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.

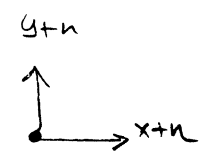
- most edge operators compute a <u>direction</u> aligned with direction of maximum grey-level change and a <u>magnitude</u> indicating extent of change
- sensitive to high frequency noise
- types of edge operators
  - (1) approximations of gradient operator
  - (2) template matching at different orientations
  - (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 $\Delta_1 = f(x+n, y) - f(x,y)$

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

<u>was very sensitive to noise</u>

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

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

very optimal system!

center weighted.

edge - templates (just really correlation)

Kirsch templates

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

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} & 1 \\ -1 & \end{bmatrix} \qquad \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}$$

$n = 1$

$$\begin{bmatrix} -1 & & 1 \\ -1 & & 1 \\ -1 & & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ & & \\ -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} & 1 & 1 \\ -1 & & 1 \\ -1 & -1 & \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & \\ 1 & & -1 \\ & -1 & -1 \end{bmatrix}$$

$n = 2$

$$\begin{bmatrix} -1 & -1 & & 1 & 1 \\ -1 & -1 & & 1 & 1 \\ -1 & -1 & & 1 & 1 \\ -1 & -1 & & 1 & 1 \\ -1 & -1 & & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ & & & & \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} & 1 & 1 & 1 & 1 \\ -1 & & 1 & 1 & 1 \\ -1 & -1 & & 1 & 1 \\ -1 & -1 & -1 & & 1 \\ -1 & -1 & -1 & -1 & \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 & 1 & \\ 1 & 1 & 1 & & -1 \\ 1 & 1 & & -1 & -1 \\ 1 & & -1 & -1 & -1 \\ & -1 & -1 & -1 & -1 \end{bmatrix}$$

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

Kirsch operator (related to edge gradient)

$$S(\underline{x}) = \max \left[1, \underbrace{\underline{\underline{\max}} \sum_{k=0}^{\overset{\#\text{ of adjacent pixels}}{n-1}} \left| f(\underline{x}_k) - f(\underline{x}) \right|} \right]$$

this part locates maximum change
the k gives direction

this part compares it with a threshold

0	1	2
7	$\underline{x}$	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 $h_k(o)$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}_{k=0} \quad \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}_{k=1} \quad \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}_{k=3} \quad \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}_{k=5}$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}_{k=7} \quad \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}_{k=2} \quad \begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}_{k=4} \quad \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}_{k=6}$$

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

basically write as a Frei-Chen series

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

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

dot products or correlations

how much edgeness is there?

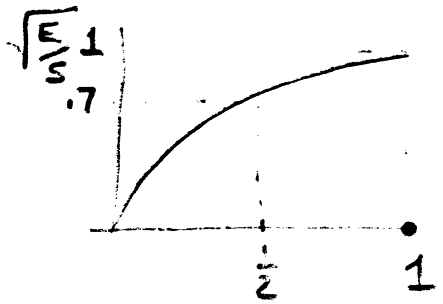
total energy $S = \sum_{k=0}^{8} \langle f, h_k \rangle^2$ these are just all the coefficients — total energy in neighborhood

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

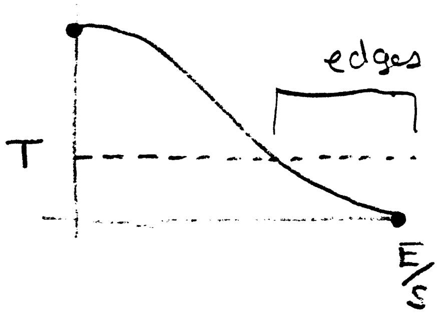
$\sqrt{\ }$ so that we are comparing energy

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

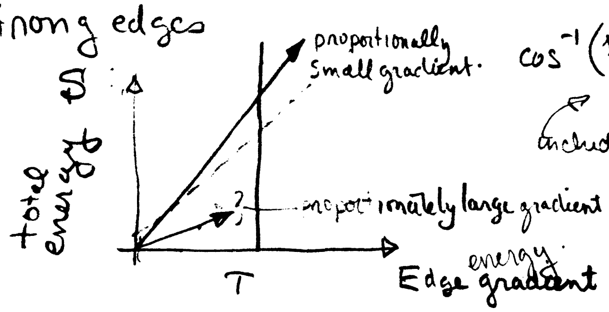


compression: limits large "edgeness"

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

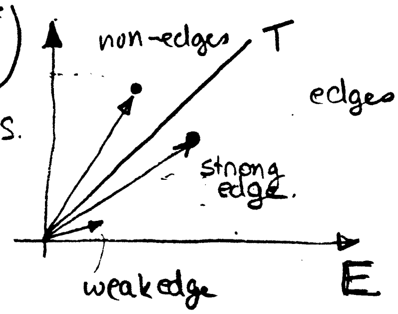


Frei-Chen method normalizes to produce uniform sensitivity to weak <u>and</u> strong edges



weak edges are lost — only strong edges survive

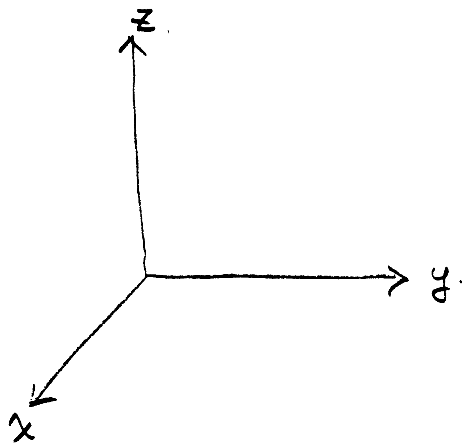
$\cos^{-1}\left(\sqrt{\frac{E}{S}}\right)$ includes S.

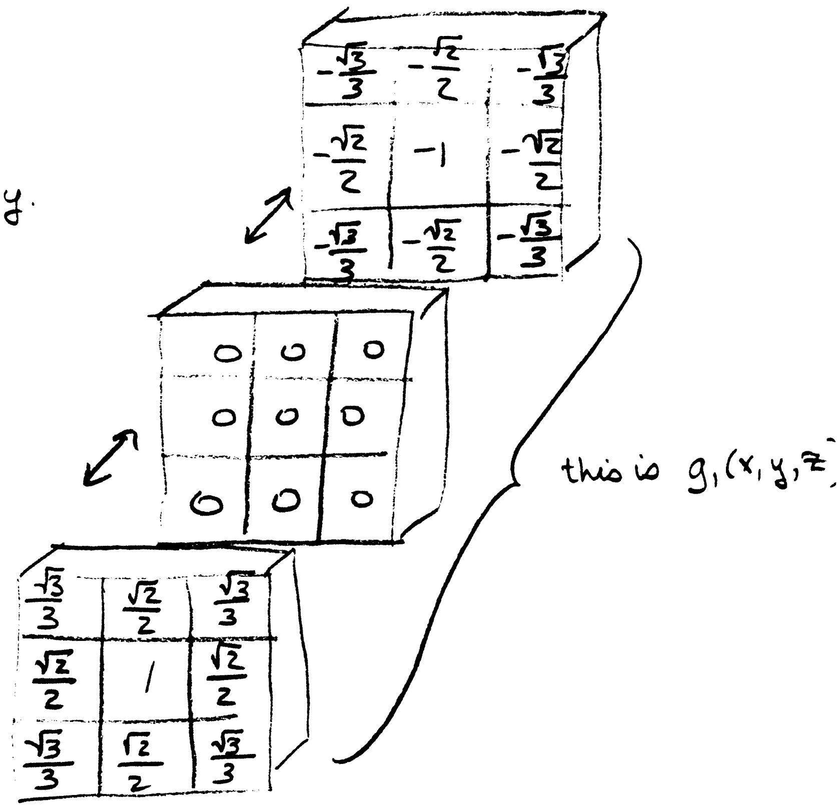


normalizes "edgeness"

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

penalty

difference between actual & ideal

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

normalizes to 1

in other words # of edges

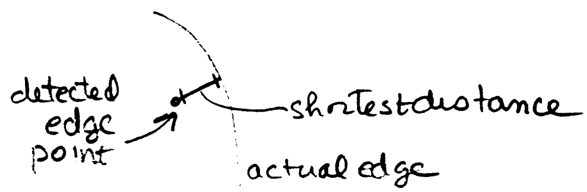
penalty for displaced edge points

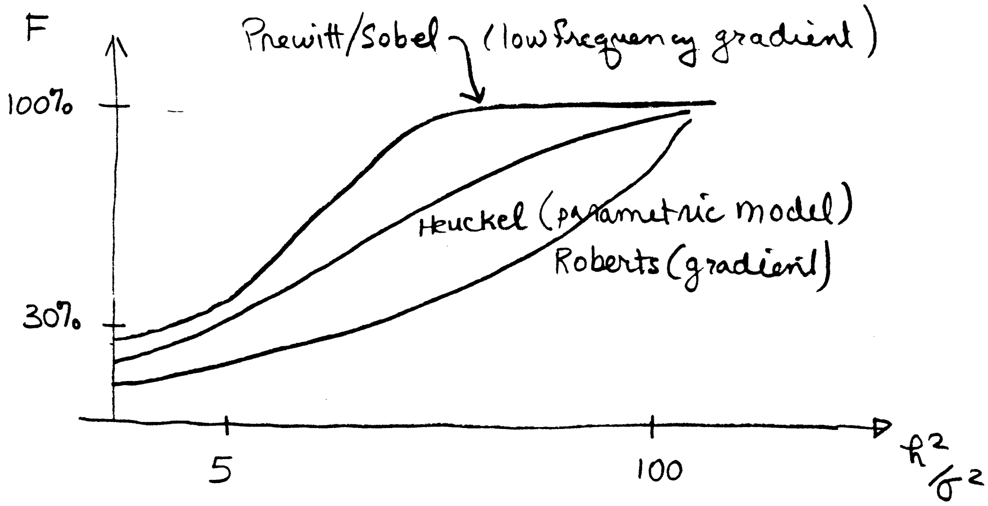
N_A = detected edge points

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

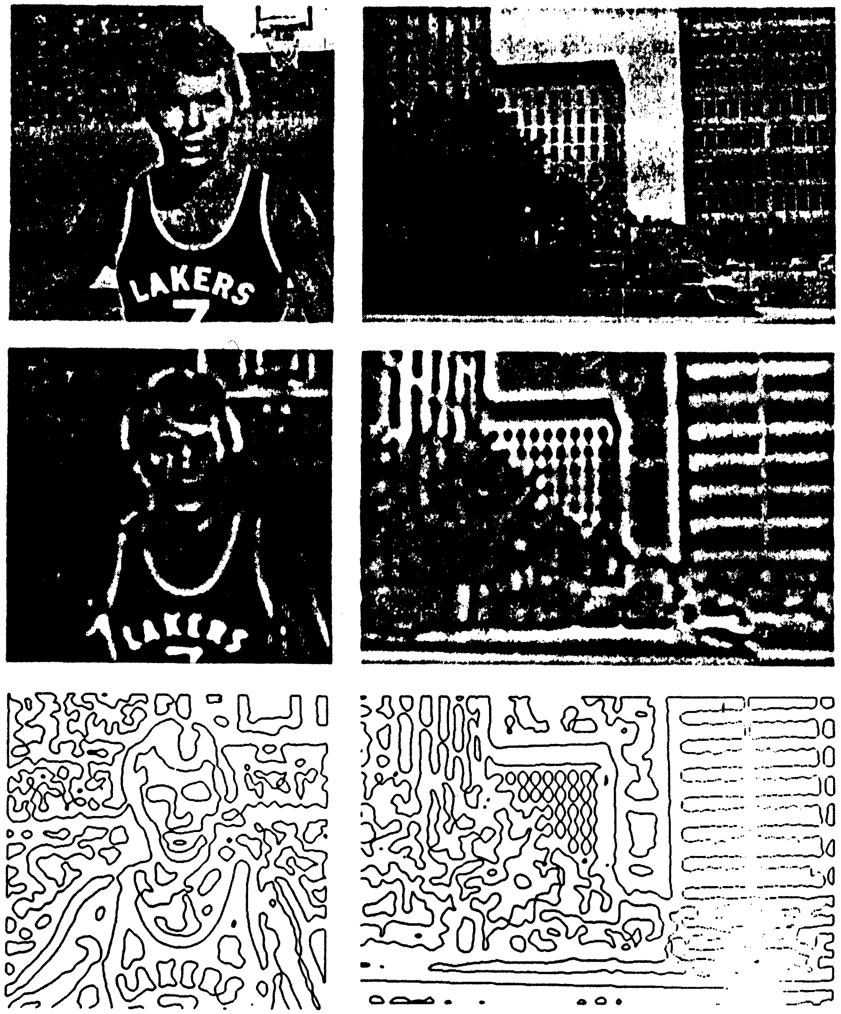


Figure 2.7
Examples of $\nabla^2 G$ Convolution and Zero-Crossings, Coarse Filter. Sample images are shown in (*a*). The convolutions of the images with a coarse $\nabla^2 G$ operator are shown in (*b*). In (*c*) the zero-crossings of (*b*) are illustrated.

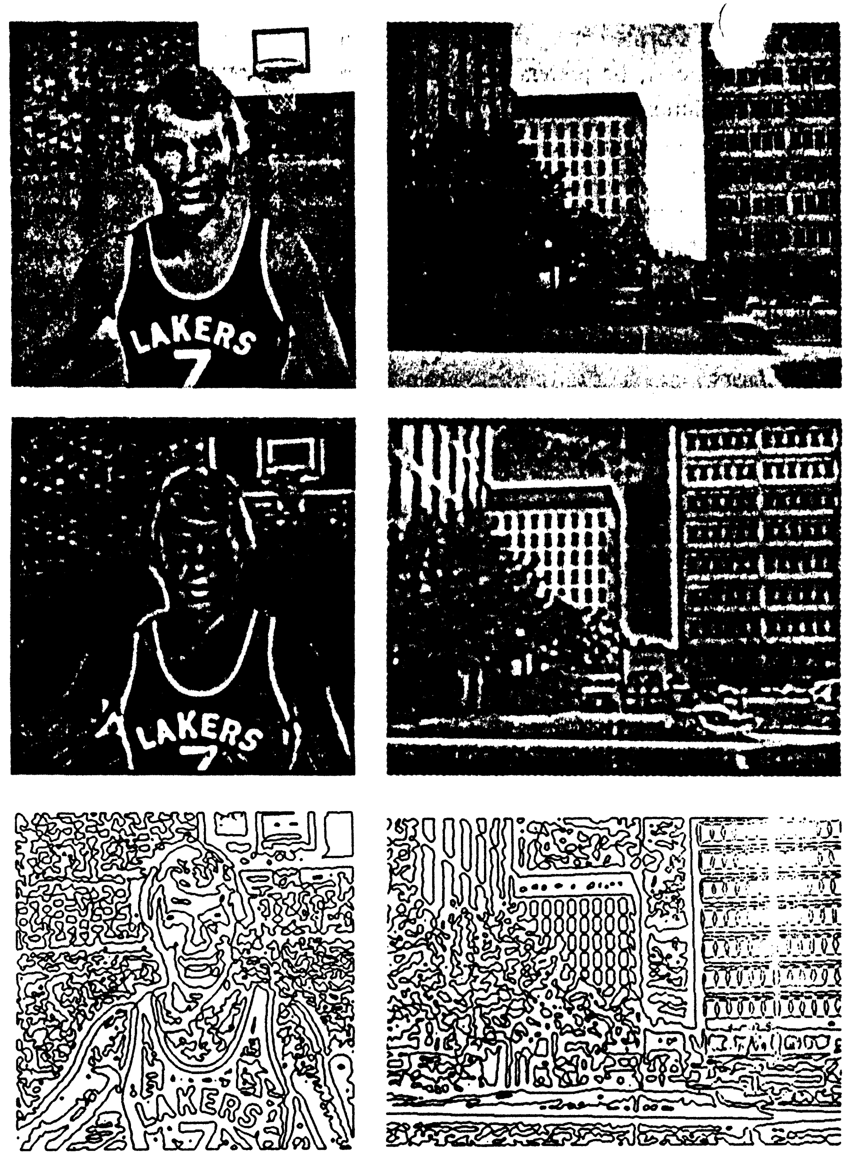


Figure 2.6
Examples of $\nabla^2 G$ Convolution and Zero-Crossings, Fine Filter. Sample images are shown in (*a*). The convolution of the images with a fine $\nabla^2 G$ operator are shown in (*b*). In (*c*) the zero-crossings of (*b*) are illustrated.

Chapter 5 - Edge Detection

edge - significant local change in the image, i.e. a point operator

used to eliminate noise

usually image intensity or first derivative of edge intensity

(Chap. 5 only covers detection & localization).

discontinuities can be step
line

both rare in real images

ramp

roof or

edges can have both line & step characteristics

Definitions

1. edge point — point $p[i,j]$ at the location of a significant local intensity change in the image
   can be integer, or sub-pixel
   usually in coordinate frame of image.
2. edge fragment — (i, j, θ) edge point coordinates + edge orientation
   "small" line segment.
3. edge detector — algorithm that produces edge points or fragments from an image → generates correct and false edges (false +) + edges that were missed (false −).
   (uses local info).
4. contour — list of edges or the mathematical curve that models the list of edges.
5. edge linking - process of forming an ordered list of edges from an unordered list. Convention, order in clockwise direction
6. edge following — searching a (filtered) image to find contours.
   (uses global info).

5.1 Gradient

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

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

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

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

more common approximation is

$$G\left[f(x,y)\right] \cong |G_x| + |G_y|$$

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

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

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

Numerical approx:

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

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

	−	+
−	i,j	i+1,j
+	i,j+1	

convolution masks.

G_x:

-1	+1

G_y:

-1
+1

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.

5.2.3 Prewitt

Same operator as Sobel but $c=1$.

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}. \qquad S_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}.$$

no emphasis on pixels closer to center of mask.

5.2.4 Comparison

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 $[i,j+1]$

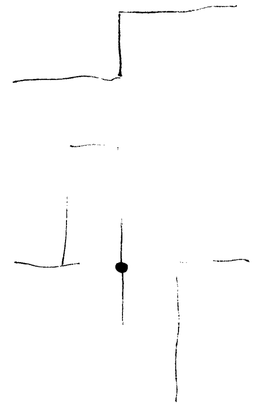
replace j by $j-1$ to center about $[i,j]$.

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

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

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

2nd 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~~ to their being severely affected by noise more than first derivative.

5.5 Image Approximation

come up with a function that approximates image and compute image properties from estimated function

let $z = f(x,y)$, a continuous intensity function.

a straight polynomial approx. would be too high a degree
do piecewise functions called facets.

Fig. 5.15 shows original image.
5.16 shows a local coordinate system for a 5x5 facet model.

Approximate image function locally at <u>every</u> pixel.
Use these functions ~~as~~ to locate edges.

simple images $\left\{\begin{array}{l}\text{piecewise constant}\\ \text{piecewise bilinear}\end{array}\right.$

more complex images. $\left\{\begin{array}{l}\text{biquadradic}\\ \text{bicubic}\\ \text{or higher.}\end{array}\right.$

done

model $f(x,y)$ as a bicubic polynomial

$$f(x,y) = k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2 + k_7 x^3 + k_8 x^2 y + k_9 x y^2 + k_{10} y^3$$

use least squares to compute the k's
can also do with masks (~~no figures~~ in 5.17). or SVD singular-value decomposition

edges – extreme maxima in 1st deriv.
zero crossing in 2nd deriv.

first deriv. in direction θ $\quad f'_\theta(x,y) = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta.$

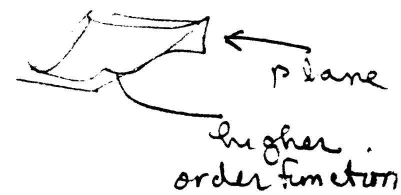
2nd deriv. in direction θ. $\quad f''_\theta(x,y) = \frac{\partial^2 f}{\partial x^2}\cos^2\theta + 2\frac{\partial^2 f}{\partial x \partial y}\cos\theta\sin\theta + \frac{\partial^2 f}{\partial y^2}\sin^2\theta.$

in general

local image intensity approximated by bicubic polynomial.

pick θ = angle of approximating plane. i.e only low order coefficients

i.e. $$\sin\theta = \frac{k_2}{\sqrt{k_2^2 + k_3^2}}$$



$$\cos\theta = \frac{k_3}{\sqrt{k_2^2 + k_3^2}}$$

$\Rightarrow$ with this choice

$$\begin{aligned} f''_\theta &= 2(3k_7 \sin^2\theta + 2k_8 \sin\theta\cos\theta + k_9 \cos^2\theta)\, x_0 \\ &\quad + 2\,(k \sin^2 + 2k_9 \sin\theta\cos\theta + 3k_{10}\cos^2\theta)\, y_0 \\ &\quad + 2\,(k_4 \sin^2\theta + k_5 \sin\theta\cos\theta + k_6 \cos^2\theta). \end{aligned}$$

for derivative only consider line in direction θ, i.e.

$$x_0 = \rho\cos\theta\,, \quad y_0 = \rho\sin\theta$$

along line $$\begin{aligned} f''_\theta(x,y) &= 6\,(k_7 \sin^3\theta + k_8 \sin^2\theta\cos\theta + k_9 \sin\theta\cos^2\theta + k_{10}\cos^3\theta)\,\rho \\ &\quad + 2\,(k_4 \sin^2\theta + k_5 \sin\theta\cos\theta + k_6 \cos^2\theta) \\ &= A\rho + B. \end{aligned}$$

$\therefore$ edge if $f''_\theta(x_0, y_0 ; \rho) = 0$ and $f'_\theta(x_0, y_0 ; \rho) \neq 0$. $\Big\}$ for some $|\rho| < \rho_0$ where ρ_0 = length of side of pixel.

mark pixel as edge pixel if location of edge falls within pixel boundaries, otherwise no.

5.6 Gaussian Edge Detection

real step edges — smoothed by ① LPF of camera optics
② BW limitations of camera electronics

approx. to image gradient

tradeoff between
① must suppress effects of noise
② must locate edge as accurately as possible

best compromise achieved with first derivative of a Gaussian

i.e. smooth an image with a Gaussian
then compute gradient

this operator is <u>NOT</u> rotationally symmetric
its symmetric along edge.
anti-symmetric $\perp$ to edge.

i.e. sensitive to edge in direction of steepest change, but insensitive to edge itself and acts as a smoothing filter in the direction along edge.

5.6.1. Canny Edge Detector

$$S[i,j] = \underbrace{G[i,j;\sigma]}_{\text{Gaussian}} \star \underbrace{I[i,j]}_{\text{image}}.$$

① Smoothed image

② compute gradient using 2x2 first difference approximations

i,j	

$$P[i,j] \approx \tfrac{1}{2}\left(S[i+1,j]-S[i,j] + S[i+1,j+1]-S[i,j+1]\right)$$

$\begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix}$

$$Q[i,j] \approx \tfrac{1}{2}\left(S[i,j]-S[i,j-1]+S[i+1,j]-S[i+1,j-1]\right)$$

$\begin{bmatrix} -1 & -1 \\ +1 & +1 \end{bmatrix}$

computed derivatives at $\left[i+\tfrac{1}{2}, j+\tfrac{1}{2}\right]$

③ Magnitude & angle images

$$M[i,j] = \sqrt{P[i,j]^2 + Q[i,j]^2}$$

$$\theta[i,j] = \arctan\left(\frac{Q[i,j]}{P[i,j]}\right)$$

can be done mostly in integer by look up methods.

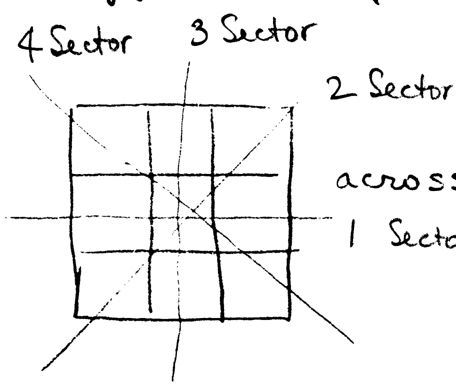
Non-maxima suppression

M[i,j] will have large values where gradient is large.
still need to find local maxima in this array to locate edges.

→ must <u>thin</u> so only points of greatest local change remain

① reduce angle θ to one of four sectors. $\zeta[i,j] = \text{Sector}(\theta[i,j])$.

② pass.



across M[i,j].

③ compare M to the two neighbors in direction specified by $\zeta[i,j]$.

④ If M[i,j] not larger, then M[i,j] = 0.

→ denote this entire process $N[i,j] = nms(M[i,j], \zeta[i,j])$

this image will still contain many false edge fragments caused by noise & fine texture.

Typically threshold N[i,j]
a single threshold usually is hard to achieve

Use double thresholding
use N[i,j] as input
use T_1 and $T_2 \approx 2T_1$ to produce two thresholded images $T_1[i,j]$ and $T_2[i,j]$.

link into contours
If a gap is encountered go to T_1 until bridged to edge in T_2

Algorithm 5.1 Canny Edge Detection

1. Smooth image with a Gaussian filter
2. compute gradient magnitude & orientation using finite-difference approximations
3. apply non maxima suppression to the gradient magnitude
4. use double thresholding algorithm to detect and link edges.

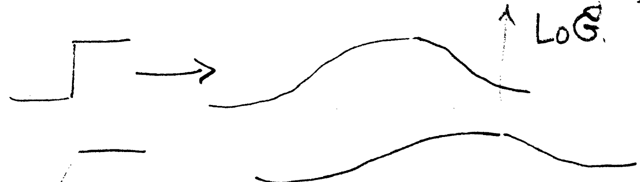
5.7 Sub-pixel Location Estimation

gradient & 2nd-order edge detection require very different algorithms.

2nd order doesn't work

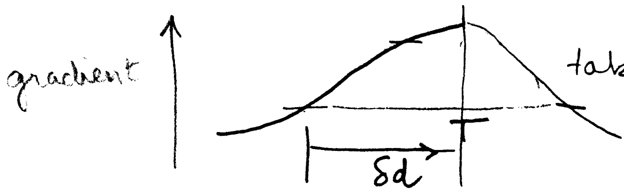
2nd order, i.e. Laplacian of Gaussian LoG
in principle, interpolate to locate zero crossing
in practice, too noisy even w/ Gaussian smoothing

gradient (still subject to noise)
Gaussian smoothing followed by gradient
LoG



If not ideal simply becomes broader.

estimate mean value



compute weighted sum of position

i.e. $$\delta d = \frac{\sum_{i=1}^{n} g_i d_i}{\sum_{i=1}^{n} g_i}$$ position, gradient

Could also compute first moment.
Lot's of statistical techniques since Gaussian profile.

5.8. Edge Detector Performance.

- criteria
  1. Probability of false edges.
  2. Prob. of missing edges.
  3. error in estimation of edge angle
  4. mean squared error of edge estimate from true edge.
  5. tolerance to distorted edges and other features such as corners and junctions

Pratt's figure of merit.

$$FM = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + d_i \alpha^2}$$

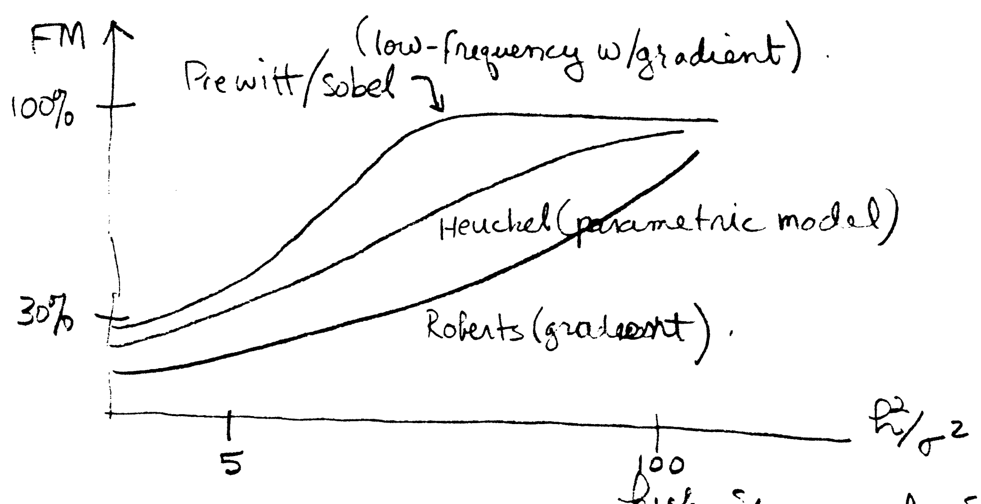
looks primarily at

I_A – detected edges.

I_I – ideal edges

d – distance error.

α – penalty for displaced edges.



h = Step edge amplitude

σ = std. deviation of Gaussian white noise.

edge relaxation – improve edge operator estimate by re-adjusting edge estimate based upon local information

algorithm:

of any edge.

$$C^0(e) = \frac{\text{gradient magnitude}}{\text{maximum gradient amplitude in image}}$$

$k = 1$

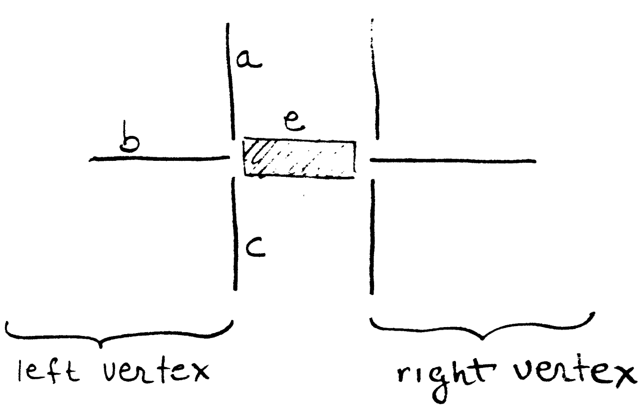
initial confidence of edge is normalized gradient

while any $C^k(e) \neq (0 \text{ or } 1)$ do
    begin
classify edge.   edge_type = f(edge_neighbors)
adjust confidence. $C^k(e) = f(\text{edge\_type}, C^{k-1}(e))$
iteration counter $k = k+1$
    end

algorithm forces all points in edge image to converge to ∅ or 1

original relaxation work done by

edge type classified by vertex type



e = edge to be updated

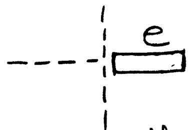
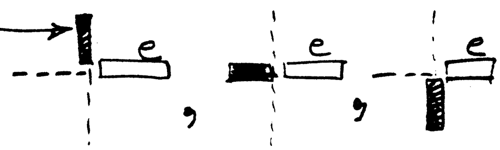
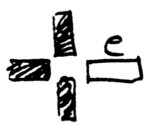
a, b, c are normalized gradient magnitudes

$m = \max(a, b, c, q)$

q = constant (a threshold)

left vertex types and associated confidences

Example
strong = 0.8
weak = 0.1
m = 0.9

type	figure	confidence	
0	 all weak edges	if all weak decrease confidence. $(m-a)(m-b)(m-c)$	$(.9-.1)(.9-.1)($ .51
1	strong edge  basically one strong edge	increase confidence. $a(m-b)(m-c)$	$.8(.8)(.8)$ .51
2	 basically two strong edges	$a\,b\,(m-c)$	confidence actually [illegible] .39
3	 basically three strong local edges	ambiguous so keep about same. $a\,b\,c$	.34

increasing strength (confidence).

actual edge type is concatenation of left <u>and</u> right vertex types

edge (e) = (i,j) with 3,3 being the strongest edge.

to update edge confidence in algorithm

increment: $C^k(e) = \min(1, C^{k-1}(e) + \delta)$

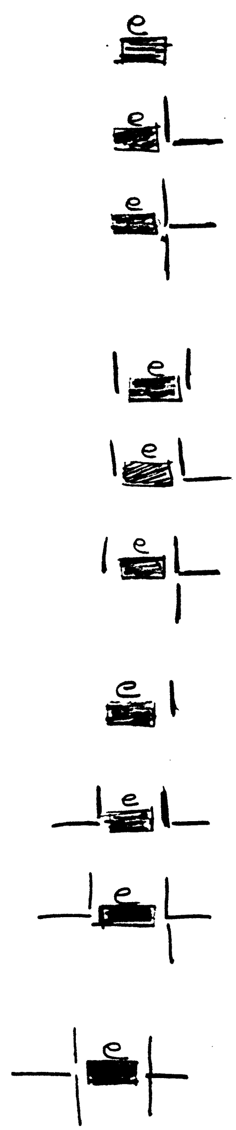
decrement: $C^k(e) = \max(0, C^{k-1}(e) - \delta)$

leave as is: $C^k(e) = C^{k-1}(e)$

where $0.1 < \delta < 0.3$ typically.

jump to here.

examples: left vertex – right vertex



decrement
0 – 0
0 – 2
0 – 3
neglect isolated edges

increment
1 – 1
1 – 2
1 – 3
connect edges whenever possible.

leave as is:
0 – 1
2 – 2
2 – 3
3 – 3
uncertain connectivity

The idea is, if a strong edge is near by update (increase) the confidence of nearby edges. As the number of nearby edges increases

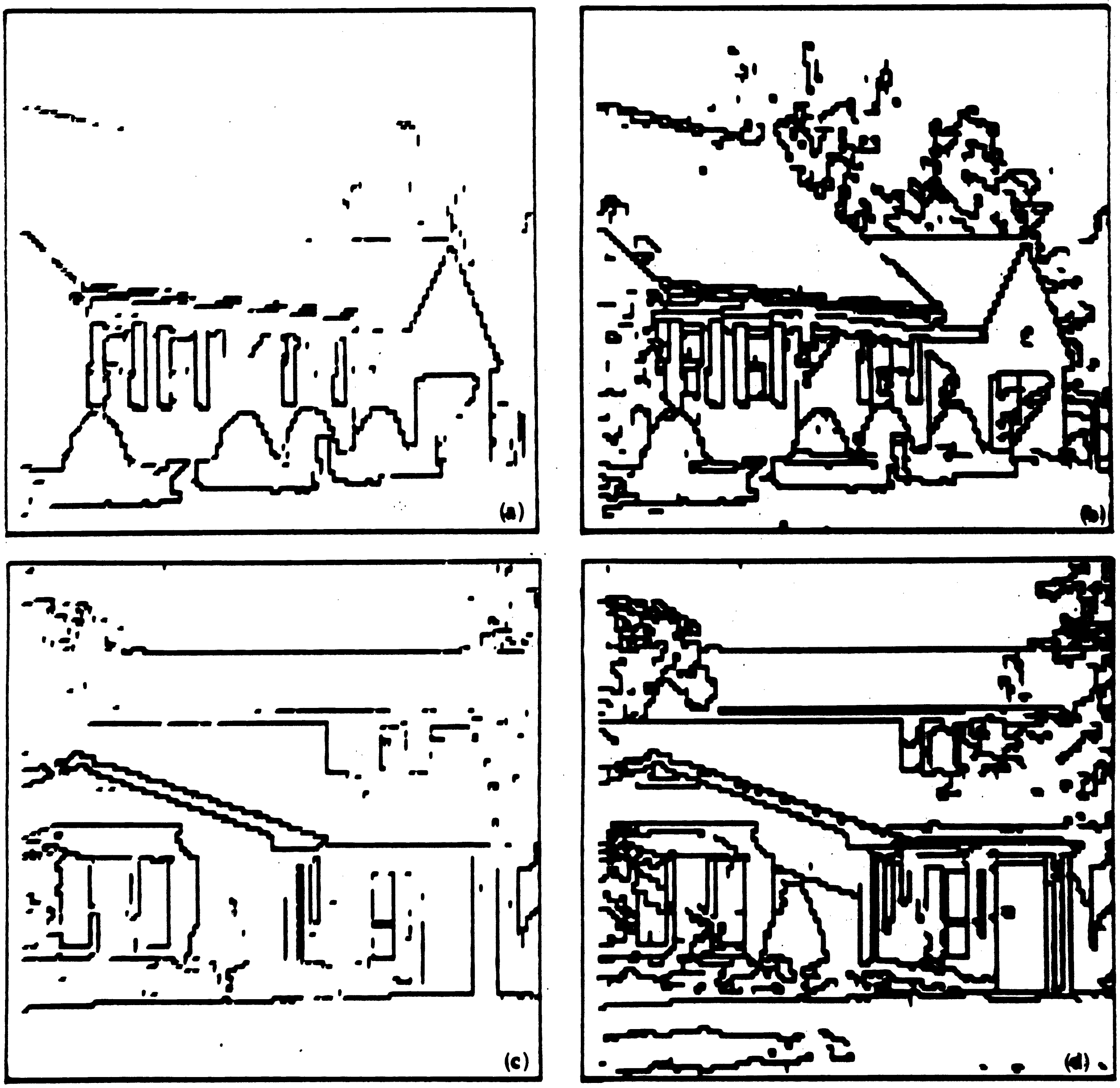


Fig. 3.22 Edge relaxation results. (a) Raw edge data. Edge strengths have been thresholded at 0.25 for display purposes only. (b) Results after five iterations of relaxation applied to (a). (c) Different version of (a). Edge strengths have been thresholded at 0.25 for display purposes only. (d) Results after five iterations of relaxation applied to (c).

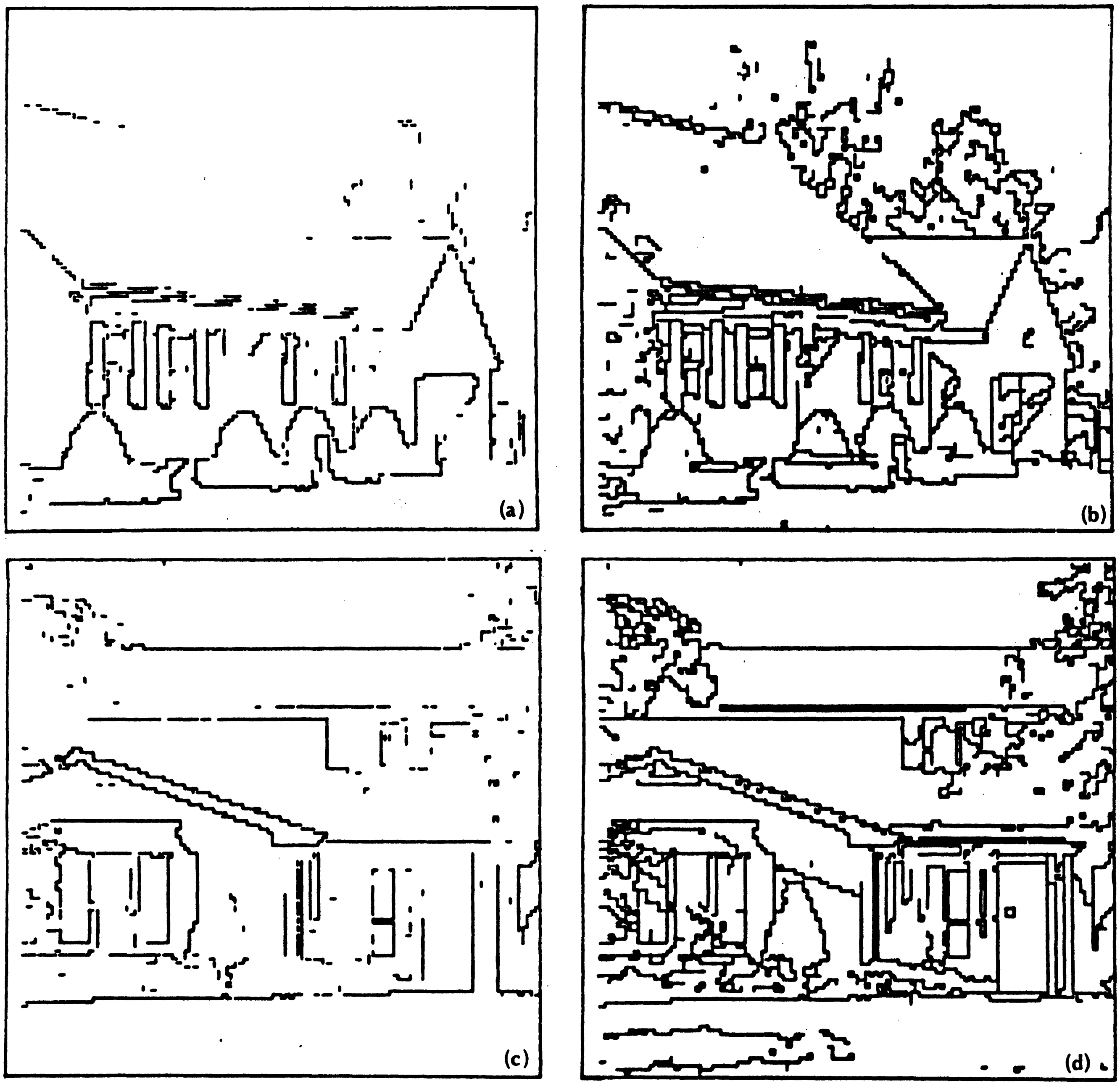


Fig. 3.22 Edge relaxation results. (a) Raw edge data. Edge strengths have been thresholded at 0.25 for display purposes only. (b) Results after five iterations of relaxation applied to (a). (c) Different version of (a). Edge strengths have been thresholded at 0.25 for display purposes only. (d) Results after five iterations of relaxation applied to (c).

edge relaxation – improve edge operator estimate by re-adjusting edge estimate based upon local information

algorithm:

$$c^0(e) = \frac{\text{gradient magnitude}}{\text{maximum gradient amplitude in image}}$$

of any edge.

$k = 1$

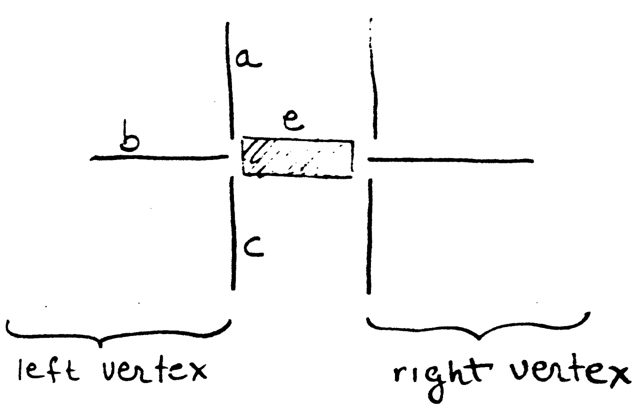
initial confidence of edge is normalized gradient

while any $c^k(e) \neq (0 \text{ or } 1)$ do
    begin
classify edge.   edge_type = f(edge_neighbors)
adjust confidence. $C^k(e) = f(\text{edge\_type}, C^{k-1}(e))$
iteration counter $k = k + 1$
    end

algorithm forces all points in edge image to converge to $\emptyset$ or 1

original relaxation work done by

edge type classified by vertex type



e = edge to be updated

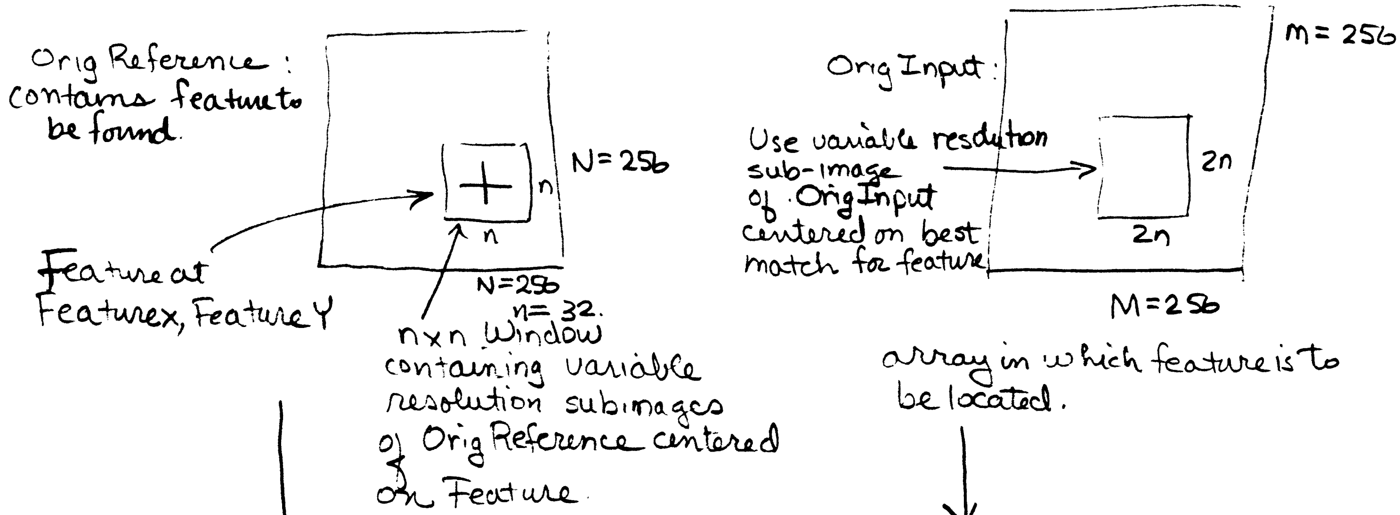
a, b, c are normalized gradient magnitudes

$m = \max(a, b, c, q)$

q = constant (a threshold)

Binary search correlation algorithm (Good for coarse to fine matching).

Used to locate a feature at some unknown location in the input image using variable resolution subimages of the input image.



Reference is a temporary array.

$2n \times 2n$
Input is a temporary array containing variable resolution sub-image of Orig Input centered on the best match.

Algorithm

1. Input := Consolidate Orig Input by a factor of $\frac{2n}{M}$ to size $2n \times 2n$.

2. Reference := Consolidate Orig Reference by the same factor.

$$\frac{2n}{M} \text{ to size } 2n\frac{N}{M} \times 2n\frac{N}{M}$$

$$\frac{2(32)}{256} = \frac{1}{4} \qquad \frac{1}{4}(256) = 64 \times 64.$$

3. Window := $n \times n$ window from Reference centered on (Feature X, Feature Y)

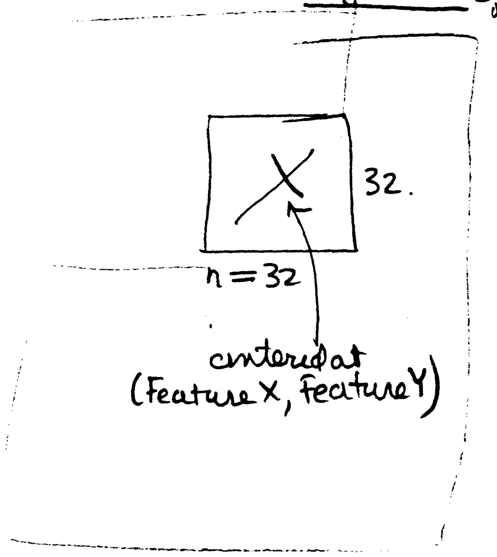
4. Calculate the correlation at $(n+1)^2$ locations in input. Normalize if necessary. Do NOT wrap around.

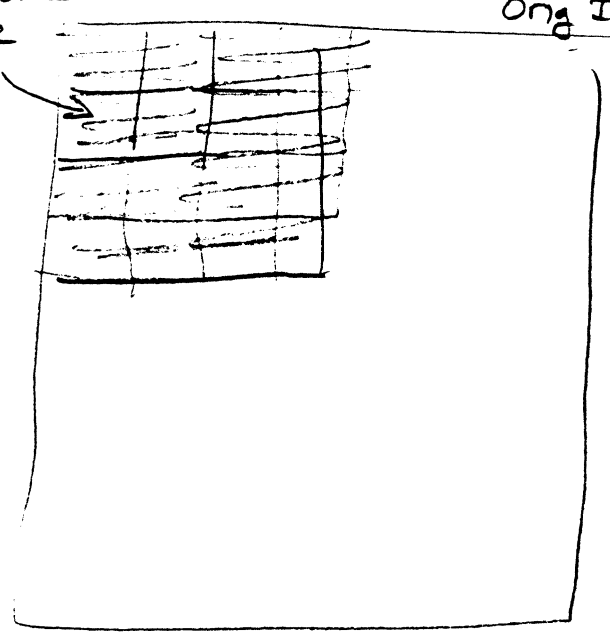
5. Input := $n \times n$ window from input enlarged by a factor of 2. centered at (Best Match X, Best Match Y)

6. Reference := reference enlarged by a factor of 2. Takes feature to new (Feature X, Feature Y)

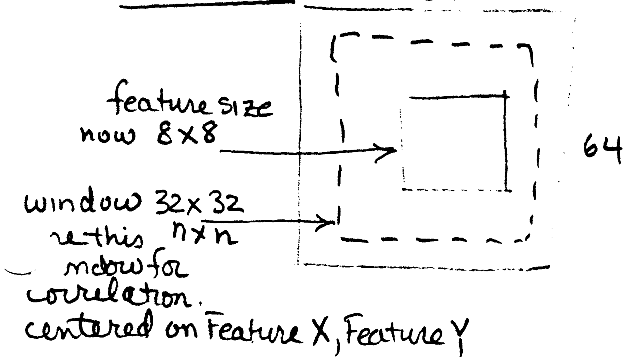
7. Go to 3. Repeat as necessary.

first steps.

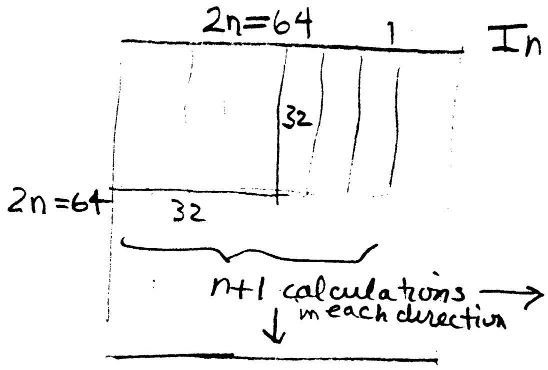




Consolidate to Reference



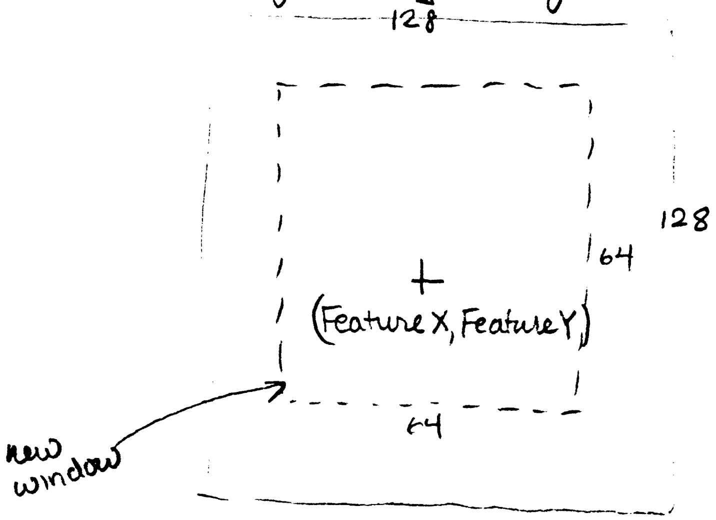
Consolidate to Input



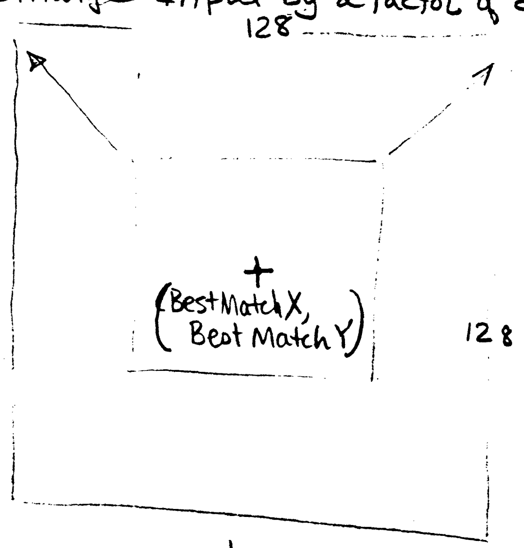
correlate Window with Input.

Iterate.

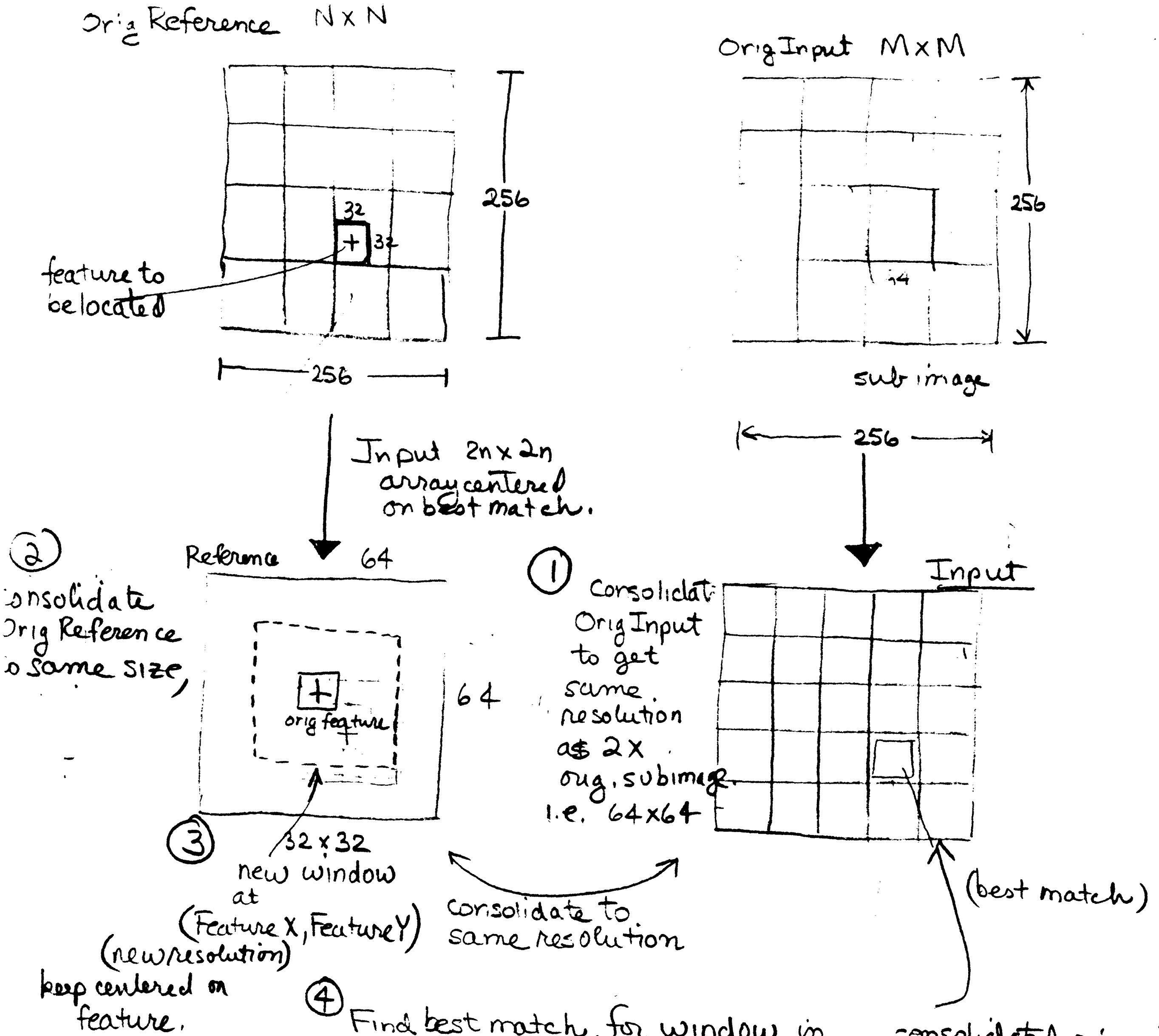
Enlarge reference by factor of 2



Enlarge Input by a factor of 2.



now do 64+1
× 64+1 calculations



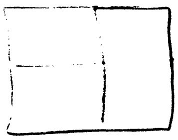
④ Find best match, for window in consolidated orig input

⑤ enlarge reference by a factor 2; enlarge Input by a factor of 2 centered at best match.

Note: Typically use twice the dimension of the original feature size to allow for tolerances in feature centering as pyramiding occurs.

basically matches gross features then fine detail.

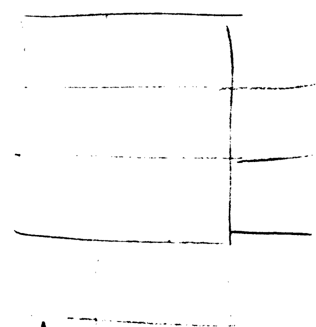
pyramid edge detection (Tanimoto)



fundamental idea: no change in greyscale values in a consolidated neighborhood implies that no grey-scale change (edge) occurs in that neighborhood.

problem: too much starting consolidation loses all edges.

```
recursive procedure refine (k,x,y)
  begin
  if k ≤ Maxlevel then
     for dx=0 until 1 do   } expand by factor of 2.
     for dy=0 until 1 do   }
        if EdgeOp(k,x+dx,y+dy) > Threshold(k)
        then refine (k+1, x+dx, y+dy)
  end;
```



```
procedure FindEdges:
  begin
  comment apply operator to every pixel in the starting level s,
          refining as necessary.
   for x:=0 until 2^s-1 do
   for y:=0 until 2^s-1 do
      if EdgeOp(s,x,y) > Threshold(s)
      then refine (s+1,x,y);
  end;
```

i.e. 4x4 or whatever.

s = level of resolution in starting image.

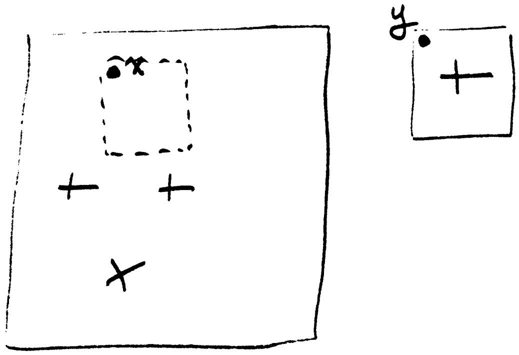
use whatever edge op you desire

can have different thresholds at different levels.

can have different operators at different levels.

Ballard and Brown

some notes on correlation using a sub-image.



range of $\underline{x}$

how to find objects of interest in image?

$$d^2(\underline{y}) = \sum_{\underline{x}} \left[f(\underline{x}) - t(\underline{x}-\underline{y}) \right]^2$$

define an Euclidian distance d

over the image

image function

slide template to location $\underline{y}$

if template matches $d=0$
otherwise $d>0$

$$= \sum_{\underline{x}} \left[\underbrace{f^2(\underline{x})}_{\text{can be}} - \underbrace{2f(\underline{x})t(\underline{x}-\underline{y})} + t^2(\underline{x}-\underline{y}) \right]$$

can be nearly constant depending upon spatial uniformity of image

constant term, the energy of the template

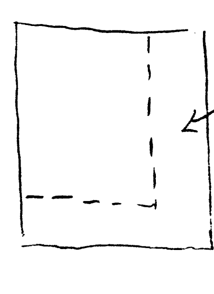
cross-correlation function $\phi_{ft}(\underline{y})$ (looks like a convolution and can be treated as a filter)

this is a correlation or dot product which is maximized when $t(x-y)$ matches $f(x)$.

two cases of template matching (correlation) — periodic (template wraps around image)

aperiodic (no wraparound can have various amounts of overlap)

to eliminate false responses normalize



can't correlate in this region for a periodic correlation

Normalization of correlation to prevent false errors:

Problems can occur due to intense noise in the image.

Example:

1	1	1
1	1	1
1	1	1

template

1	1	0	0	0
1	1	1	0	0
1	0	1	0	0
0	0	0	0	0
0	0	0	0	8

Correlation

7	4	2	x	x
5	3	2	x	x
2	1	9	x	x
x	x	x	x	x
x	x	x	x	x

correct best response

error due to large noise

x = undefined

To prevent such errors we must normalize according to the statistics of each image.

$f_1(\underline{x})$	$f_2(\underline{x})$	images to be matched
g_1	g_2	patches to be matched. Typically g_1 is all of f_1 g_1 is the patch of f_1 that is covered by the displaced patch g_2.

$$\sigma(g_1) = \sqrt{E(g_1^2) - E^2(g_1)}$$

$$\sigma(g_2) = \sqrt{E(g_2)^2 - E^2(g_2)}$$

standard deviations
E is the normal expectation operator.

Normalized correlation

$$\underline{N}(\underline{y}) = \frac{E(g_1 g_2) - E(g_1)E(g_2)}{\sigma(g_1)\sigma(g_2)}$$

← this is some sort of cross correlation

← normalized by the variances

Correlation is very expensive in terms of computer time. If we could quickly determine which areas of the picture contained "interesting" information we could limit detailed calculations to those "interesting" areas.

Sequential similarity detection algorithm (SSDA)

Correlation $$\phi_{ab}(\underline{y}) = \sum_{i,j} \underbrace{a_{ij}(\underline{x})}_{\text{image}} \; \underbrace{b_{ij}(\underline{x}-\underline{y})}_{\text{shifted template or image}}$$

If a & b are highly correlated, this summation will be essentially all 1's and yield a large ϕ_{ab}. Threshold the summation after say 10 summations. If the sum is less than T (the threshold) it is probably uncorrelated and summation will stop.

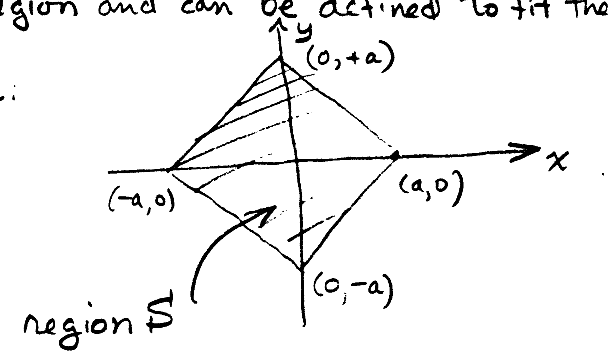
Moravec Interest Operator – produces candidate interest points based upon image activity.

define the variance operator

$$\mathrm{Var}(\underline{x}) = \mathrm{Var}(x,y) = \underbrace{\sqrt{\sum_{k,l \text{ in } S} \left[f(x,y) - f(x+k, y+l)\right]^2}}_{\text{Note that uniform image areas will have little if any variance.}}$$

The region S is a local region and can be defined to fit the application.

Moravec's original operator:



Algorithm

1. $\mathrm{IntOpVal}(\underline{x}) := \min\limits_{|y| \le 1} \left[\mathrm{var}(\underline{x}+\underline{y})\right]$ — initially set to local minimum of variance. This tells us in an average sense how much information is nearby

2. $\mathrm{IntOpVal}(x) := 0$ unless $\mathrm{IntOpVal}(\underline{x}) \ge \mathrm{IntOpVal}(\underline{x}+\underline{y})$ for $|y| \le 1$ — Now find only local maxima.

3. $\underline{x}$ is an interesting point only if $\mathrm{IntOpVal}(\underline{x}) > T$ — T is an empirically set threshold.