filter mask, or window.*
- The values in a filter kernel are coefficients.
- Kernels are usually of odd size: 3x3, 5x5, 7x7

Why odd?
want to make them symmetric

- This permits them to be properly centered on a pixel
  - Consider a horizontal cross-section of the kernel.
  - Size of cross-section is odd since there are 2n+1 coefficients: *n* neighbors to the left + n neighbors to the right + center pixel

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

each # has weight



Input: f(x,y)

move it over and u get new neighborhood

Spatial Filtering Process

- Slide filter kernel from pixel to pixel across an image.
- Use raster order: left-to-right from the top to the bottom.
- Let pixels have grayvalues f_i.
- The response of the filter at each (x,y) point is:

old tvs
electron gun

$$R = h_1 f_1 + h_2 f_2 + \ldots + h_{mn} f_{mn} \quad \longleftarrow \textbf{1D indexing}$$

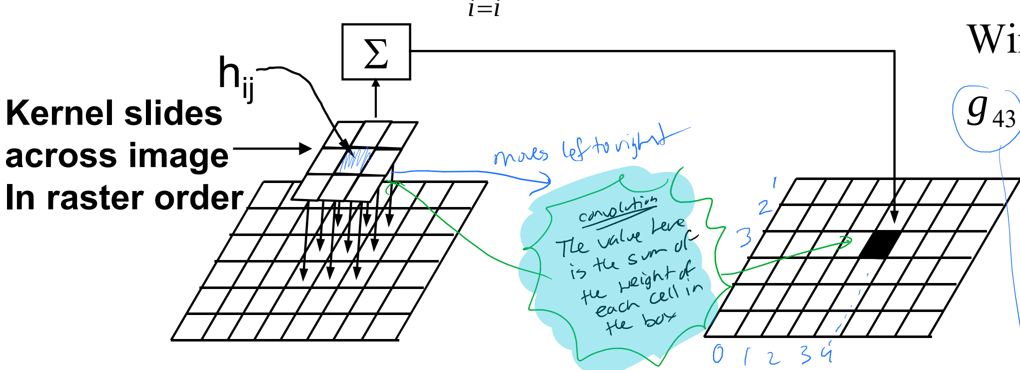
$$= \sum_{i=i}^{mn} h_i f_i$$

2D indexing

Window centered at (4,3)

$$g_{43} = h_1 f_{32} + h_2 f_{33} + h_3 f_{34} + h_4 f_{42} + h_5 f_{43} + h_6 f_{44} + h_7 f_{52} + h_8 f_{53} + h_9 f_{54}$$

h = weights



Wolberg: Image Processing Course Notes

row 4
col 3

When you move it over
you get g_{53} with a new neighborhood

Moving average: only the window moves (the group of pixels you look at)
also used in finance

Linear Filtering

- Let f(x,y) be an image of size *MxN.*
- Let h(i,j) be a filter kernel of size *mxn.*
- Linear filtering is given by the expression:

running sum

$$g(x, y) = \sum_{i=-s}^{s} \sum_{j=-t}^{t} h(i, j) f(x + i, y + j)$$

offsets from index x and y

where s = (m-1)/2 and t = (n-1)/2

- For a complete filtered image this equation must be applied for x = 0, 1, 2, … , M-1 and y = 0, 1, 2, … , N-1.

Spatial Averaging

- Used for blurring and for noise reduction
- Blurring is used in preprocessing steps, such as
  - removal of small details from an image prior to object extraction
  - bridging of small gaps in lines or curves
- Output is average of neighborhood pixels.
- This reduces the “sharp” transitions in gray levels.
- Sharp transitions include:
  - random noise in the image
  - edges of objects in the image
- Smoothing reduces noise (good) and blurs edges (bad)

3x3 Smoothing Filters

- The constant multiplier in front of each kernel is equal to the sum of the values of its coefficients.
- This is required to compute an average.

Weight assigned by closeness / importance

Sum of values in box

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

Box filter

all are the same

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Weighted average

these pixels are close its like 1

they are further away
· thats why this pixel has less weight / less important

The center is the most important and other pixels are inversely weighted as a function of their distance from the center of the mask. This reduces blurring in the smoothing process.

convolution
The value here is the sum of the weight of each cell in the box

Some pixels more important than others

multiply $\frac{1}{16}$ to each value in the box

Unweighted/Weighted Averaging

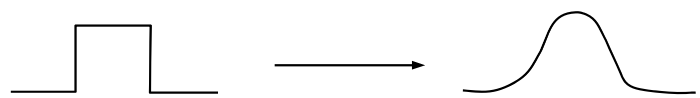
blur

will use
blur to
make sharper

- Unweighted averaging (smoothing filter):

$$g(x,y) = \frac{1}{m}\sum_{i,j} f(i,j)$$

- Weighted averaging:



$$g(x,y) = \sum_{i,j} f(i,j)h(x-i,y-j)$$

Original image

7x7 unweighted averaging

7x7 Gaussian filter

10–7
rec fine idq

2:05 aftr

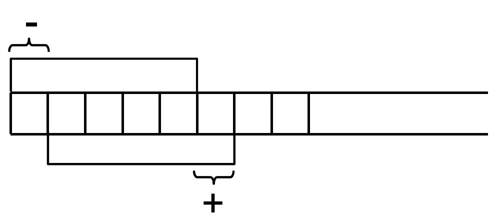
Unweighted Averaging

- Unweighted averaging over a 5-pixel neighborhood along a horizontal scanline can be done with the following statement:

```
for(x=2; x<w-2; x++)
    out[x]=(in[x-2]+in[x-1]+in[x]+in[x+1]+in[x+2])/5;
```

- Each output pixel requires 5 pixel accesses, 4 adds, and 1 division. A simpler version (for unweighted averaging only) is:

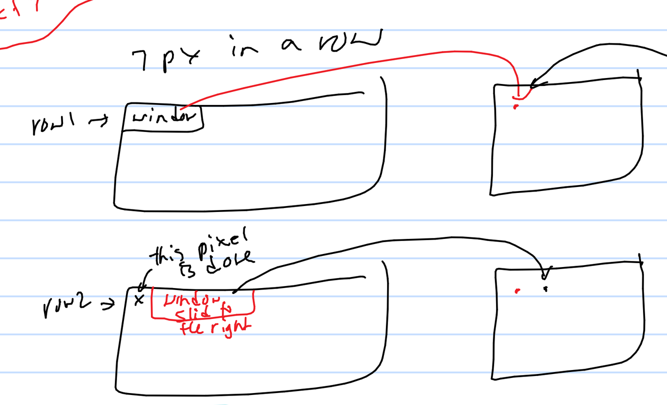
```
sum=in[0]+in[1]+in[2]+in[3]+in[4];
for(x=2; x<w-2; x++){
     out[x] = sum/5;
     sum+=(in[x+3] – in[x-2]);
}
```



Limited excursions reduce size of output

2:05 oct 7

7 px in a row

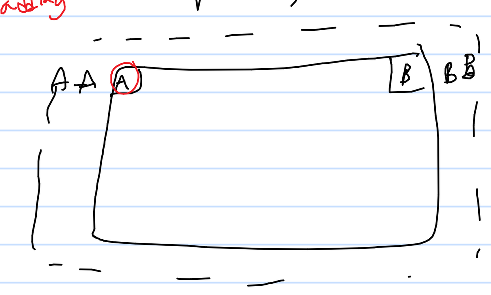


· row1 : take all pxls in the whole and output avg to

· keep moving window till row done
· then restart window in next row

Border replecation
If the edge pxls is A the padding is also A

padding



$\text{blur}_x(\text{in})$ — 1^{st} Pass Algo: bluring in horizontal process

$\downarrow$

$\text{blur}_y(\text{blur}_x(\text{in}))$ — 2^{nd} pass Algo: bluring in verticle process

everything in image proces is 2^{nd} pass algo

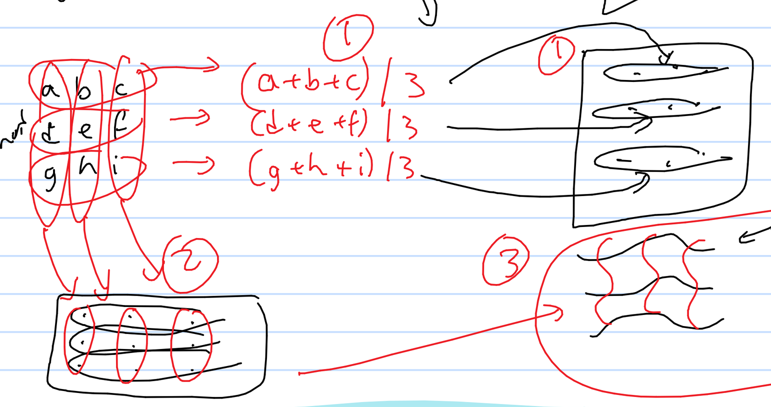
ex
a b c
d e f
g h i
$\rightarrow \frac{a+b+\cdots+i}{9}$

same thing but ↓ is faster

2:15

this method won't work for gaussein weighting (pixel e weighs more than f)

less pixel access when neighbourhood bigger



(1) $(a+b+c)/3$
$(d+e+f)/3$
$(g+h+i)/3$

blur horizontally

weight in gaussing verticle pass

2:22

$(I * k_1) * k_1$ (input image; sum of prods)

$= I * (\underbrace{k_1 * k_1}_{k_2})$

$= I * k_2$

Convolution of the neighborhoods)

convuling 3 pixel neigh with another 3px neigh

1/3	1/3	1/3

$\frac{1}{3}\cdot 0 \quad \frac{1}{3}\cdot\frac{1}{3}$

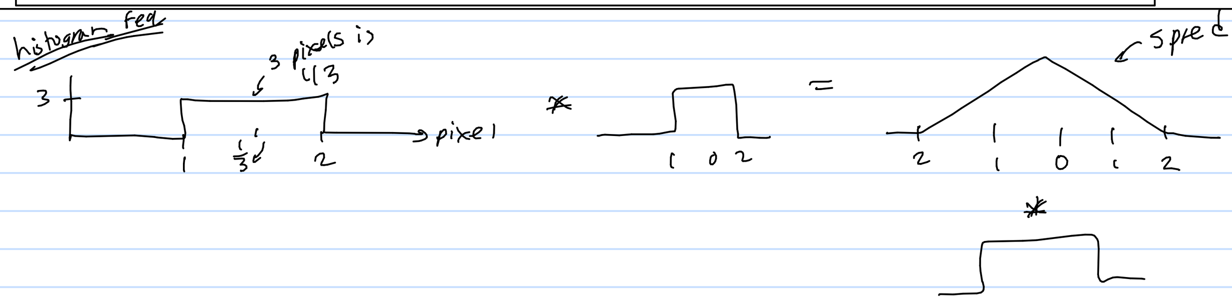
$0 + 2/9 = 2/9$

$= 3/9$

$= 2/9$

$\frac{1}{9}, \frac{2}{9}, \frac{3}{9}, \frac{2}{9}, \frac{1}{9}$

this is a triangle



If you keep convolving result with ⎍ You will get gaussian wave!!!

ex

$\wedge * \boxed{\tfrac{1}{3}\ \tfrac{1}{3}\ \tfrac{1}{3}}$

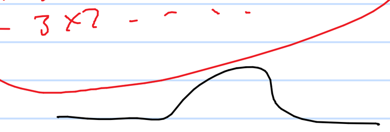
remember this is Just horizontally!!! needa do it vertically tho

$\boxed{\tfrac{1}{3} \quad \tfrac{1}{3} \quad \tfrac{1}{3}}$

$\boxed{\tfrac{1}{9} \quad \tfrac{2}{9} \quad \tfrac{3}{9} \quad \tfrac{2}{9} \quad \tfrac{1}{9}}$ ← convoluting the triangle with a box again

comparing increases by 2
ex 3x3 comparison
now 3x5
then 3x7 . . . .

$= \frac{1}{27} \quad \frac{3}{27} \quad \frac{6}{27} \quad \frac{7}{27} \quad \frac{6}{27} \quad \frac{3}{27} \quad \frac{1}{27}$



verticle

1/27	3/27	6/27	7/27	6/27	3/27	1/27
1/27	3/27	6/27	7/27	6/27	3/27	1/27
1/27	3/27	6/27	7/27	6/27	3/27	1/27

$\frac{3}{27}$ . . . . .

2:34 go back

2:35
Oct 7 cont

Image Averaging

- Consider a noisy image g(x,y) formed by the addition of noise η(x,y) to an original image f(x,y):

  g(x,y) = f(x,y) + η(x,y)

- If the noise has zero mean and is uncorrelated then we can compute the image formed by averaging K different noisy images:

$$\bar{g}(x,y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x,y)$$

← resulting image = avg of sum

ex: take 100 pics of sky. each pixel has some random noise

add up 100 exact pics with random noise in each pic

↓

sum and then average them up

↓

output pic is better

- The variance of the averaged image diminishes:

$$\sigma^2{}_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2{}_{\eta(x,y)}$$

- Thus, as K increases the variability (noise) of the pixel at each location (x,y) decreases assuming that the images are all registered (aligned).

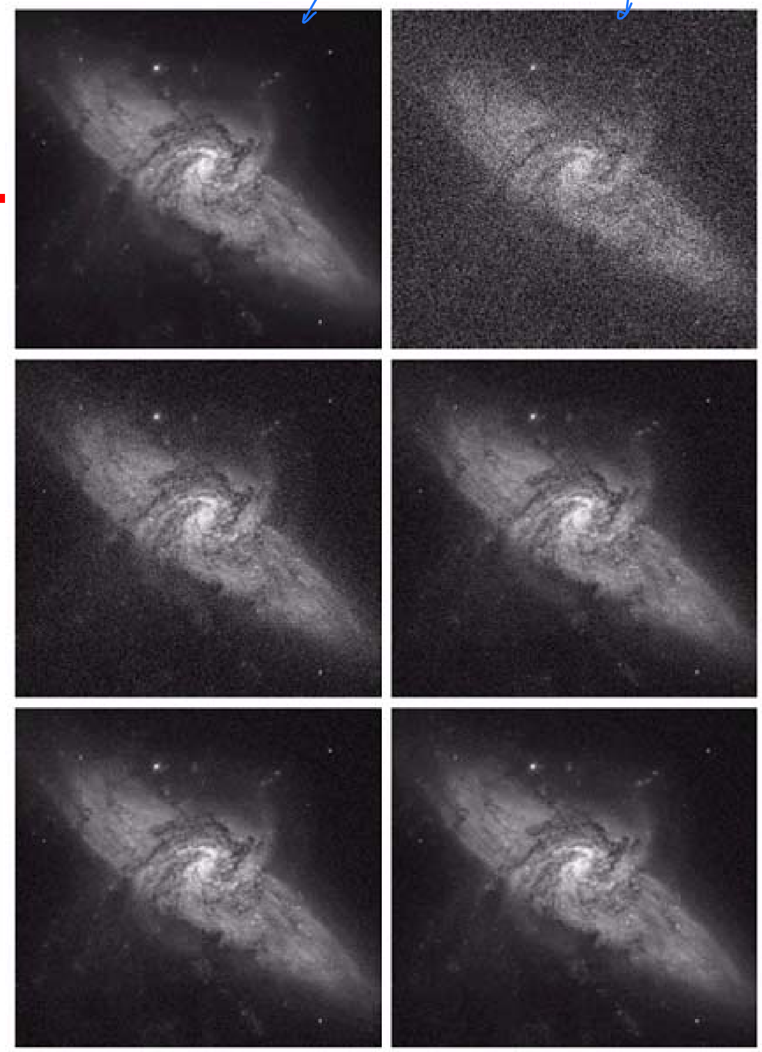
Noise Reduction (1)

Using bluring

- Astronomy is an important application of image averaging.
- Low light levels cause sensor noise to render single images virtually useless for analysis.

a

b (lots of noise)

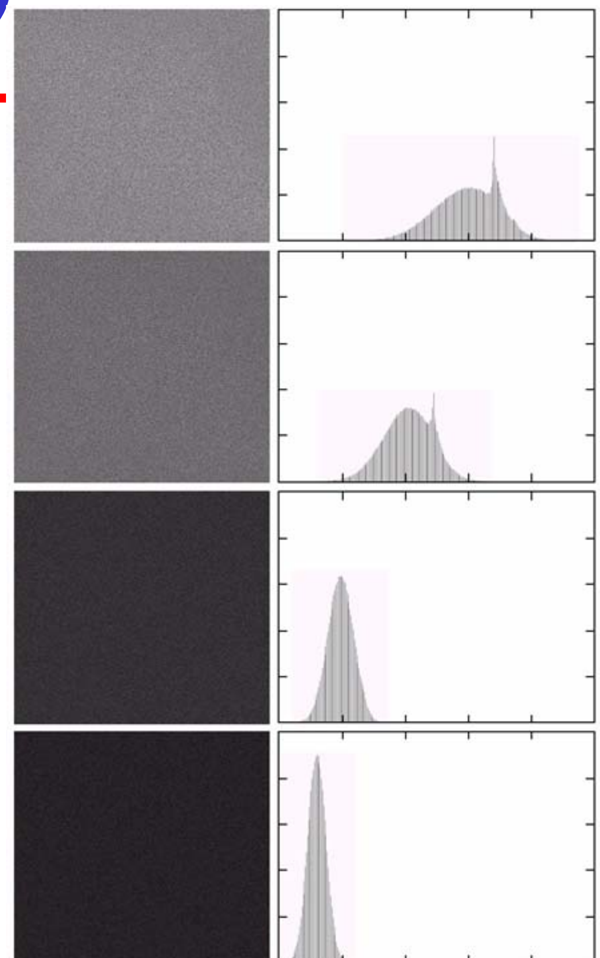


a b
c d
e f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64,$ and 128 noisy images. (Original image courtesy of NASA.)

Noise Reduction (2)

- Difference images and their histograms yield better appreciation of noise reduction.
- Notice that the mean and standard deviation of the difference images decrease as K increases.



a b

FIGURE 3.31 (a) From top to bottom: Difference images between Fig. 3.30(a) and the four images in Figs. 3.30(c) through (f), respectively. (b) Corresponding histograms.

smaller error

narrow so error confined to smaller interval

General Form: Smoothing Mask

2:40

i and j → for 3x3 neighborhood.

- Filter of size *mxn* (where *m* and *n* are odd)

$$g(x,y) = \frac{\sum_{i=-s}^{s} \sum_{j=-t}^{t} h(i,j) f(x+i, y+j)}{\sum_{i=-s}^{s} \sum_{j=-t}^{t} h(i,j)}$$

if the weight dont add up to 1, this part here will help to make weight 1?

ex for the $\boxed{1/3 \quad 1/3 \quad 1/3}$ neighborhood ex, the weights = 1 so wont need to use this.

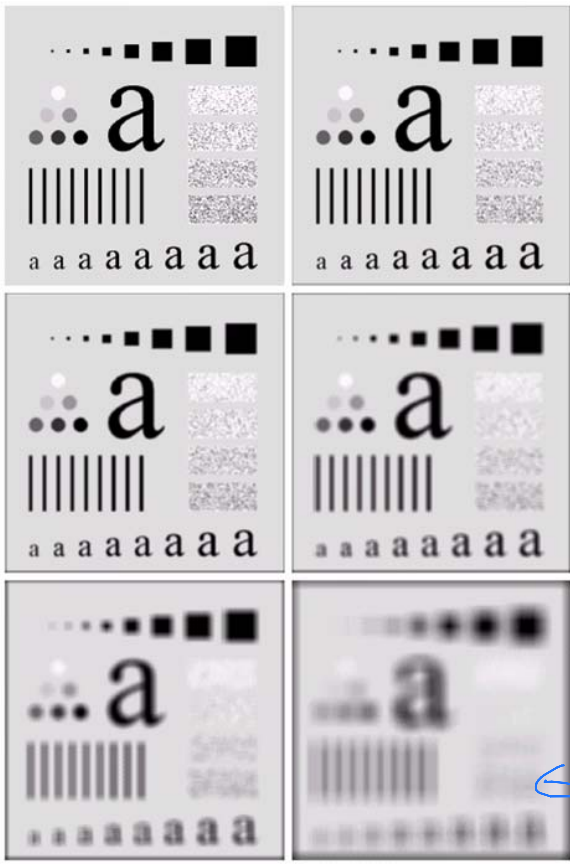
summation of all coefficients of the mask

Note that s = (m-1)/2 and t = (n-1)/2

Smoothing = blurring

Example

a	b
c	d
e	f

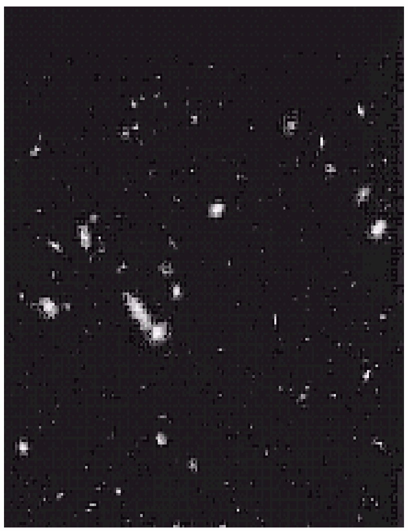


- a) original image 500x500 pixel
- b) - f) results of smoothing with square averaging filter of size n = 3, 5, 9, 15 and 35, respectively.
- Note:
  - big mask is used to eliminate small objects from an image.
  - the size of the mask establishes the relative size of the objects that will be blended with the background.

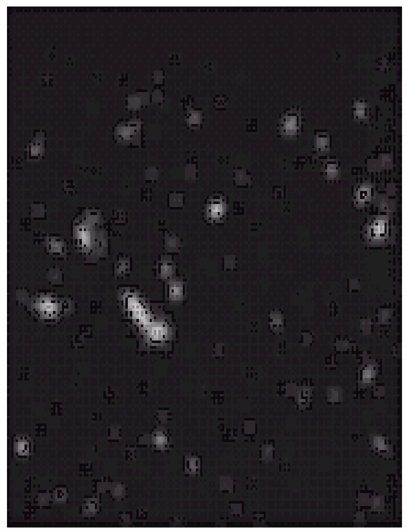
bigger neighborhood = larger blur

Example

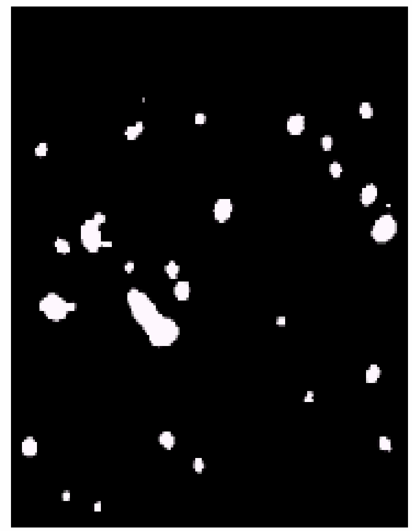
- Blur to get gross representation of objects.
- Intensity of smaller objects blend with background.
- Larger objects become blob-like and easy to detect.



original image



result after smoothing with 15x15 filter



result of thresholding

HW2: can use bluring to make image sharper

cool

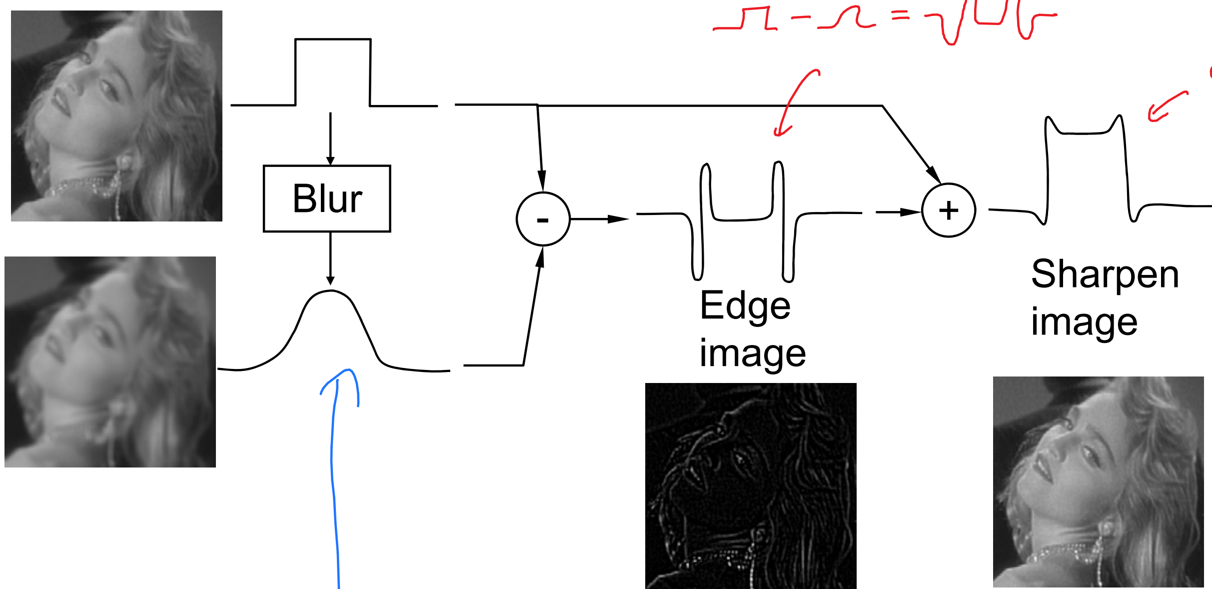
see text page

Unsharp Masking

- Smoothing affects transition regions where grayvalues vary.
- Subtraction isolates these edge regions.
- Adding edges back onto image causes edges to appear more pronounced, giving the effect of image sharpening.

edges

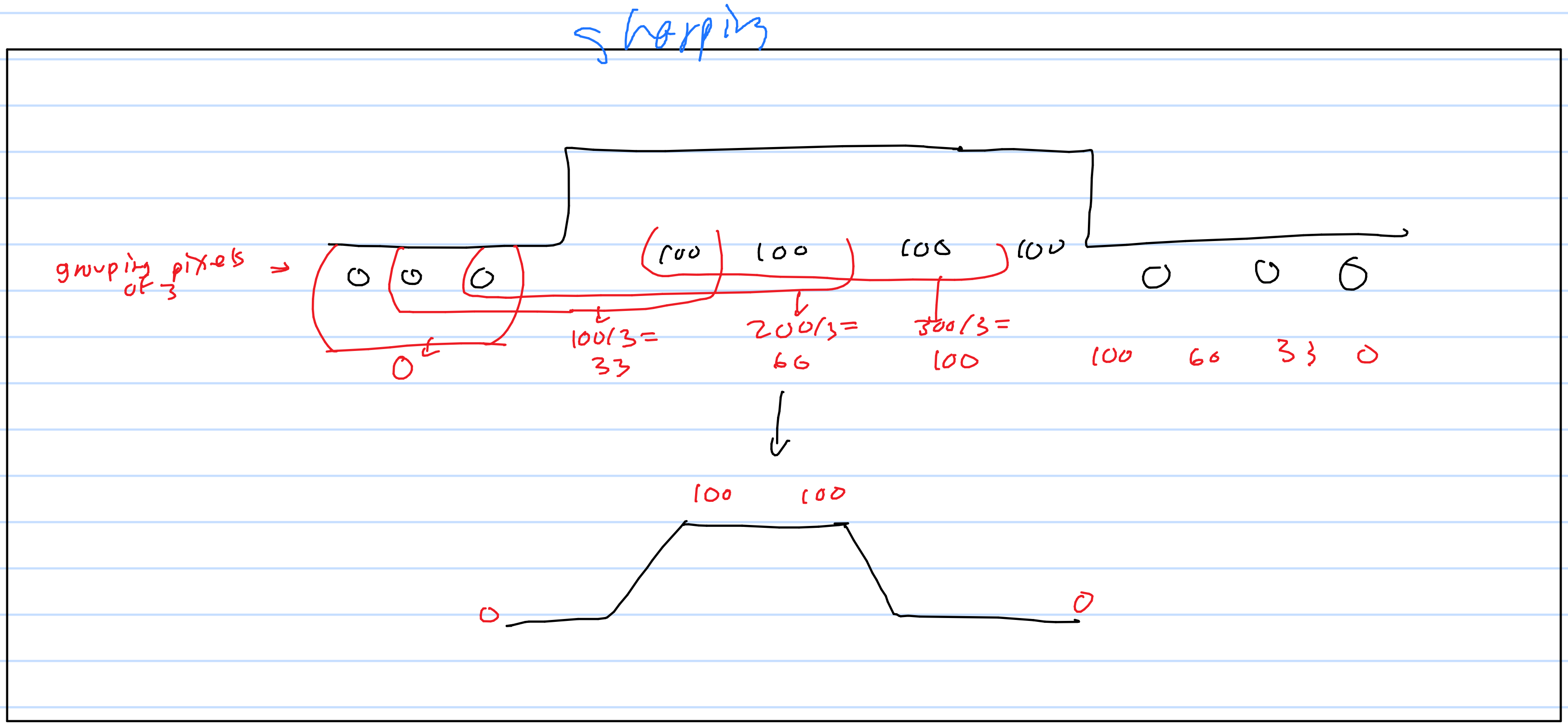
adding edges back to the original image

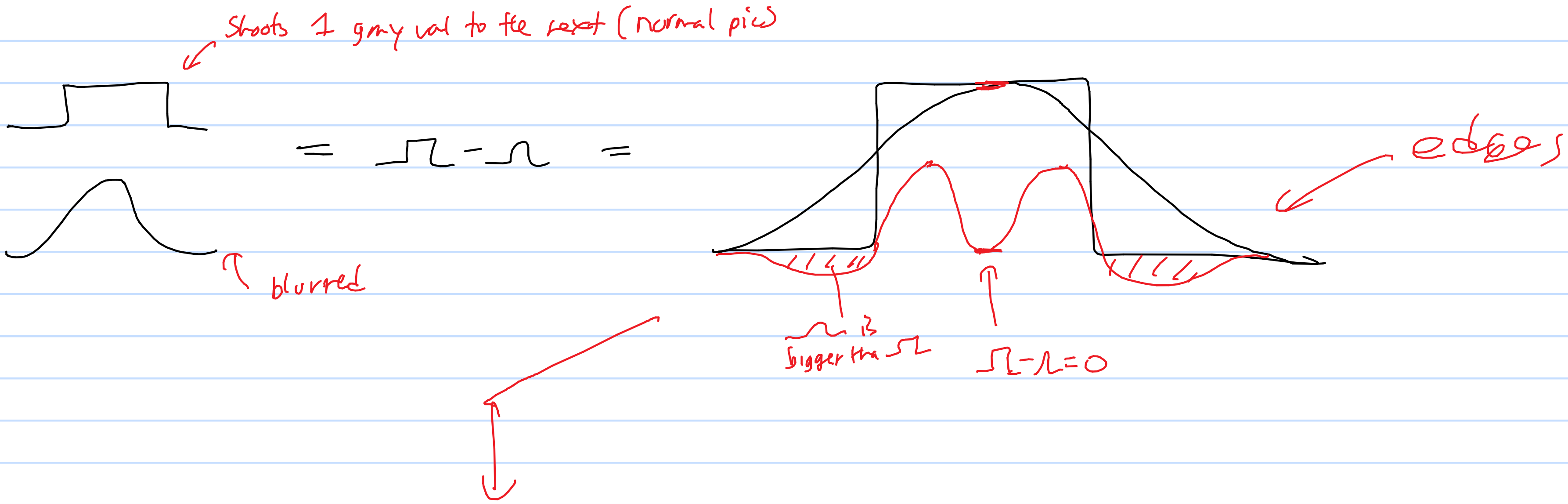


Filter size
how thick the edges?

factor size
how bright do you want the edges

blur used to extract edges





how add this back to image
to sharpen it

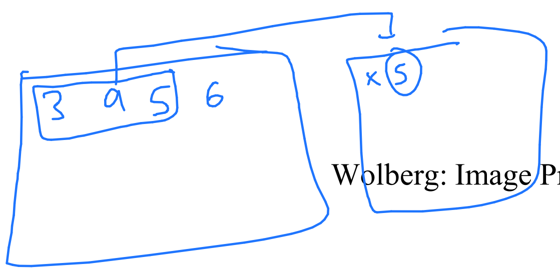
filter size → thin edges or large edges (thick brush strokes)

factor

Order-Statistics Filters

- Nonlinear filters whose response is based on ordering (ranking) the pixels contained in the filter support.
- Replace value of the center pixel with value determined by ranking result.
- Order statistic filters applied to *nxn* neighborhoods:
  - median filter: $R = \text{median}\{z_k \mid k = 1,2,\ldots,n^2\}$
  - max filter: $R = \max\{z_k \mid k = 1,2,\ldots, n^2\}$
  - min filter: $R = \min\{z_k \mid k = 1,2,\ldots, n^2\}$

move a 3x3 window → collect the median of the neighborhood and output it



Median Filter

= used to get rid of the noise

- When you pick a neighborhood, noise is usually at the beginning or end so picking median keeps you safe.
- however, if neighborhood is also noisy → need to broaden neighborhood or you might not be able to do anything else

- Sort all neighborhood pixels in increasing order.
- Replace neighborhood center with the median.
- The window shape does not need to be a square.
- Special shapes can preserve line structures.
- Useful in eliminating intensity spikes: salt & pepper noise.

center of neighborhood

10	20	20
20	200	15
25	20	25

(10,15,20,20,20,20,25,25,200)
Median = 20
Replace 200 with 20

Median Filter Properties

- Excellent noise reduction
- Forces noisy (distinct) pixels to conform to their neighbors.
- Clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $n^2/2$ (one-half the filter area), are eliminated by an *n* x *n* median filter.
- k-nearest neighbor is a variation that blends median filtering with blurring:
  - Set output to average of *k* nearest entries around median

10	18	19
20	200	15
25	20	25

median

(10,15,18,19,20,20,25,25,200)

k:0 return median

avg median + k to left and right

k=1: replace 200 with (19+20+20)/3

avg median from -k to +k

k=2: replace 200 with (18+19+20+20+25)/5

k=3: replace 200 with (15+18+19+20+20+25+25)/7

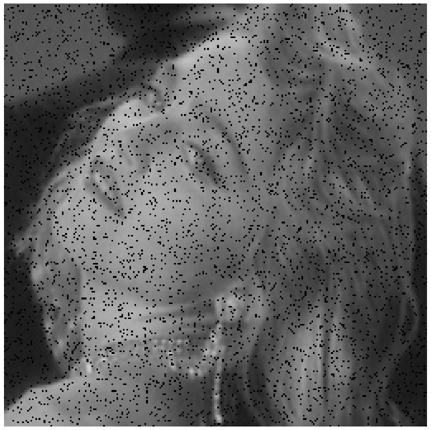
k=4: replace 200 with (10+15+18+19+20+20+25+25+200)/9

Salt and pepper noise

Examples (1)

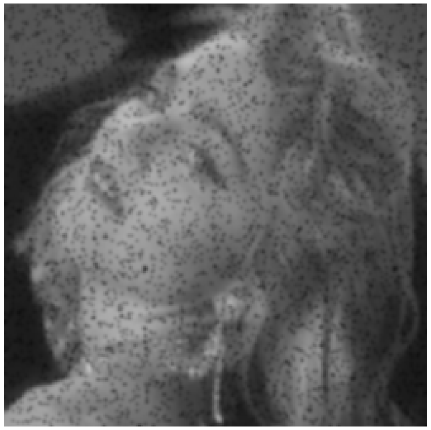
median filtering is better

Why not blur noise away? bad: will lose signal



Additive salt & pepper noise

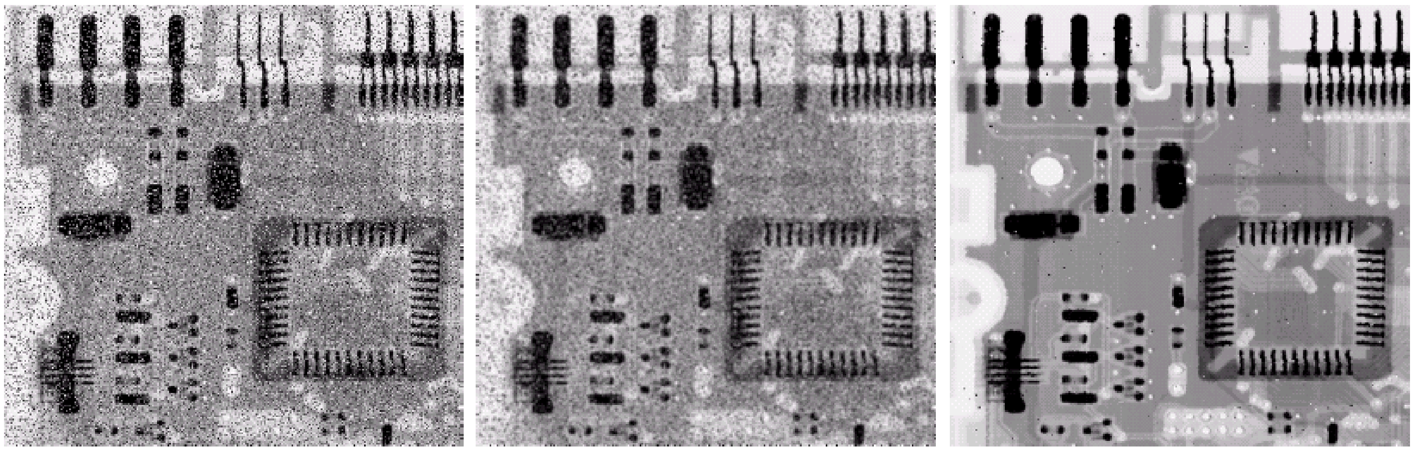
Median filter output



Blurring output

if even the median is noisy use larger size kernel

Examples (2)



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

may not need to sort by self
just use quick sort from
the languages.

10-16 2:15

Derivative Operators

- The response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied.
- Image differentiation
  - enhances edges and other discontinuities (noise)
  - deemphasizes area with slowly varying graylevel values.
- Derivatives of a digital function are approximated by differences.

First-Order Derivative

- Must be zero in areas of constant grayvalues.
- Must be nonzero at the onset of a grayvalue step or ramp.
- Must be nonzero along ramps.

position on curve

images are sometimes x, y, z dimention so you take partial

~~I think~~

$$\frac{dx}{dt} = \frac{f(x+\Delta) - f(x)}{\Delta}$$

your right neighbor minus yourself =

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

Δ is 1 bec the smallest change from 1 pixel of an image to another pixel is 1!

take the neighbor to the right and subtract you from it

Second-Order Derivative

- Must be zero in areas of constant grayvalues.
- Must be nonzero at the onset of a grayvalue step or ramp.
- Must be zero along ramps of constant slope.

$$\frac{\partial^2 f(x)}{\partial x^2} = \partial f(x) - \partial f(x-1)$$

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

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

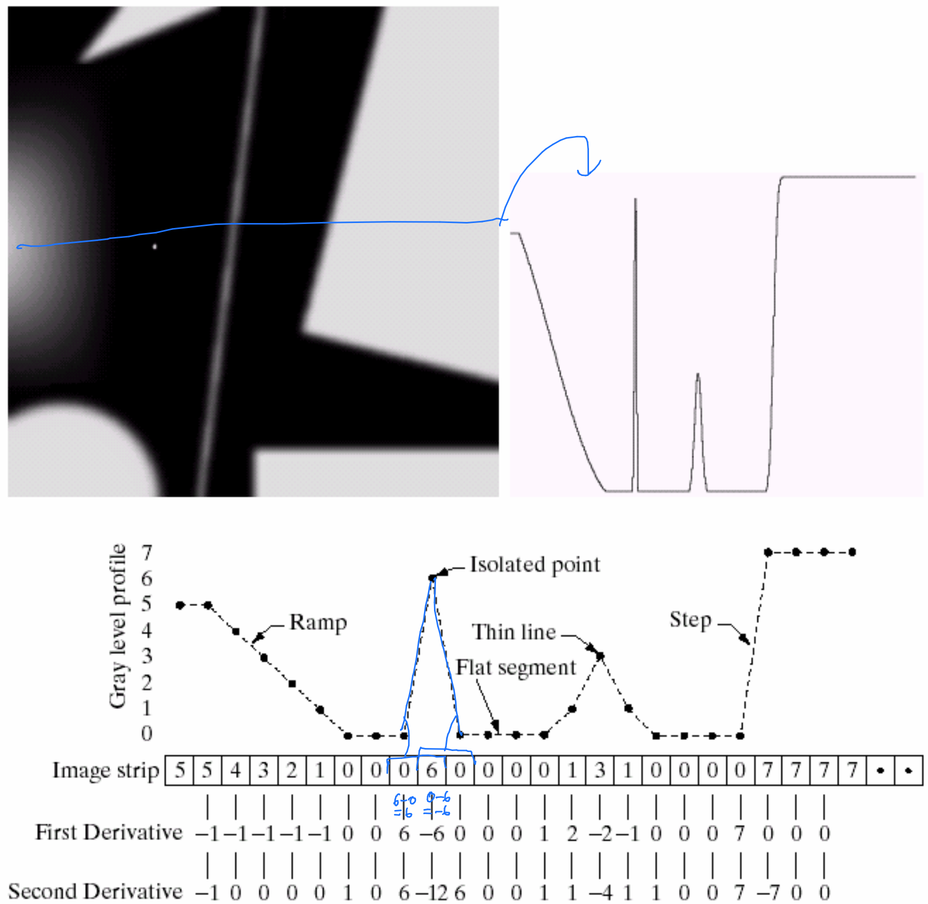
$\partial f(x-1) = f(x) - f(x-1)$

$f(x+1) f$

Example

a b
c

FIGURE 3.38
(a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).



Comparisons

- 1st-order derivatives:
  - produce thicker edges
  - strong response to graylevel steps
- 2nd-order derivatives:
  - strong response to fine detail (thin lines, isolated points)
  - double response at step changes in graylevel

Laplacian Operator

- Simplest isotropic derivative operator
- Response independent of direction of the discontinuities.
- Rotation-invariant: rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.
- Since derivatives of any order are linear operations, the Laplacian is a linear operator.

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

Discrete Form of Laplacian

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

where

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

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

neighbor to left

neigh to right

3x3 neighbor
ignore diagonal
neighbors bc
they are * by 6

$$\boxed{\begin{aligned}\nabla^2 f = [\, & f(x+1, y) + f(x-1, y) \\ & + f(x, y+1) + f(x, y-1) - 4f(x, y)]\end{aligned}}$$

neighbor above us

neigh below us

myself *4

Laplacian Mask

Kernal: set of weights

4 connected neighbors
north, south, east, west
all share borders

eight connected neighbor

Some ppl dont like having -#s @ center of kernal so they inverse everything

Isotropic result for rotations in increments of 90°

0	1	0
1	−4	1
0	1	0

Isotropic result for rotations in increments of 45°

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a b
c d

FIGURE 3.39 (a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

Another Derivation

$$\frac{1}{9}*\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$ Unweighted Average Smoothing Filter

all neighbors gets added together and get divided by 9

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}$$ Retain Original

retain orig image

just like original image – blurred img = sharp edge image

orig img – unweighted avg smooth filter

$$\frac{1}{9}*\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}$$ Original – Average (negative of Laplacian Operator)

original minus the Average

"Edge detection"

In constant areas: 0 ← **Summation of coefficients in masks equals 0.**

Near edges: high values

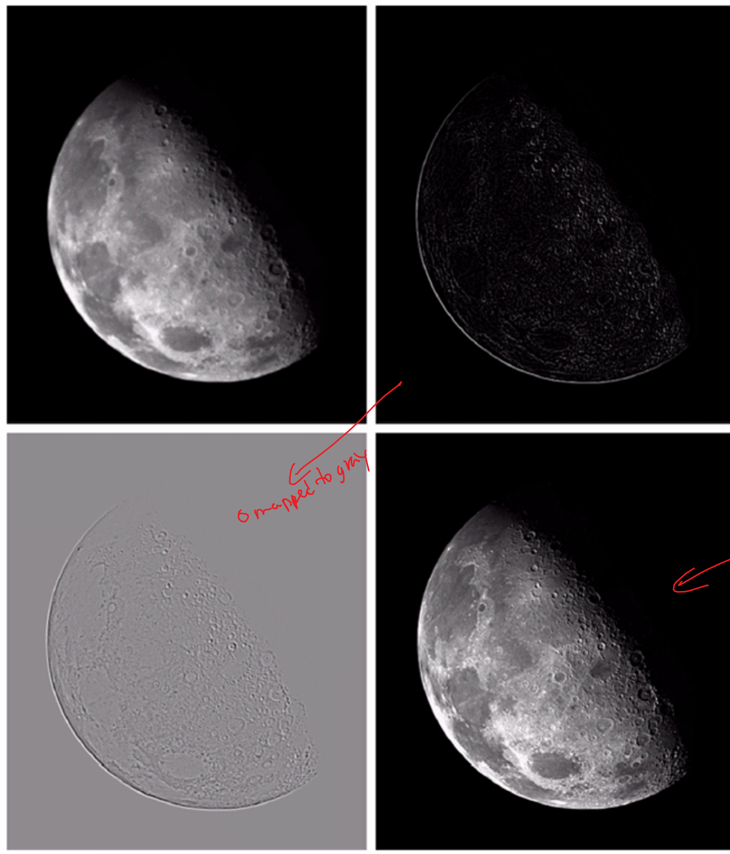
Effect of Laplacian Operator

- Since the Laplacian is a derivative operator
  - it highlights graylevel discontinuities in an image
  - it deemphasizes regions with slowly varying gray levels
- The Laplacian tends to produce images that have
  - grayish edge lines and other discontinuities all superimposed on a dark featureless background

Example

a b
c d

FIGURE 3.40 (a) Image of the North Pole of the moon. (b) Laplacian-filtered image. (c) Laplacian image scaled for display purposes. (d) Image enhanced by using Eq. (3.7-5). (Original image courtesy of NASA.)



0 mapped to gray

-1	-1	-1
-1	8	-1
-1	-1	-1

WOW

much better image!!

better edges!!!

Can do it in 1 step with kernel

Simplification

- Addition of image with Laplacian can be combined into one operator:

$$
\begin{aligned}
g(x,y) &= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\quad + f(x,y+1) + f(x,y-1) - 4f(x,y)] \\
&= 5f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\quad + f(x,y+1) + f(x,y-1)]
\end{aligned}
$$

If I just use this kernel on the moon there would be no need to get multiple images

0	-1	0
-1	5	-1
0	-1	0

=

0	0	0
0	1	0
0	0	0

original

+

0	-1	0
-1	4	-1
0	-1	0

4 connected

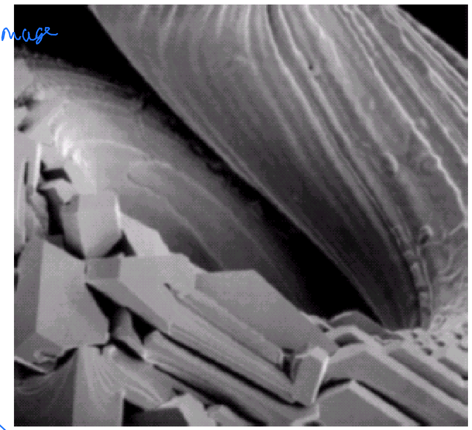
Example

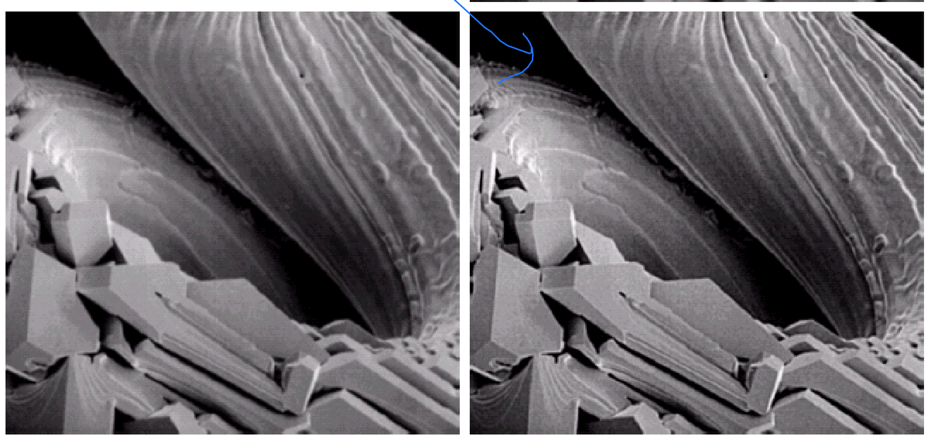
0	−1	0
−1	5	−1
0	−1	0

Laplacian is 8
but added orig image
which is 1 so
8+1=9

this kernel is
better and
sharper

−1	−1	−1
−1	9	−1
−1	−1	−1





a b c
d e

FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Gradient Operator (1)

- The gradient is a vector of directional derivatives.

$$\nabla \mathbf{f} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

keep the deriv of x and y in image in a vector

don't add it!

- Although not strictly correct, the magnitude of the gradient vector is referred to as the gradient.
- First derivatives are implemented using this magnitude

$$\begin{aligned} \nabla f &= \mathrm{mag}(\nabla \mathbf{f}) \\ &= [f_x^2 + f_y^2]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \end{aligned}$$

now can find sq root it to get strength of the direction

approximation:

$$\nabla f \approx |f_x| + |f_y|$$

Gradient Operator (2)

- The components of the gradient vector are linear operators, but the magnitude is not (square,square root).
- The partial derivatives are not rotation invariant (isotropic), but the magnitude is.
- The Laplacian operator yields a scalar: a single number indicating edge strength at point.
- The gradient is actually a vector from which we can compute edge magnitude and direction.

• 1 number the strength

• gives 2 things magnitude and angle:

$$f_{mag}(i,j) = \sqrt{f_x^2 + f_y^2} \quad \text{or} \quad f_{mag}(i,j) = |f_x| + |f_y|$$

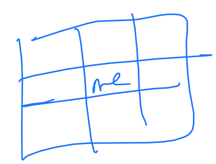
$$f_{angle}(i,j) = \tan^{-1}\frac{f_y}{f_x}$$

where

neighbors north south

$$\boxed{\begin{aligned} f_x(i,j) &= f(i+1,j) - f(i-1,j) \\ f_y(i,j) &= f(i,j+1) - f(i,j-1) \end{aligned}}$$

neighbors east west



Summary (1)

Continuous	Digital
$f(x)$	$v(i)$
$f'(x)$	$v'(i) = v(i) - v(i-1)$
$f''(x) = \nabla^2 f(x)$	$v''(i) = v'(i) - v'(i-1)$

$$
\begin{aligned}
v''(i) &= v'(i) - v'(i-1) \\
&= [v(i) - v(i-1)] - [v(i-1) - v(i-2)] \\
&= v(i-2) - 2v(i-1) + v(i) \\
&= (1 \quad -2 \quad 1)(v(i-2) \quad v(i-1) \quad v(i)) \\
&= (1 \quad -2 \quad 1)(v(i-1) \quad v(i) \quad v(i+1))
\end{aligned}
$$

centered on i-1

centered on one

	-1	
-1	4	-1
	-1	

-1	-1	-1
-1	8	-1
-1	-1	-1

The Laplacian is a scalar, giving only the magnitude about the change in pixel values at a point. The gradient gives both magnitude and direction.

0	0	0
100	100	100
100	100	100

horizontal edge

vertical edge

0	100	100
0	100	100
0	100	100

this is a vertical edge but I won't find it if I go vertically - will find if I go horizontally

2:57

Summary (2)

One dimensional:

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\text{mask}_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

(2) put results in the mask

Two dimensional:

Sobel Operator:

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{mask}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Prewitt Operator:

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{mask}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(1) use both the mask to an image to find edge

(3) find mag and angle

$$f_{mag}(i,j) = \sqrt{f_x^2 + f_y^2} \quad \text{or} \quad f_{mag}(i,j) = |f_x| + |f_y|$$

$$f_{angle}(i,j) = \tan^{-1}\frac{f_y}{f_x}$$

$$\boxed{\text{in}} \rightarrow \boxed{\text{Gradient}} \rightarrow \boxed{\text{Mag}},\ \boxed{\text{Angle}}$$

$$\boxed{\text{in}} \rightarrow \boxed{\text{Laplacian}} \rightarrow \boxed{\text{Mag}}$$

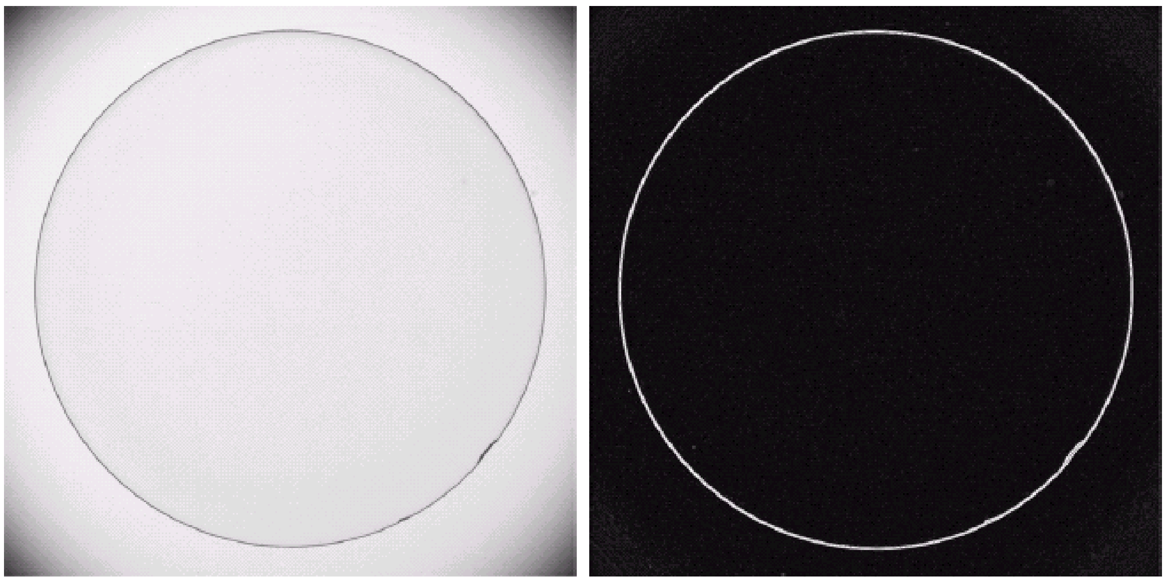
produce new images

Make new img

$$\boxed{\text{input image}} \rightarrow \boxed{f_x},\ \boxed{f_y} \rightarrow \boxed{\text{magnitude}},\ \boxed{\text{angle}}$$

← now you know the mag and angle of img

Example (1)



a b

FIGURE 3.45 Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

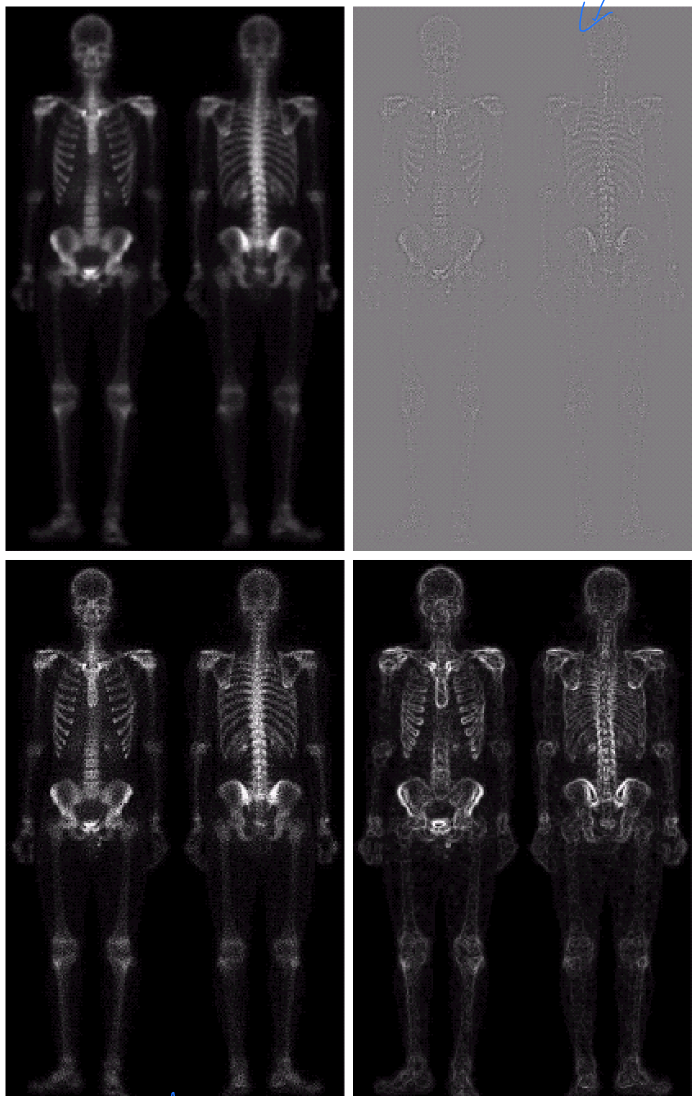
Example (2)

- **Goal:** sharpen image and bring out more skeletal detail.
- **Problem:** narrow dynamic range and high noise content makes the image difficult to enhance.
- **Solution:**

  1. Apply Laplacian operator to highlight fine detail
  2. Apply gradient operator to enhance prominent edges
  3. Apply graylevel transformation to increase dynamic range

orig

Laplacian of orig
(0 → gray)



a b
c d

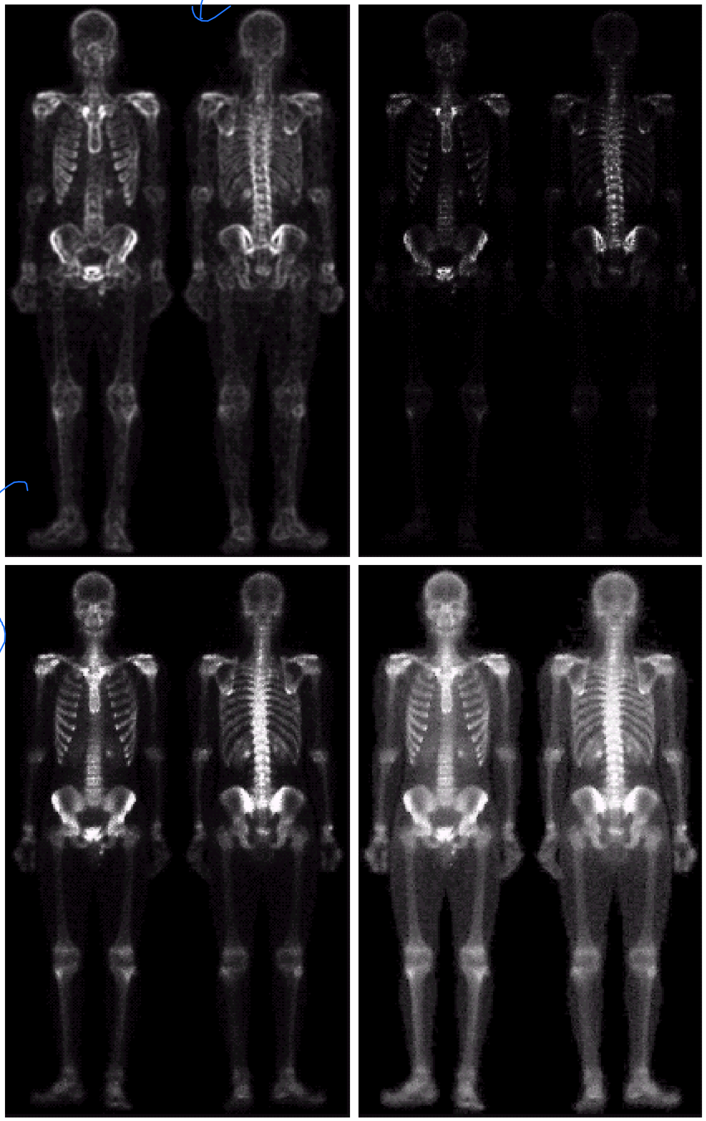
FIGURE 3.46
(a) Image of whole body bone scan.
(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).

Wo

Sharpeded img
orig + edge detect

another edge detection imag

Sobel + 5x5 pixel



Sharpened

e	f
g	h

FIGURE 3.46 *(Continued)* (e) Sobel image smoothed with a 5 × 5 averaging filter. (f) Mask image formed by the product of (c) and (e). (g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Wc