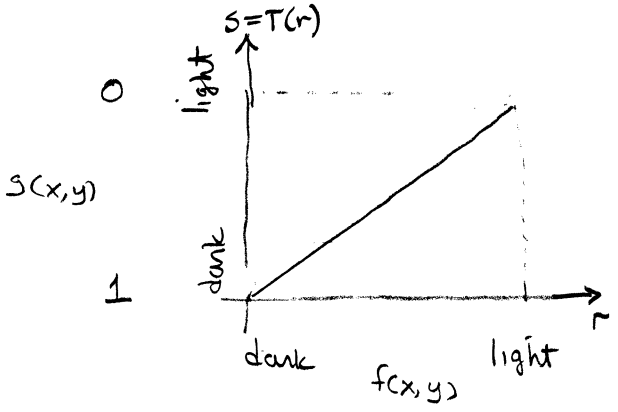
Spatial filtering

4.1.1. Spatial Domain Methods.

gray level transformation

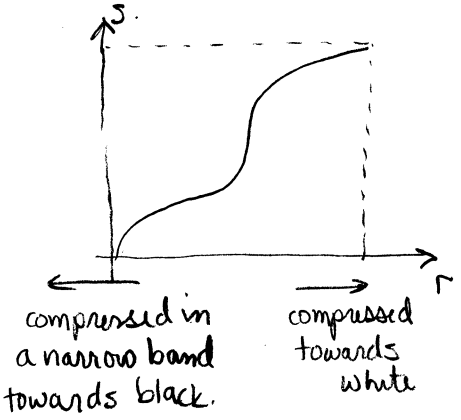
$$g(x,y) = T[f(x,y)]$$

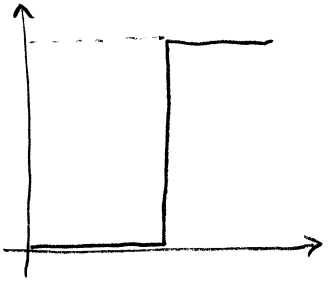
often point operations
i.e. 1×1



where r & s are variables denoting the gray level of f(x,y) & g(x,y) at (x,y)

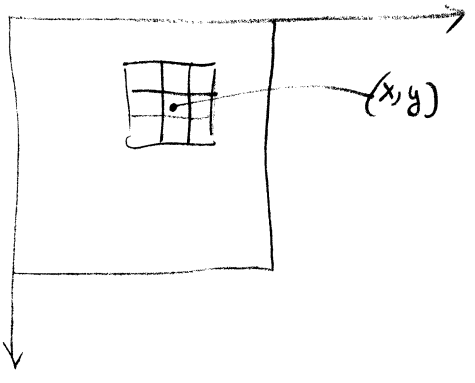
contrast stretching





extreme case (binary).

local neighborhoods.



3×3 neighborhood

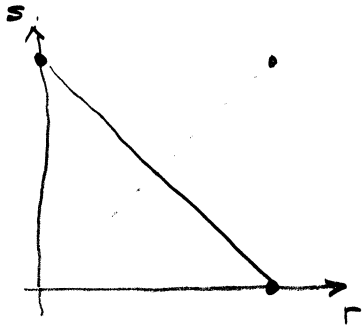
templates
windows
filters
} also called

4.1.2. Frequency Domain Methods (Wilson).

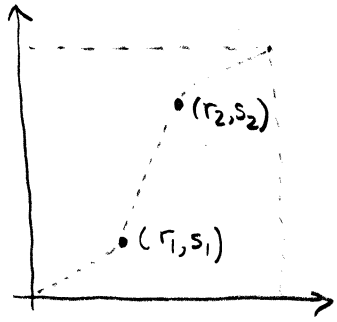
4.2 Enhancement by point processing

4.2.1. Some simple intensity transforms.

image negatives



contrast stretching

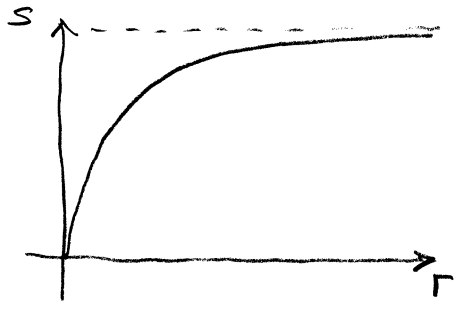


usually $r_1 \leq r_2$

$s_1 \leq s_2$

to prevent intensity artifacts

compression of dynamic range.

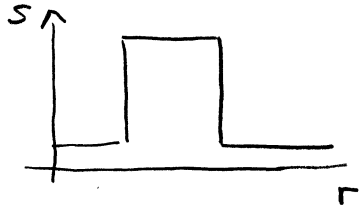


scaling constant

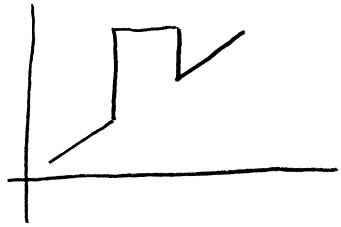
$s = c \log(1+r)$

brings up weak detail.

gray level slicing



highlights range of intensities reduces all others to a low background



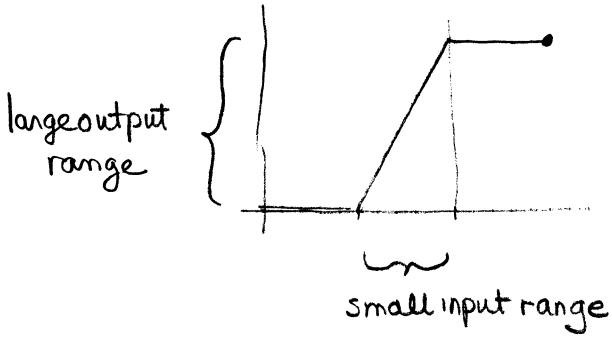
keeps intensities of background.

bit plane slicing

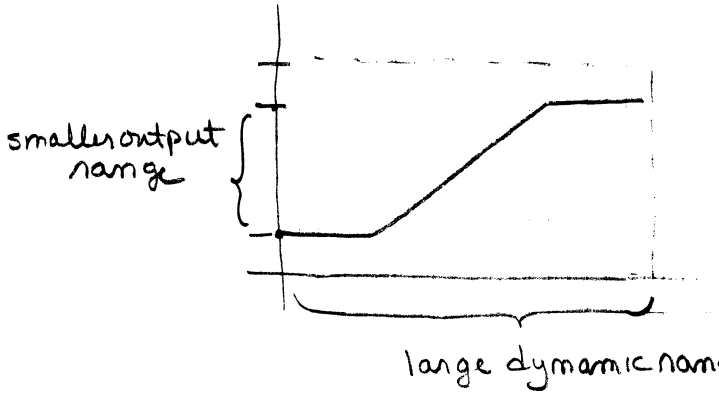
show planes which correspond to specific bits

Examples of image · compression/expansion

If you have a small gray scale.



expansion



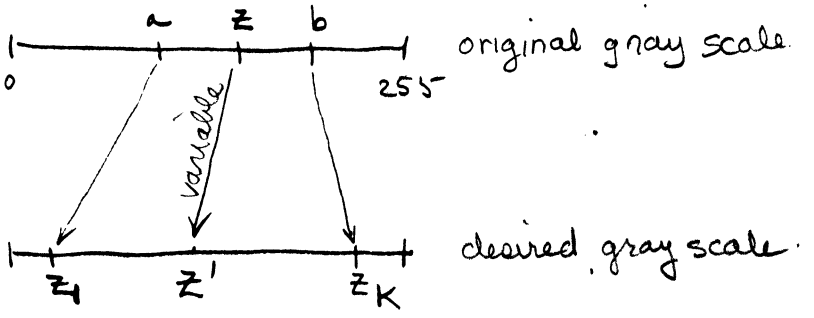
compression

often used for improving enhancing displays

Jain - Chapter 4 Image Filtering

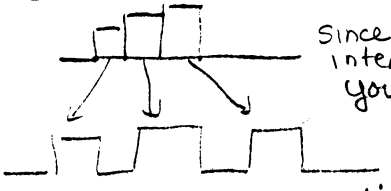
4.1 histogram equalization — improve contrast of an image by uniformly redistributing the gray values. (do before Thresholding) primarily good for viewing

most people don't do this



simple proportionality $z' = \left(\frac{z_K - z_1}{b - a}\right)(z - a) + z_1$

Problem: what was originally 3 digital bins remains three bins with gaps. This is due to intensity quantization.

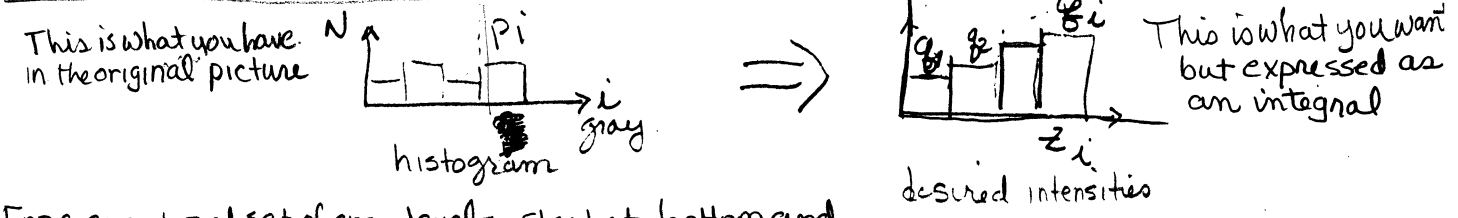


since images are digital in intensity you might get gaps if you expand too much

gaps between bins (values in histogram)

How about if you want a different shape

If you know the distribution you want a priori then use a cumulative function (This is to avoid quantization problems.)



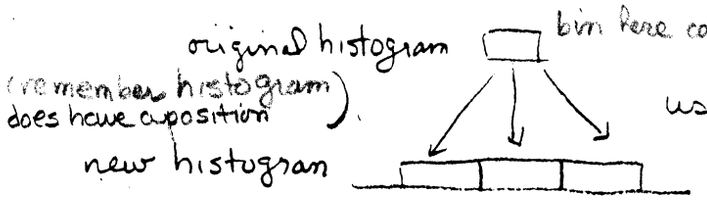
For a quantized set of gray levels start at bottom and then Do one level at a time

find k_1 in original image which bounds q_1 on each side $\Rightarrow \sum_{i=1}^{k_1-1} P_i \le q_1 < \sum_{i=1}^{k_1} P_i$

i.e. upper and lower bounds for the bins

then find k_2, the next level and repeat! $\Rightarrow \sum_{i=1}^{k_2-1} P_i \le q_2 < \sum_{i=1}^{k_2} P_i$

One problem may be how to split up bins if you are expanding an image,



i.e. n output bins

$q_k, q_{k+1}, \cdots, q_{k+n-1}$ edges of bins

random number $r \in [0, 1]$.

output bin is $k + [n * r]$.

$[\ ]$ greatest integer function

n # of bins

Histogram Equalization based upon idea of histogram as a probability

① histogram is viewed as a probability density function $p_r(r)$ and the p.d.f. of the transformed image is $p_s(s)$.

② histogram c.d.f. increase monotonically.

$$p_s(s) = \left[p_r(r) \frac{dr}{ds} \right]_{r = T^{-1}(s)} \quad (1)$$

transformed image — input histogram — this is because we start with what we want

Example 1: Suppose $s = T(r) = \int_0^r p_r(w)\,dw$ ← this is the cumulative distribution function

then $\frac{ds}{dr} = p_r(r)$ by taking derivatives

Evaluating (1) $p_s(s) = \left[p_r(r) \frac{1}{p_r(r)} \right]_{r = T^{-1}(s)}$.

$= [1]_{r = T^{-1}(s)}$

$= 1 \quad 0 \le s \le 1$

This gives a uniform density.

This is a weak proof. The idea is that if we use a transform which is the CDF of $p_r(w)$ we will have uniform distribution, i.e., they are equally populated! in the new image.

> simple proof

$\int p_s(s)\,ds = \int p_r(r)\,dr$ integral is simple sum of pixels.

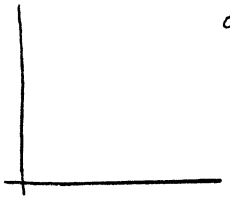
$\therefore$ map $p_s(s)\,ds = p_r(r)\,dr$

# of pixels here = # of pixels here.

subject to constraint on $\int p_s(s)\,ds$.

this mathematically relates input and output

discrete version is a histogram of probability density function



$$p_r(r_k) = \frac{n_k}{n}$$

n_k = # of pixels with level k.
n = # of pixels in image.
$0 \le r_k \le 1$
and $k = 0, 1, \ldots, L-1$

This is histogram equalization $\left\{ s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} \right\}$ a cumulative transform

This is a transform we can implement.

$$s_k = \sum_{j=0}^{k} p_r(r_j)$$

discrete version of $\int_0^r p_r(w)\,dw$

$$r_k = T^{-1}(s_k)$$

any pixel of value r_k will get transformed to s_k.

Example: weld

(1) convert histogram to probabilities, i.e. divide by $(512)^2$, number of pixels
normalize vertical axis to (0,1) by dividing by 255, number of gray levels (8 bits)

(2) see graph in text. Compute $s_k = \sum_{j=0}^{k} \frac{n_j}{n}$

It remaps components – it does not simply redistribute them!

Histogram specification (to get a specific form using pdfs is complex).

(1) Equalize image using $s = T(r) = \int_0^r p_r(w)\,dw$, gives uniform s

(2) specify a desired density function $p_z(z)$

(3) Equalize using this density function $v = G(z) = \int_0^z p_z(w)\,dw$

(4) invert transformation
$z = G^{-1}(s)$ } this is the problem, esp. when some levels are zero
If we invert this we get the z-levels which correspond to s. However, this often does not exist!

(5) apply to equalized image from (1)

Local enhancement

(1) define a neighborhood and move around. Use the histogram of the points locally and then transform. Use to map center pixel. (good for bringing out details).

Can do iteratively.

(2) Divide image into regions:

You can use other transformations

$$g(x,y) = A(x,y)\left[f(x,y) - m(x,y)\right] + m(x,y)$$

where $A(x,y) = k\frac{M}{\sigma(x,y)}$ local gain factor amplifies local variations i.e. if $\sigma(x,y)$ small, larger A if $\sigma(x,y)$ large, smaller A.

$m(x,y)$ – neighborhood mean

$\sigma(x,y)$ – " std. dev.

m – global mean

k – constant.

Image subtraction (still a point operation)

$$g(x,y) = f(x,y) - h(x,y)$$

basis for

- MPEG
- mask mode radiography

4.2.4 Image Averaging

if $g(x,y) = f(x,y) + \underbrace{\eta(x,y)}$

uncorrelated
zero mean noise

if you compute $\bar{g}(x,y) = \frac{1}{M} \sum_{i=1}^{M} g_i(x,y)$

i.e. average the input images. Not always possible.

then $E\{\bar{g}(x,y)\} = f(x,y)$

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

or $$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{M}} \sigma_{\eta(x,y)}.$$

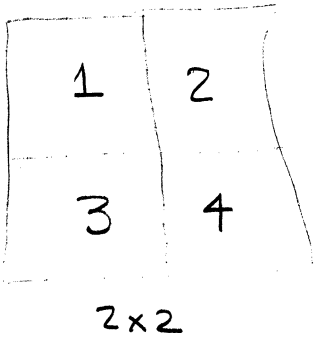
one problem is registering multiple images, also moving objects

Here is a little MATLAB program you can run to see histogram ~~equalization~~, stretching

```
load forest
I = ind2gray(X, map);      % forest is an indexed image
                           this converts it
J = imadjust(I, [0. 0.5], [], [])
```

- I: input image
- [0. 0.5]: range of values in input image
- []: bottom : top in output image. empty matrices mean use defaults [0 1], [1]
- []: value of γ uses transform $y = x^{\gamma}$. $\gamma = 1$ does linear stretching
- [1]: matrix for color.

```
subplot(2,2,1), imhist(I,128);
subplot(2,2,2), imshow(I,128);
subplot(2,2,3), imhist(J,128);
subplot(2,2,4), imshow(J,128);
```



2x2

for equalization use:

```
J = histeq(I, n)
```

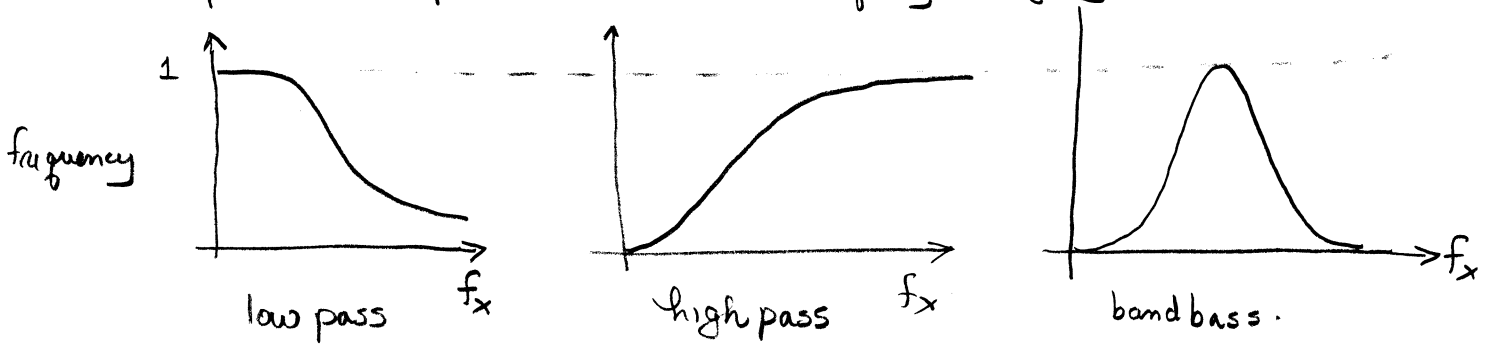
n: # of gray levels, default level: 64

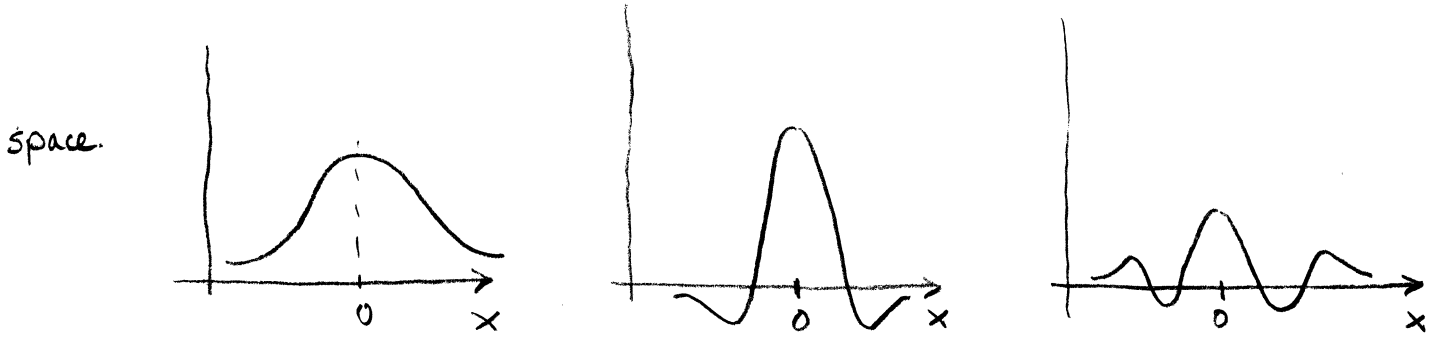
4.3 Spatial filtering

uses spatial masks rather than Frequency domain (using FFT).

- low pass – attenuate high frequencies → blurring
- high pass – attenuate low frequencies → sharpen edges & details
- band pass – image restoration, not used in image enhancement.

linear operations in space are related to their frequency analogs by Fourier transforms.

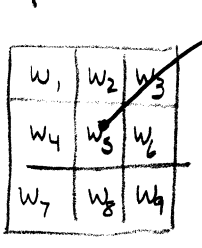




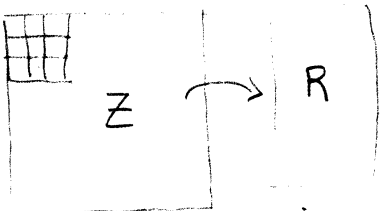
convolutional filtering input image Z, mask W, output R

$R(x,y) = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9$ for a 3×3 mask

Create a second image for R



$(x,y) \rightarrow R(x,y)$



we will defer discussion of what to do at edges

linear operations such as filtering

plus non-linear operations – noise reduction, etc.

- median (very useful)
- max
- min

4.3.2. Smoothing filters — blurring (preprocessing) and noise reduction

Simplest type of filter to visualize.

low pass spatial filtering (neighborhood averaging).

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

1	1	1
1	1	1
1	1	1

all positive coefficients

↑ scale back for display.

Gaussian filters are also VERY useful.

gaussian
centerweighted average.

4.3.3. Sharpening filters

—highlight fine detail. (basically derivative filters).

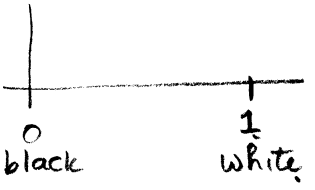
basic shape indicates positive coefficients near center
negative coefficients near periphery

classic 3x3 sharpening filter

$\frac{1}{9}$

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

eliminates zero frequency term.
significantly reducing contrast.



High boost filter (good edge enhancement).

Type 1: Highpass = Original − Lowpass ← unsharp masking subtract blurred from original

Type 2: High boost = A(original) − Lowpass

$= (A-1)(\text{original}) + (\text{original}) - \text{Lowpass}$

$= (A-1)\,\text{Original} + \text{Highpass}$

A=1 std. highpass
A>1 partially restores LF components lost.

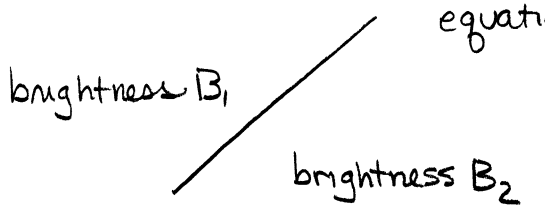
$\frac{1}{9} \times$

-1	-1	-1
-1	w	-1
-1	-1	-1

$w = 9A-1$ where $A \geq 1$

Differential Operators

① Let's model an edge and look at differential operators



equation of edge

$$x\sin\theta - y\cos\theta + \rho = 0$$

define $u(t) = \begin{cases} 1 & z > 0 \\ \frac{1}{2} & z = 0 \\ 0 & z < 0 \end{cases}$.

② The image brightness can be written as

$$E(x,y) = B_1 + (B_2 - B_1)\, u(x\sin\theta - y\cos\theta + \rho)$$

This is just gray scale.

③ Compute derivatives to get the gradient of the brightness function

$$\frac{\partial E}{\partial x} = +(B_2 - B_1)\sin\theta \underbrace{\delta(x\sin\theta - y\cos\theta + \rho)}_{\text{an impulse at the edge.}}$$

similarly $\frac{\partial E}{\partial y} = -(B_2 - B_1)\cos\theta\, \delta(x\sin\theta - y\cos\theta + \rho)$

brightness gradient (a vector) $= \begin{bmatrix} \frac{\partial E}{\partial x} \\ \frac{\partial E}{\partial y} \end{bmatrix}$

- coordinate system independent
- magnitude and orientation follow rotations and translations of edge in image.
- points in direction of largest increase

Squared gradient

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 = (B_2 - B_1)^2 \sin^2\theta\, \delta^2(x\sin\theta - y\cos\theta + \rho) + (B_2 - B_1)^2 \cos^2\theta\, \delta^2(x\sin\theta - y\cos\theta + \rho)$$

$$= (B_2 - B_1)^2\, \delta^2(x\sin\theta - y\cos\theta + \rho)$$

- non linear
- rotationally symmetric (finds all edges equally well)

We can also compute higher order derivatives

$$\frac{\partial^2 E}{\partial x^2} = (B_2 - B_1)\sin^2\theta\ \delta'(x\sin\theta - y\cos\theta + \rho)$$

(δ': doublet)

$$\frac{\partial^2 E}{\partial x \partial y} = -(B_2 - B_1)\sin\theta\cos\theta\ \delta'(x\sin\theta - y\cos\theta + \rho)$$

$$\frac{\partial^2 E}{\partial y^2} = (B_2 - B_1)\cos^2\theta\ \delta'(x\sin\theta - y\cos\theta + \rho)$$

One of the most useful edge operators is the Laplacian

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} = (B_2 - B_1)\,\delta'(x\sin\theta - y\cos\theta + \rho)$$

- rotationally symmetric

Quadratic Laplacian

$$\left(\frac{\partial^2 E}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 E}{\partial x^2}\right)\left(\frac{\partial^2 E}{\partial y^2}\right) + \left(\frac{\partial^2 E}{\partial y^2}\right)^2$$

$$= (B_2 - B_1)^2\,\delta'^2(x\sin\theta - y\cos\theta + \rho)$$

this is non-linear

Laplacian is the only derivative operator that is:

- linear
- maintains sign of brightness
  (allows you to reconstruct image from edge image)

Discrete edge operators

- We can't use continuous functions on images
  need discrete approximations to derivative operators

2x2 group of pixels

$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i,j}$	$E_{i+1,j}$

these are physically separated by ϵ
Typically $\epsilon = 1$ since we ignore physical dimensions in most pictures.

Approximate derivative by

$$\frac{\partial E}{\partial x} \approx \frac{E_{i+1,j} - E_{i,j}}{\epsilon}$$

Average for upper and lower

$$\overline{\frac{\partial E}{\partial x}} \cong \frac{1}{2}\left[\underbrace{\frac{E_{i+1,j+1} - E_{i,j+1}}{\epsilon}}_{①} + \underbrace{\frac{E_{i+1,j} - E_{i,j}}{\epsilon}}_{②}\right]$$

gradient templates:

$$\frac{1}{2\epsilon}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Similarly for $\frac{\partial E}{\partial y}$

$$\overline{\frac{\partial E}{\partial y}} \cong \frac{1}{2\epsilon}\left[\underbrace{\left(E_{i+1,j+1} - E_{i+1,j}\right)}_{③} + \underbrace{\left(E_{i,j+1} - E_{i,j}\right)}_{④}\right]$$

$$\frac{1}{2\epsilon}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Squared gradient is a lot of algebra (multiply out 4 terms ①, ②, ③, ④ to get 16 terms.

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{1}{4\epsilon^2}\Big[$$

squaring original terms:

$$E_{i+1,j+1}^2 - 2E_{i+1,j+1}E_{i,j+1} + E_{i,j+1}^2$$
$$E_{i+1,j}^2 - 2E_{i+1,j}E_{ij} + E_{ij}^2$$
$$E_{i+1,j+1}^2 - 2E_{i+1,j+1}E_{i+1,j} + E_{i+1,j}^2$$
$$E_{i,j+1}^2 - 2E_{i,j+1}E_{i,j} + E_{i,j}^2$$

now do cross-terms:

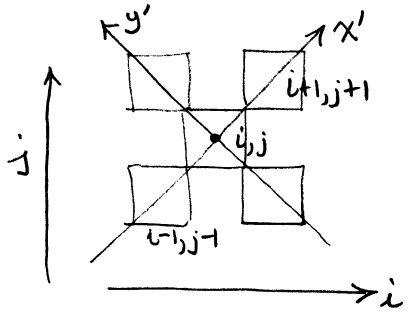
$$+2E_{i+1,j+1}E_{i+1,j} - 2E_{i+1,j+1}E_{i,j}$$
$$-2E_{i,j+1}E_{i+1,j} + 2E_{i,j+1}E_{i,j}$$
$$+2E_{i+1,j+1}E_{i,j+1} - 2E_{i+1,j+1}E_{i,j}$$
$$-2E_{i+1,j}E_{i,j+1} + 2E_{i+1,j}E_{i,j}$$

Collecting terms (lots cancel)

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{2E_{i+1,j+1}^2 - 4E_{i+1,j+1}E_{i,j} + 2E_{i,j}^2 + 2E_{i,j+1}^2 - 4E_{i,j+1}E_{i+1,j} + 2E_{i+1,j}^2}{4\epsilon^2}$$

Laplacian (cont.)

Consider a 45° rotated coordinate system



$$\frac{\partial^2 E}{\partial x'^2} = \frac{1}{2\epsilon^2}\left[E_{i+1,j+1} - 2E_{i,j} + E_{i-1,j-1}\right]$$

$$\frac{\partial^2 E}{\partial y'^2} = \frac{1}{2\epsilon^2}\left[E_{i-1,j+1} - 2E_{i,j} + E_{i+1,j-1}\right]$$

basic derivative

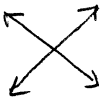
$$\frac{\dfrac{E_{i+1,j+1} - E_{i,j}}{\sqrt{2}\,\epsilon} - \dfrac{E_{i,j} - E_{i-1,j-1}}{\sqrt{2}\,\epsilon}}{\sqrt{2}\,\epsilon}$$

$$\frac{\partial^2 E}{\partial x'^2} + \frac{\partial^2 E}{\partial y'^2} \cong \frac{1}{2\epsilon^2}\left[E_{i+1,j+1} + E_{i-1,j-1} + E_{i-1,j+1} + E_{i+1,j-1} - 4E_{i,j}\right]$$

corresponding stencil is

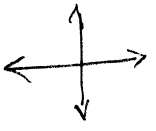
$\frac{1}{2\epsilon^2}$

1		1
	-4	
1		1

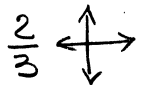

Laplacian

$\frac{1}{\epsilon^2}$

	1	
1	-4	1
	1	

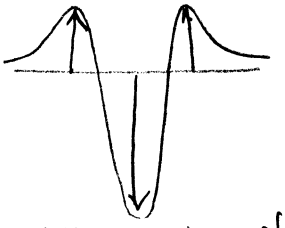

Laplacian

A popular approximation of the Laplacian is

$\frac{2}{3}$  $+ \frac{1}{3}$ 

$\frac{1}{6\epsilon^2}$

1	4	1
4	-20	4
1	4	1


approximation of continuous function

We can approximate other functions this way.

5.1 Gradient

gradient is the two dimensional approx. to the first derivative.

$$\underline{G}\,[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

vector $\underline{G}$ points in direction of maximum rate of increase of $f(x,y)$

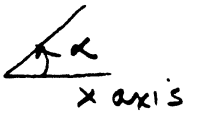
magnitude = max. rate of increase in direction of $\underline{G}$

$$\left|G\,[f(x,y)]\right| = \sqrt{G_x^2 + G_y^2}$$

more common approximation is

$$G\,[f(x,y)] \cong |G_x| + |G_y| \quad \text{faster}$$

$$\text{or } G[f(x,y)] \cong \max\,[|G_x|, |G_y|].$$

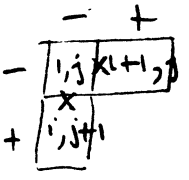
direction of $\underline{G}$ $\quad \alpha(x,y) = \text{Tan}^{-1}\left(\frac{G_y}{G_x}\right)$. 

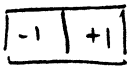
magnitude $\neq f(\alpha) \Rightarrow$ isotropic operator. which is good

Numerical approx:

$$G_x \cong f[i,j+1] - f[i,j].$$

$$G_y \cong f[i+1,j] - f[i,j].$$



convolution masks. 

G_y.

BAD. actually gradient G_x at $[i, j+\frac{1}{2}]$ and G_y at $[i+\frac{1}{2}, j]$.

BETTER. $\quad G_x = \begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix}$. $\quad G_y = \begin{bmatrix} -1 & -1 \\ +1 & +1 \end{bmatrix}$.

interpolated point is $[i+\frac{1}{2}, j+\frac{1}{2}]$ for both.

better yet to use 3x3 operators.

utility of edge information

- boundaries of objects tend to show up as intensity discontinuities in images
- often an edge can be recognized from only a crude outline
- boundary representation is easy to integrate into object recognition algorithms

edge operator detects the presence of local edges.

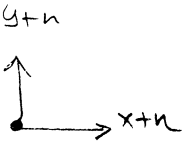
- most edge operators compute a <u>direction</u> aligned with direction of maximum grey-level change and a <u>magnitude</u> indicating extent of change
- sensitive to high frequency noise
- types of edge operators
  - (1) approximations of gradient operator
  - (2) template matching at different orientations
  - (3) curve fitting intensity information to edge models.

→ Laplacian has fallen into disuse

earliest edge operator — Robert's operator. (gradient)

$$s(\underline{x}) = \sqrt{\Delta_1^2 + \Delta_2^2}$$

$$\phi(\underline{x}) = \tan^{-1}\left(\frac{\Delta_2}{\Delta_1}\right)$$



where $\Delta_1 = f(x+n, y) - f(x, y)$

$\Delta_2 = f(x, y+n) - f(x, y)$

was very sensitive to noise

use local averaging to reduce noise effects.

Prewitt operator

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

Sobel operator

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

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

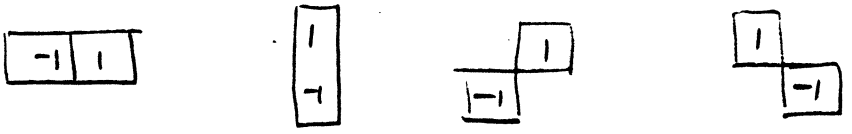
very optimal system!

center weighted.

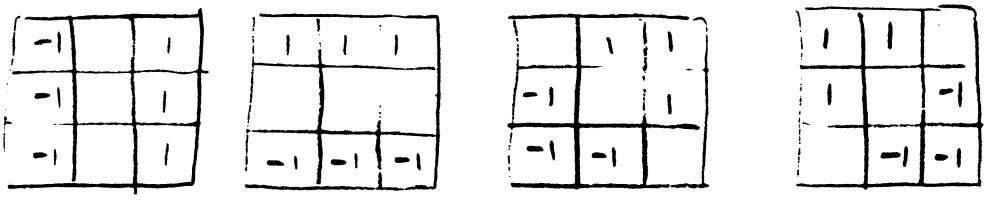
edge-templates (just really correlation)

Kirsch templates

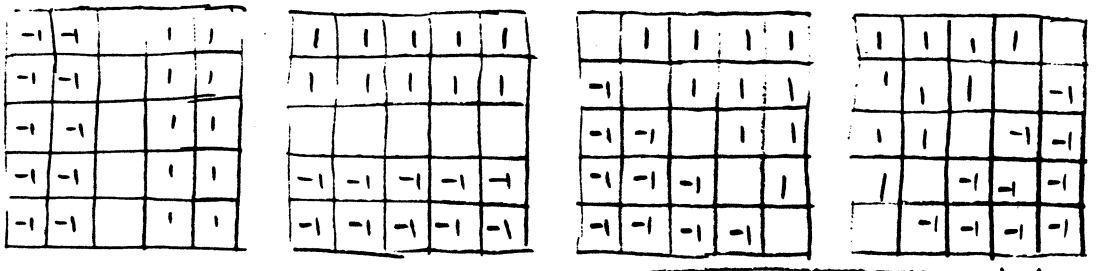
$n = \frac{1}{2}$



$n = 1$



$n = 2$



} there are $f(\underline{x}_k)$

Kirsch operator (related to edge gradient)

$$S(\underline{x}) = \max\left[1, \max_{k=0}^{n-1} |f(\underline{x}_k) - f(\underline{x})|\right]$$

(n = # adjacent pixels)

this part locates maximum change the k gives direction

this part compares it with a Threshold.

0	1	2
7	x	3
6	5	4

used this to give direction

Kirsch operator has been replaced by above templates.

HUNCH: human vision uses low-level template matching edge operators.

where is an edge ? gradient is rarely equal to zero
threshold gradient: edge at $\underline{x}$ iff $\nabla f(\underline{x}) > T$
does this work? not if edge is weak.

edge activity

Frei-Chen – edge operators are actually orthogonal basis function of edgeness, Represent local activity as a "Fourier"-like series

basis functions $h_k(o)$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad k=0$$

$$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \quad k=1$$

$$\begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix} \quad k=3$$

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad k=5$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad k=7$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \quad k=2$$

$$\begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix} \quad k=4$$

$$\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} \quad k=6$$

$$\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \quad k=8$$

write image in neighborhood of $\underline{x}$ as

basically write as a Frei-Chen series

$$f(\underline{x}_0) = \sum \frac{\langle f, h_k \rangle}{\langle h_k, h_k \rangle} h_k(\underline{x} - \underline{x}_0)$$

↖ dot products or correlations

how much edgeness is there?

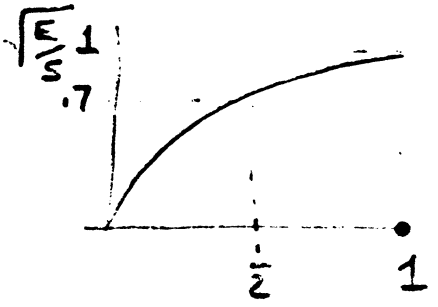
total energy $S = \sum_{k=0}^{8} \langle f, h_k \rangle^2$ these are just all the coefficients — total energy in neighborhood

edge energy $E = \sum_{k=1}^{2} \langle f, h_k \rangle^2$ these are just the horizontal and vertical edges — vertical and horizontal edge energy.

removes sign and does edge compression. $\theta = \cos^{-1}\left(\sqrt{\frac{E}{S}}\right)$

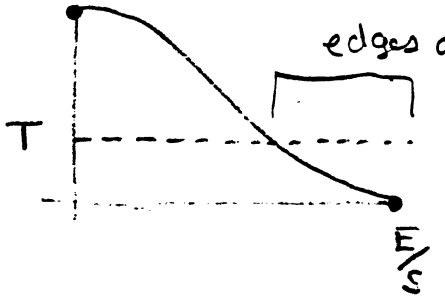
$\sqrt{\ }$ so that we are comparing energy

if $\theta < T \Rightarrow$ an edge, otherwise not.

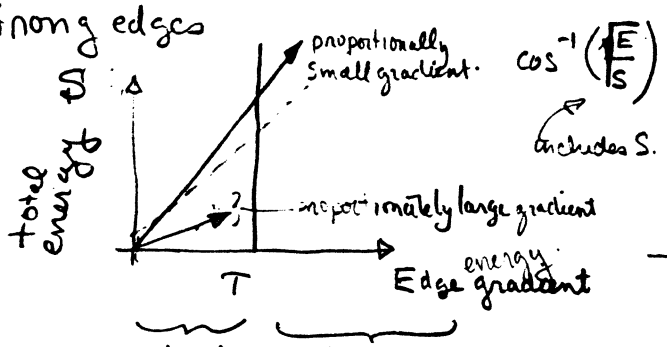


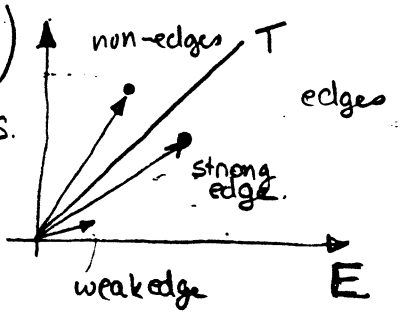
compression limits large "edgeness"

$\cos^{-1}\left(\sqrt{\frac{E}{S}}\right)$



Frei-Chen method normalizes to produce uniform sensitivity to weak and strong edges

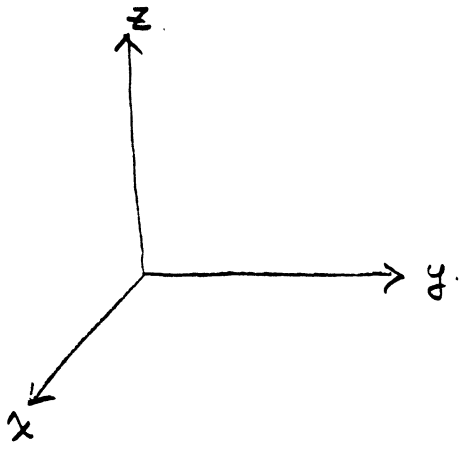


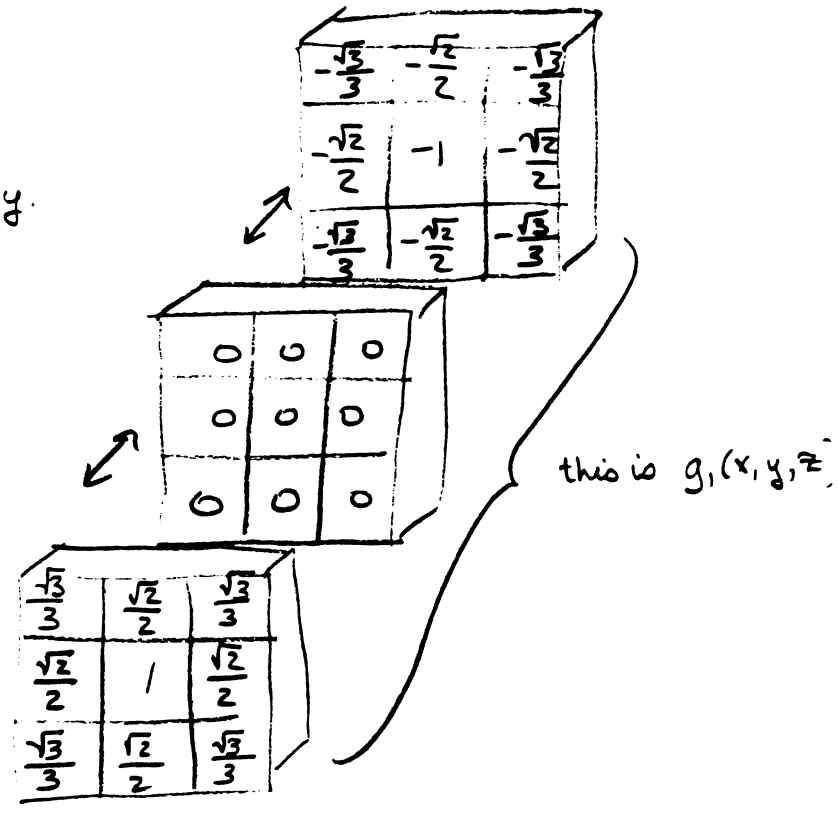


normalizes "edgeness"

a large fraction of edge energy identifies edges!

three-dimensional edge operators (Zucker-Hummel)





x directed basis function. $g_1(x,y,z)$

y-directed $g_2(x,y,z)$

z-directed $g_3(x,y,z)$

this is x directed.

$\underline{n} = (a,b,c)$.

compute $a = \langle g_1, f(x) \rangle$

$b = \langle g_2, f(x) \rangle$

$c = \langle g_3, f(x) \rangle$

surface normal $\underline{n} = (a,b,c)$

surface element detected if $|n| > T$, some threshold.

how good is an edge operator?

Pratt's figure of merit

$$F = \frac{1}{\max(N_A, N_I)} \sum_{i=1}^{N_A} \frac{1}{1 + (a d_i^2)}$$

penalty; difference between actual & ideal

normalizes to 1

in other words # of edges

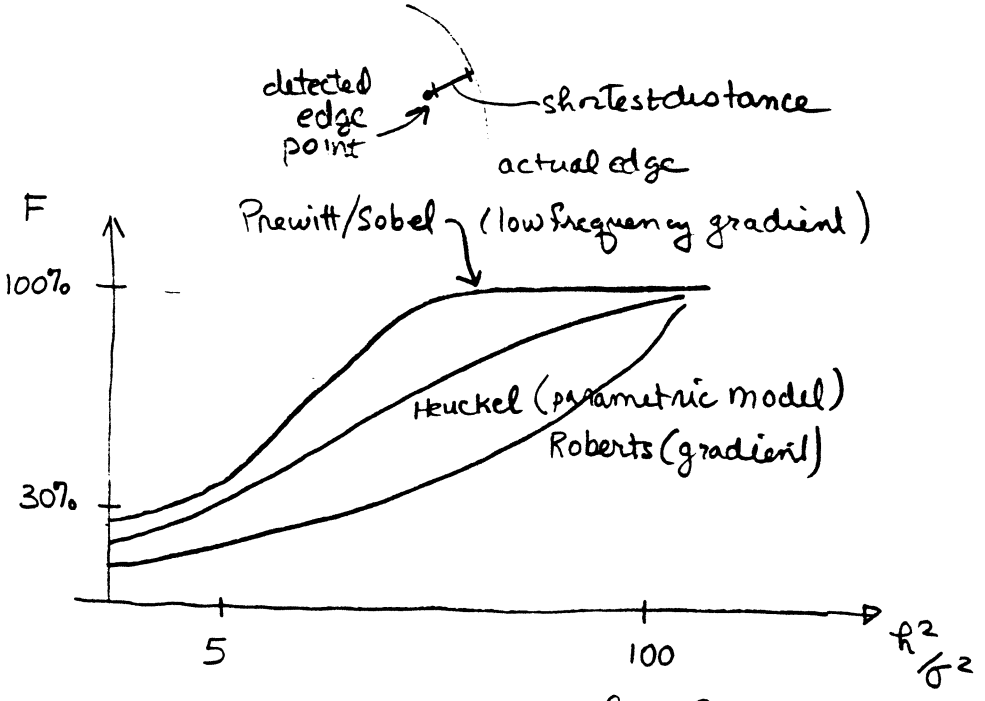
penalty for displaced edge points

N_A = detected edge points

N_I = actual edge points

a = scaling constant

d = signed separation distance of an actual edge point normal to a line of ideal edge points.



high $\frac{S}{N}$ → h = step edge amplitude

σ = standard deviation of Gaussian white noise

MATLAB filter types

h = fspecial ('type', params)

'gaussian', n, sigma — $n = n \times n$; sigma is mainlobe width σ in pixels.

'sobel' — gives 3×3 for vertical derivative $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
h' gives 3×3 for horiz. derivative
i.e. h' = fspecial ('sobel')

'laplacian', alpha — $0 \le \alpha \le 1$ controls shape.

$$\nabla^2 \approx \frac{4}{\alpha+1} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

'log', n, sigma — n×n laplacian of Gaussian with Gaussian mainlobe σ pixels

'prewitt' — $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$ to rotate use h'

'averaging', n — returns n×n averaging filter
$\frac{\text{ones}(n,n)}{n^2}$?

'unsharp', alpha — $\frac{1}{\alpha+1} \begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$

Example

```
load trees
I = ind2gray (X, map);
h = fspecial ('sobel');
A = filter2 (h, I);
```

I: input image
h: 2 dimensional mask.

5.3 Second Derivative Operators.

Laplacian

second directional derivative

use 2nd derivatives because they let us find <u>local</u> maxima in gradient values, i.e. find zero crossings of the 2nd derivative of edge intensity.

5.3.1. Laplacian

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

approximate with difference equations

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

$$= \frac{\partial}{\partial x} f[i,j+1] - \frac{\partial f}{\partial x}[i,j].$$

$$= \left(f[i,j+2] - f[i,j+1]\right) - \left(f[i,j+1] - f[i,j]\right)$$

$$= f[i,j+2] - 2f[i,j+1] + f[i,j].$$

this is centered at $[i, j+1]$

replace j by $j-1$ to center about $[i,j]$.

$$\frac{\partial^2 f}{\partial x^2} = f[i,j+1] - 2f[i,j] + f[i,j-1].$$

$$\frac{\partial^2 f}{\partial y^2} = f[i+1,j] - 2f[i,j] + f[i-1,j]$$

combining, we get the following mask.

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

If we want to weigh center more

$$\nabla^2 \approx \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}.$$

Example of row:

	1	2	3	4	5.			
result of Laplacian on image w/ simple step	0	0	0	6	-6	0	0	0

↖ this should be edge location has to be interpolated.

result of Laplacian To ramp edge.	0	0	0	3	0	-3	0	0

↑ this is ideal and exactly on pixel. again, usually need to interpolate

5.3.2. Second Directional Derivative

and derivative computed in direction of gradient

$$\frac{\partial^2}{\partial u^2} = \frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}}{f_x^2 + f_y^2}.$$

neither this or Laplacian frequently used due to their being severely affected by noise more than first derivative.

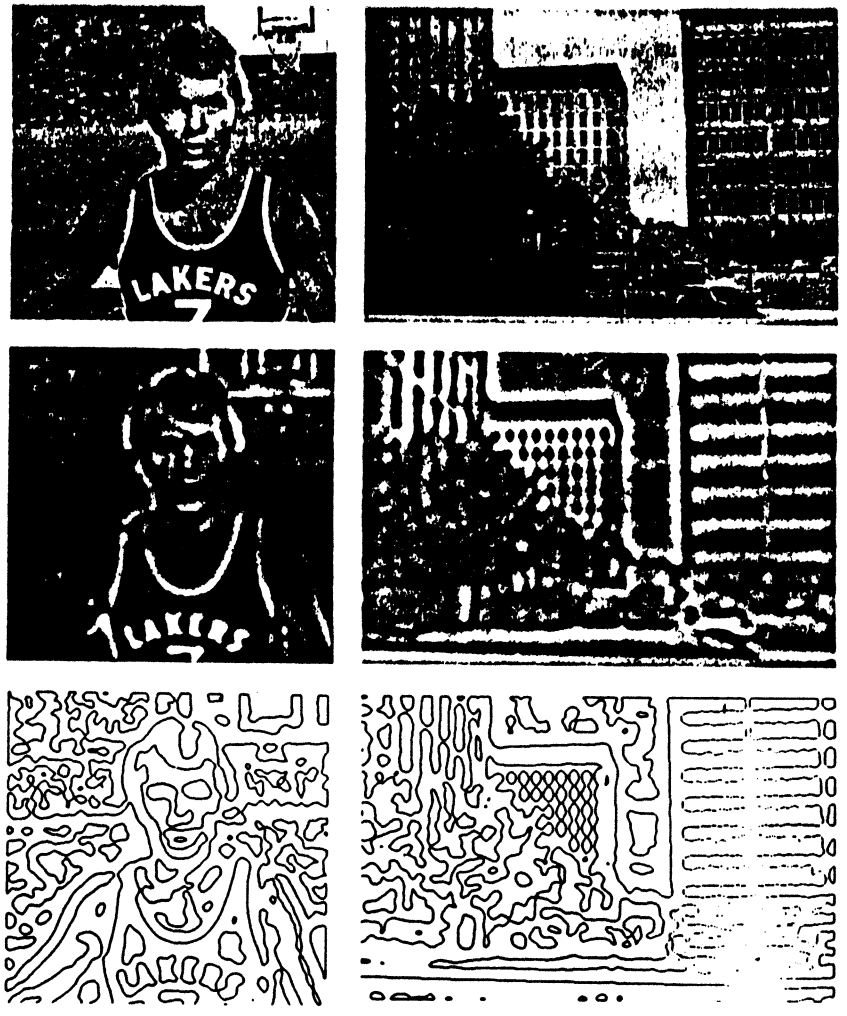


Figure 2.7
Examples of $\nabla^2 G$ Convolution and Zero-Crossings, Coarse Filter. Sample images are shown in (*a*). The convolutions of the images with a coarse $\nabla^2 G$ operator are shown in (*b*). In (*c*) the zero-crossings of (*b*) are illustrated.

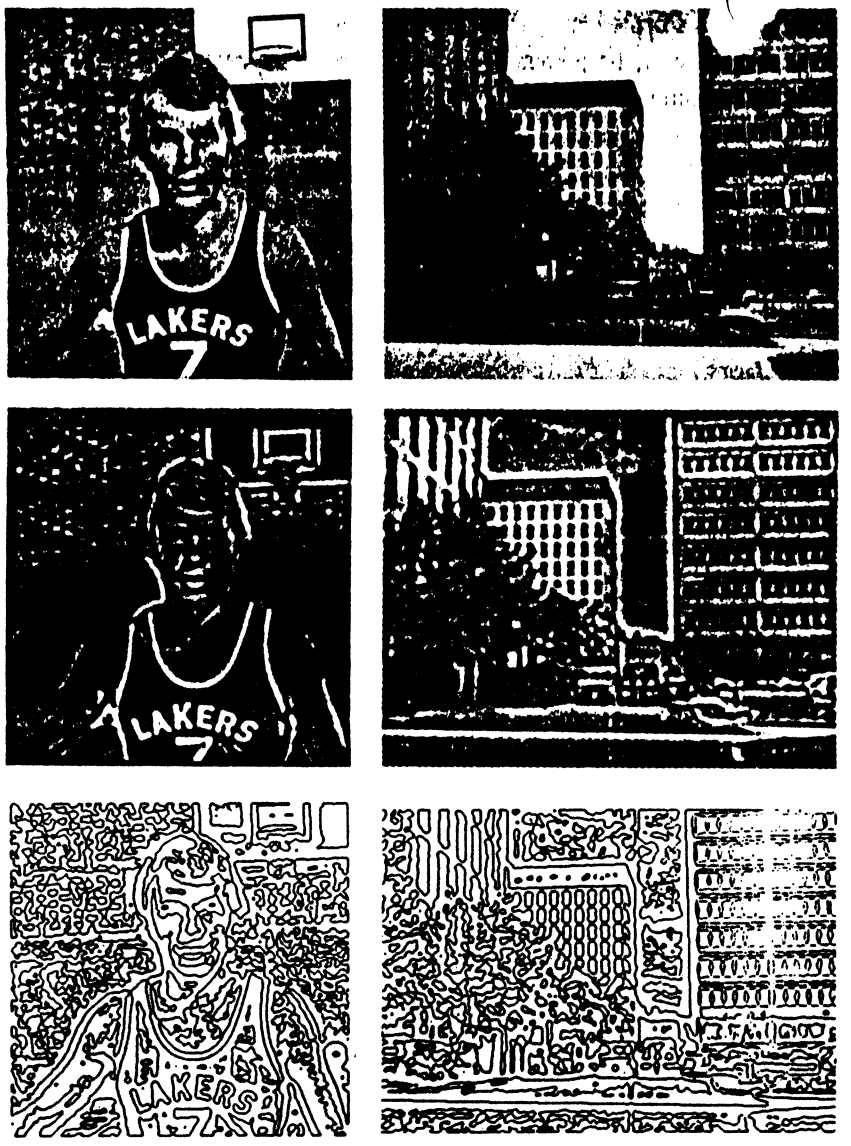


Figure 2.6
Examples of $\nabla^2 G$ Convolution and Zero-Crossings, Fine Filter. Sample images are shown in (*a*). The convolution of the images with a fine $\nabla^2 G$ operator are shown in (*b*). In (*c*) the zero-crossings of (*b*) are illustrated.

things remaining to cover

1. feature extraction
2. matched filtering and template matching