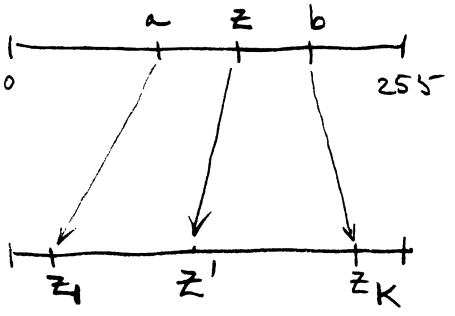
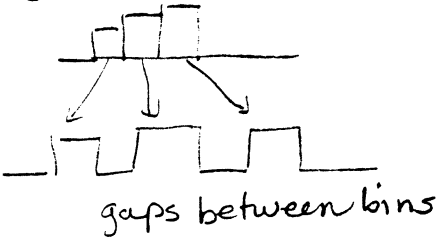
Jain - Chapter 4 Image Filtering

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



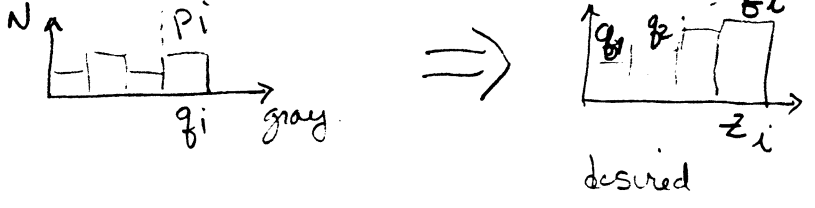
simple proportionality $z' = \left(\frac{z_K - z_1}{b-a}\right)(z-a) + z_1$

Problem what was originally



gaps between bins

If you know the distribution you want a priori then use a cumulative function



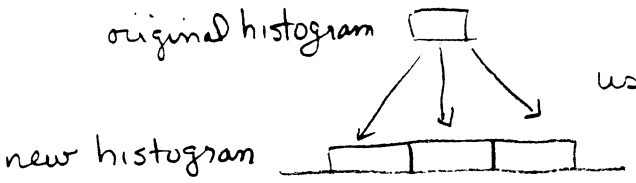
Do one level at a time

find k_1 in original image $\Rightarrow \sum_{i=1}^{k_1-1} \leq q_1 < \sum_{i=1}^{k_1} P_i$

i.e. upper and lower bounds for the bins

find k_2 $\Rightarrow \sum_{i=1}^{k_2-1} P_i \leq q_2 < \sum_{i=1}^{k_2} P_i$

One problem may be how to split up bins:



use a uniform random number generator to spread evenly

i.e. n output bins

$g_K, g_{K+1}, \cdots, g_{K+n-1}$

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

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

greatest integer function

4.2 Linear Systems.

space invariance

Input $\delta(x,y)$ → Linear system → $g(x,y)$

impulse response to $\delta(x,y)$.

if irrespective of position of input pulse called space invariant

i.e.

$\delta(x-x_0, y-y_0)$ → Linear space invariant system → Output $g(x-x_0, y-y_0)$

linearity $a \cdot f_1(x,y) + b \cdot f_2(x,y) \Rightarrow a\, h_1(x,y) + b\, h_2(x,y)$

where $f_1(x,y) \Rightarrow h_1(x,y)$

$f_2(x,y) \Rightarrow h_2(x,y)$.

For these systems, output is convolution of $f(x,y)$ with $g(x,y)$.

convolution (continuous)

$$h(x,y) = f(x,y) \star g(x,y)$$

$$= \int_{-\infty}^{\infty}\int f(x',y')\, g(x-x', y-y')\, dx'\, dy'$$

discrete

$$h[i,j] = f[i,j] \star g[i,j]$$

$$= \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]\, g[i-k, j-l]$$

depends on size of mask $n \times m$

$g[i,j]$ is called the convolution mask

↓ translate $g[\ ]$ to pixel $[i,j]$.
take weighted sum of pixels using convolution mask.

convolution is linear! and spatially invariant
However, you can change filter weights in different parts of image.

Fourier Transform

frequency component $f[k,l] = \frac{1}{4\pi^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} F(u,v)\, e^{jku}\, e^{jlv}\, du\, dv$

Fourier transform $F(u,v) = \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]\, e^{-jku}\, e^{-jlv}$

technically continuous

mathematically you can transform image and convert convolution in image domain to multiplication in frequency domain

most machine vision algorithms non-linear or spatially varying so this is not useful. Also, really only useful with large filter masks

4.3 Linear Filters — usually used to remove noise

salt & pepper noise — random occurrences of black and white

impulse noise — random occurrances of white

Gaussian noise — variations in intensity follow Gaussian or normal distribution
good for electronics & sensor noise

linear smoothing filter good for removing Gaussian noise and, usually, all other types of noise.

any filter that is NOT a weighted sum of pixels is a nonlinear filter

Mean filter

$$h[i,j] = \frac{1}{M} \sum_{(k,l) \in N} f[k,l].$$

for a 3x3 neighborhood

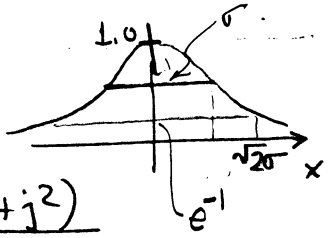
$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} f[k,l].$$

convolution mask $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ just averages.

increasing size of mask reduces noise and image detail

4.5 Gaussian Filter

look up Figure 4.11



1-D zero mean $g(x) = e^{-\frac{x^2}{2\sigma^2}}$

2-D zero mean discrete $g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$

$e^{-1} = e^{-\frac{x^2}{2\sigma^2}}$ $\quad \frac{x^2}{2\sigma^2} = 1$

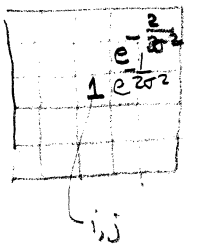
$x = \sqrt{2}\sigma$

Properties of Gaussian filters:

1. rotationally symmetric so smoothing (and edges) same in all directions
2. has a single lobe (spatially localizes smoothing)
3. Fourier transform of a Gaussian is a Gaussian
   ⇒ single lobe in frequency domain
   noise — usually high-frequency (noise & fine texture)
   edge — typically components at both high and low frequencies
   ∴ retain edge info while losing noise if use Gaussian filter
4. larger σ implies wider Gaussian filter and greater smoothing
5. can implement large Gaussian filters because Gaussian filter separable into 1-D functions.

4.5.1 Rotational Symmetry of Gaussian filter.

2-D Gaussian $\boxed{g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}}$ ← definition



in polar coordinates $r^2 = i^2 + j^2$ so that

$$g(r,\theta) = e^{-\frac{r^2}{2\sigma^2}}$$

No rotational dependence, symmetric

4.5.2. Fourier Transform Property of Gaussian filter.

(do in frequency domain methods)

$$\mathcal{F}\{g(x)\} = \int_{-\infty}^{\infty} g(x)\, e^{-j\omega x}\, dx$$

symmetric

$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}}\, e^{-j\omega x}\, dx$$

$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} (\cos\omega x - j\sin\omega x)\, dx$$

$$= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \cos\omega x\, dx - j\int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \sin\omega x\, dx.$$

($\sin\omega x$ → 0)

symmetric — anti-symmetric

$$\therefore\ \mathcal{F}\{g(x)\} = \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} \cos\omega x\, dx = \sqrt{2\pi}\,\sigma\, e^{-\frac{\omega^2}{2\nu^2}} \qquad \nu^2 = \frac{1}{\sigma^2}$$

spread of Gaussian in space is σ
spread of Gaussian in frequency is $\frac{1}{\sigma}$

i.e. narrow spatial Gaussian does less smoothing since it affects fewer frequencies.

4.5.3 Gaussian Separability of Gaussians

$$g[i,j] \star f[i,j] = \sum_{k=1}^{m} \sum_{l=1}^{n} g[k,l]\, f[i-k, j-l]$$

(f: image)

Gaussian filter

$$= \sum_{k} \sum_{l} e^{-\frac{k^2+l^2}{2\sigma^2}} f[i-k, j-l]$$

$$= \sum_{k} e^{-\frac{k^2}{2\sigma^2}} \left[\sum_{l} e^{-\frac{l^2}{2\sigma^2}} f[i-k, j-l] \right]$$

convolution with vertical 1-D Gaussian function

followed by 1-D horizontal convolution

these could be reversed.

could also transpose image and repeat coding

4.5.4 Cascading Gaussians

convolution of Gaussians yields larger Gaussian

$$g[k] \star g[k] = \sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} e^{-\frac{(k-l)^2}{2\sigma^2}}$$

both σ in this case.

$$= \sum_{l=1}^{n} e^{-\frac{\left(\frac{k}{2}+l\right)^2}{2\sigma^2}} e^{-\frac{\left(\frac{k}{2}-l\right)^2}{2\sigma^2}} \qquad l = l + k/2.$$

$$= \sum_{l=1}^{n} e^{-\frac{\left(2l^2 + \frac{k^2}{2}\right)}{2\sigma^2}}$$

$$= e^{-\frac{k^2}{4\sigma^2}} \sum_{l=1}^{n} e^{-\frac{2l^2}{2\sigma^2}}$$

$$= c\, e^{-\frac{k^2}{2[\sqrt{2}\sigma]^2}}$$

this is $\sigma + \sigma \rightarrow \sqrt{2}\sigma$

then scaled by area of Gaussian filter

4.5.5. Designing Gaussian filters.

Good approximation is coefficients of binomial expansion

```
          1
        1   1
      1   2   1
    1   3   3   1
etc. 1   4   6   4   1
```

Pascal's triangle

use integer coefficients

Good up to filter size of 10. (numbers get too large for many computers)

So, simply repeat application of Gaussians

find σ by fitting Gaussian to coefficients

Or, compute directly from discrete Gaussian distribution

$$g[i,j] = k\, e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

k: normalizing constant

$$\frac{g[i,j]}{k} = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

You can pick σ.

pick value for σ and evaluate

You usually want integer filter weights so normalize to the smallest value. Express all coefficients as multiples of this value.

But, weights do NOT sum to 1.

$\Rightarrow$ Normalize to ensure that regions of uniform intensity are not affected.

Normalize by $\sum\sum g[i,j]$

because you will change intensity

Noise

2/2/95

- salt & pepper noise – random occurrences of black and white
- impulse noise – random occurrances of white
- Gaussian noise – variations in intensity follow Gaussian or normal distribution
  good for electronics & sensor noise

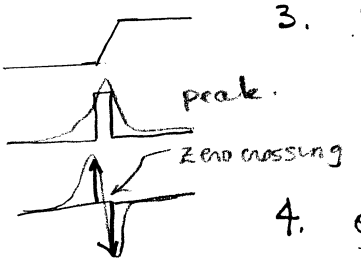
linear smoothing filter good for removing Gaussian noise and, usually, all other types of noise.

any filter that is NOT a weighted sum of pixels is a nonlinear filter

5.4 Laplacian of Gaussian LoG combines Gaussian filtering with Laplacian

Characteristics

1. smoothing filter is Gaussian
2. enhancement step is 2nd derivative (laplacian in 2-D).
3. detection criterion – (isotropic operator). presence of zero crossing in 2nd derivative accompanied by corresponding large peak in 1st derivative



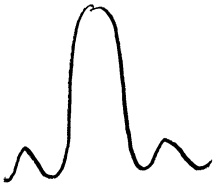
4. estimate edge location to sub pixel resolution using linear interpolation.

$$h(x,y) = \nabla^2 [(g(x,y) \star f(x,y)]$$

(laplacian; gaussian filter; image)

derivative rule for convolution

$$h(x,y) = \nabla^2 g(x,y) \star f(x,y).$$



$$\text{and } \nabla^2 g(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Also called Mexican Hat or sombrero operator.

convolve image w/ Gaussian
compute laplacian of result
$\equiv$ convolve image with filter that is Laplacian of the Gaussian filter.

blurring depends on σ
one idea is to use filters of different σ and analyze behavior of edges at these different scales.

4.4 median filter

good at removing salt & pepper noise and impulse noise without loss of detail

<u>non-linear, but spatially invariant</u>

for a 3x3 window (typically want to use odd # of pixels)

1. sort pixels into ascending order by gray level
2. select value of middle pixel as new value for pixel [i,j].

Spatial filters - general principles and implementation issues (4.1)
- feature extraction, edge detection (7.1)
- matched filtering and template matching (3.3.8, 9.3.1)
- image enhancement and noise reduction (4.1-4.3)
(little MATLAB primer).

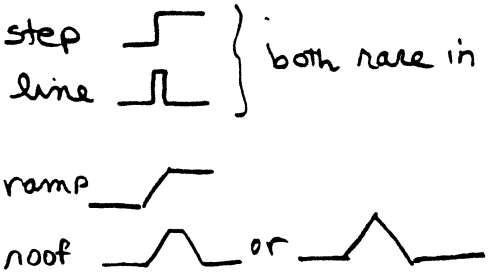
Reading Assignment 2.1 to 2.4

Chapter 5 - Edge Detection

edge - significant local change in the image (used to eliminate noise), i.e. a point operator
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



edges can have both line & step characteristics

Definitions

1. edge point — point $p[i,j]$ (can be integer, or sub-pixel usually in coordinate frame of image.) at the location of a significant local intensity change in the 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 +) 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).

5.2 Steps in Edge Detection

1. filtering – gradient very sensitive to noise
   only 1×2 or 2×2.
   more filtering → weaker edges.

2. enhancement – emphasize pixels where there is a significant change in local intensity (usually gradient magnitude).

3. detection – typically, local thresholding of magnitude

4. localization – estimate (up to subpixel) location and orientation.

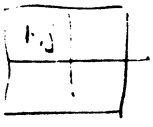
5.6.1. Canny Edge Detector

$$S[i,j] = G[i,j;\sigma] \star I[i,j]$$

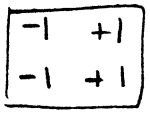
(Gaussian, image)

① Smoothed image with Gaussian

② compute gradient using 2×2 first difference approximations

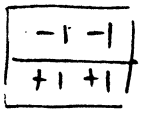


$$P[i,j] \approx \frac{1}{2}\left(S[i+1,j] - S[i,j] + S[i+1,j+1] - S[i,j+1]\right)$$



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

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



③ compute magnitude & angle images

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

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

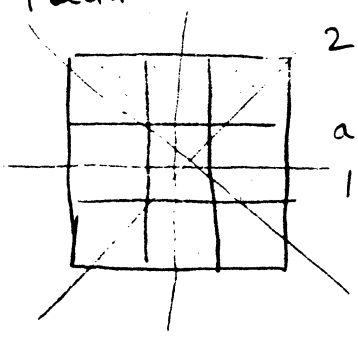
can be done mostly in integer by look up methods.

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 thin 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 $N[i,j] = \underset{\text{non-maximal suppression}}{nms}(m[i,j], \zeta[i,j])$

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

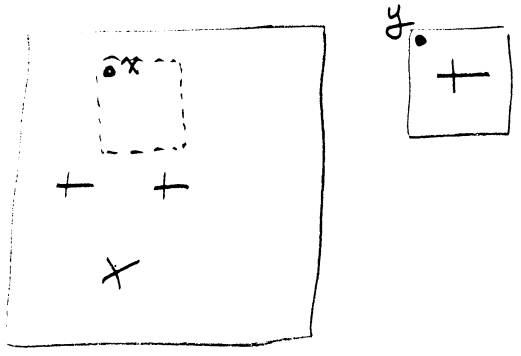
Typically threshold $N[i,j]$
a single threshold usually is hard to achieve

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 linked to edge in T_2

Ballard and Brown

some notes on correlation using a sub-image.



range of $\underline{x}$

how to find objects of interest in image?

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

- $d^2(\underline{y})$: define an Euclidian distance d; if template matches $d=0$, otherwise $d>0$
- $\sum_{\underline{x}}$: over the image
- $f(\underline{x})$: image function
- $t(\underline{x}-\underline{y})$: slide template to location $\underline{y}$

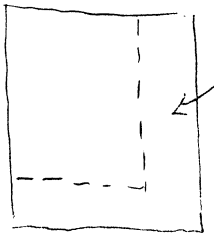
$$= \sum_{\underline{x}} [f^2(\underline{x}) - 2f(\underline{x})t(\underline{x}-\underline{y}) + t^2(\underline{x}-\underline{y})]$$

- $f^2(\underline{x})$: can be nearly constant depending upon spatial uniformity of image
- $2f(\underline{x})t(\underline{x}-\underline{y})$: cross-correlation function $\phi_{ft}(\underline{y})$ (looks like a convolution and can be treated as a filter)
- $t^2(\underline{x}-\underline{y})$: constant term, the energy of the template

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

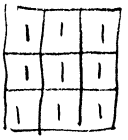


can't correlate in this region for a periodic correlation

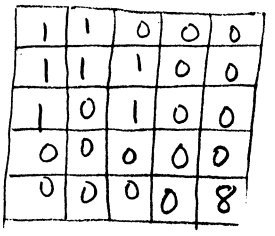
Normalization of correlation to prevent false errors:

Problems can occur due to intense noise in the image.

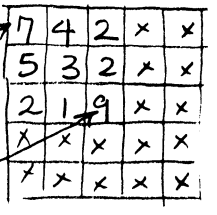
Example:



template



Correlation



correct best response

error due to large noise

X = undefined

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

$f_1(\underline{x})$ $f_2(\underline{x})$ images to be matched

q_1 q_2 patches to be matched. Typically q_1 is all of f_1. q_1 is the patch of f_1 that is covered by the displaced patch q_2.

$$\sigma(q_1) = \sqrt{E(q_1^2) - E^2(q_1)}$$

$$\sigma(q_2) = \sqrt{E(q_2)^2 - E^2(q_2)}$$

standard deviations

E is the normal expectation operator.

Normalized correlation

$$\underline{N}(y) = \frac{E(q_1 q_2) - E(q_1)E(q_2)}{\sigma(q_1)\sigma(q_2)}$$

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

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.

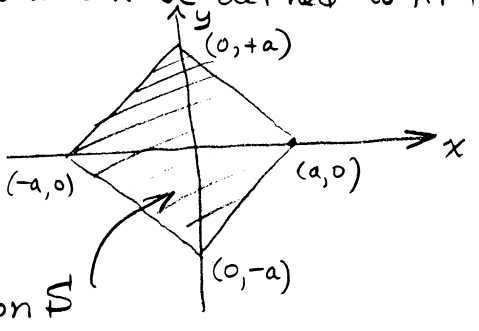
define the variance operator

$$Var(\underline{x}) = Var(x,y) = \sqrt{\sum_{k,l \text{ in } S} \left[f(x,y) - f(x+k, y+l)\right]^2}$$

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. $\text{IntOpVal}(\underline{x}) := \min_{|y|\le 1} [var(\underline{x}+\underline{y})]$

   initially set to local minimum of variance. This tells us in an average sense how much information is nearby

2. $\text{IntOpVal}(x) := 0$ unless $\text{IntOpVal}(\underline{x}) \ge \text{IntOpVal}(\underline{x}+\underline{y})$ for $|y| \le 1$

   Now find only local maxima.

3. $\underline{x}$ is an interesting point only if $\text{IntOpVal}(\underline{x}) > T$

   T is an empirically set threshold.

■ DISCRETE VERSIONS OF CONVOLUTION AND CORRELATION

The discrete version of convolution and correlation follow from the analog formulations. The discrete convolution of functions f_1 and h, to produce f_2, may be written as

$$f_2(m, n) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f_1(i, j)\, h(m - i, n - j) \qquad (3\text{-}42e)$$

where the limits on the summation are due to the assumed rectangular region of support (or an $M \times N$ "window") of h. In contrast with Eq. 3-42e, "periodic" convolution of the functions f and h is defined as

$$f_2(m, n) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f_1(i, j)\, h([m - i], [n - j]) \qquad (3\text{-}42f)$$

where h is assumed periodic with

$$[m - i] = (m - i) \bmod M$$

and

$$[n - i] = (n - i) \bmod N$$

Note, without proof, that the second definition of discrete convolution is necessary to yield the multiplicative property of the corresponding discrete signal spectra in the frequency domain (i.e., the DFTs of f_1 and h). Recall that due to sampling, f_1 is also assumed periodic. This assumption poses a small dilemma, since neither function, practically speaking, is periodic. To overcome this dilemma, and therefore be able to use the frequency domain analysis tools, we assume that f_1 and h are defined over larger sampling lattices whose corresponding extents are much greater than the regions of support of f_1 and h. *This is accomplished in practice by forming the larger arrays by padding* f *and* h *in two dimensions with zeros to achieve the larger extent.*

Similarly, the discrete correlation of f_1 and h may be written by analogy with Eq. 3-42c as

$$f_2(m, n) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f_1(i, j)\, h(m + i, n + j) \qquad (3\text{-}42g)$$

The graphical interpretations for these operations using discrete functions are analogous to their continuous counterparts.

Relation between Convolution and the Fourier Transform

It is a well known 1-D result that convolution of two waveforms in the spatial domain corresponds to the equivalent operation of multiplication of their respective Fourier transforms in the Fourier (or frequency) domain. This result carries over directly into 2-D in both the continuous and discrete formulations, with the proviso that in the latter periodic convolution is used. This result is significant since it facilitates study of the frequency domain behavior of many of the discrete spatial or "window" functions used for smoothing, sharpening, and other enhancement and restoration functions. Furthermore, efficient implementation of many spatial

domain computations may be achieved in the frequency domain. This partially explains the utility of fast, parallel, and optical implementation of the Fourier transform.

The 2-D Discrete Fourier Transform

This transform was alluded to in Eq. 3-34 and Eq. 3-35. In addition, we have seen how to write its 1-D counterpart in matrix notation. Consider a 2-D image function matrix $[f]$, written as the $N \times N$ matrix:

notice we start in upper left.

this is the sampled image.

$$[f] = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & & & \\ \vdots & & & \\ f(N-1,0) & & \cdots & f(N-1,N-1) \end{bmatrix} \tag{3-43}$$

The 2-D Discrete Fourier Transform (DFT) and Discrete Inverse Fourier Transform (DIFT) definitions, respectively, may initially be written using summation notation

(DFT)

$$F(k,l) = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)\exp\left[\frac{-2\pi j(km+ln)}{N}\right] \tag{3-44}$$

and

(DIFT)

$$f(m,n) = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} F(k,l)\exp\left[\frac{2\pi j(km+ln)}{N}\right] \tag{3-45}$$

Three notes regarding this transform pair are in order:

This is different than Horn.

1. Note that the scale factor $(1/N^2)$ in Eqs. 3-44 and 3-45 has been distributed equally between the DFT and the DIFT. This is a matter of convenience; it is equally valid to formulate these transforms distributing the scale factor differently.
2. Continuous versions of these transforms are considered later.
3. The DIFT of Eq. 3-45 may be implemented using the structure of the forward transform Eq. 3-44, by replacing coefficients $F(k, l)$ with their conjugates and conjugating the result; that is, an alternate formulation in the form of Eq. 3-44 is

$$f(m,n) = \left[\frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} F^*(k,l)\exp\left[-\frac{2\pi j(km+ln)}{N}\right]\right]^* \tag{3-45b}$$

An examination of the exponential term in Eqs. 3-44 and 3-45 indicates

$$\exp\left[\frac{-2\pi j(km+ln)}{N}\right] = \exp\left[\frac{-2\pi jkm}{N}\right]\exp\left[\frac{-2\pi jln}{N}\right] \tag{3-46}$$

Each of the factors on the right hand side of Eq. 3-46 is a complex number of

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 every pixel.
Use these functions to locate edges.

simple images { piecewise constant; piecewise bilinear }
more complex images { bi quadradic; bi cubic; or higher } — some

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

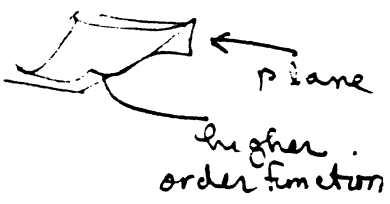
in general

local image intensity approximated by bicubic polynomial.

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

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

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



$\Rightarrow$ with this choice

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

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

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

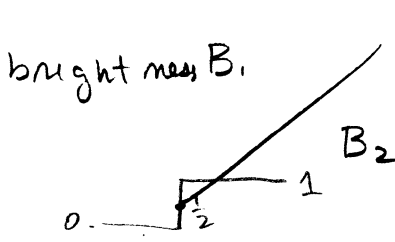
along line
$$f''_\theta(x,y) = 6(k_7\sin^3\theta + k_8\sin^2\theta\cos\theta + k_9\sin\theta\cos^2\theta + k_{10}\cos^3\theta)\rho + 2(k_4\sin^2\theta + k_5\sin\theta\cos\theta + k_6\cos^2\theta)$$

$$= A\rho + B.$$

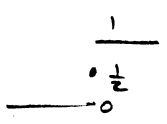
$\therefore$ edge if $f''_\theta(x_0, y_0; \rho) = 0$ and $f'_\theta(x_0, y_0; \rho) \neq 0$. } for some $|\rho| < \rho_0$ where ρ_0 = length of side of pixel

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

8.2 Differential operators (use to find edges of objects).



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



if edge is along line $x\sin\theta - y\cos\theta + \rho = 0$. equation of edge

image brightness function $B(x,y) = B_1 + (B_2 - B_1)\, u(x\sin\theta - y\cos\theta + \rho)$

by inspection the parial derivatives are, of the brightness function are.

$\frac{\partial}{\partial x} u(x\sin\theta - y\cos\theta + \rho) \rightarrow \sin\theta\, \delta(x\sin\theta - y\cos\theta -$

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

the derivative of a unit step is the $\sin\theta$, a magnitude, and an impulse

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

mathematical basis

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 notations & translations of pattern

Squared gradient

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

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

↑ no sin or cos θ

is non-linear but notationally symmetric, finds all angles equally well

consider generating higher order derivatives

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

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

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

Laplacian

known as doublet.

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

also rotationally symmetric

rotationally symmetric.
linear

quadratic Laplacian

non linear

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

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

Laplacian is only one that is

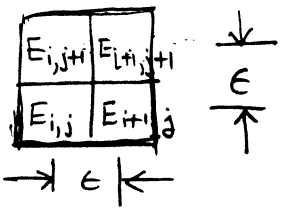
- linear
- maintains sign of brightness

} allows up to reconstruct image from edge image

8.3 Discrete approximations to derivative operators.

2×2 group of pixels

Use 2×2 matrix



separated by ϵ.

slope at top $= \frac{E_{i+1,j+1} - E_{i,j+1}}{\epsilon}$.

slope at bottom $= \frac{E_{i+1,j} - E_{i,j}}{\epsilon}$

derivatives can be approximated

average → gradient

$$\overline{\frac{\partial E}{\partial x}} \approx \frac{1}{2\epsilon}\Big(\underbrace{(E_{i+1,j+1} - E_{i,j+1})}_{\text{upper derivative}} + \underbrace{(E_{i+1,j} - E_{ij})}_{\text{lower derivative}}\Big) \leftarrow \text{sum and average.}$$

for averaging; for slope

$$\overline{\frac{\partial E}{\partial y}} \approx \frac{1}{2\epsilon}\Big((E_{i+1,j+1} - E_{i+1,j}) + (E_{i,j+1} - E_{i,j})\Big)$$ Same in y direction.

squared gradient (tells nothing about direction of edge)

gradient tells us that.

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

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

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

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

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

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

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

just multiply out four terms to get 16 terms.

using above 2×2's.

just use definitions at one point.

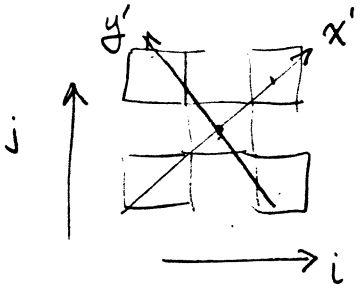
$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{1}{2\epsilon^2}\left(E_{i+1,j+1} - E_{i,j}\right)^2 + \left(E_{i,j+1} - E_{i+1,j}\right)^2$$

$$= \frac{2E_{i+1,j+1}^2 - 4E_{i+1,j+1}E_{ij} + 2E_{ij}^2 + 2E_{i,j+1}^2 - 4E_{i,j+1}E_{i+1,j} + 2E_{i+1,j}^2}{4\epsilon^2}$$

results are the same.

Laplacian (cont.)

consider a 45° rotated coordinate system. define a different Laplacian.



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

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

where does the 2 come from! comes from using $\sqrt{2}\epsilon$ instead of ϵ

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

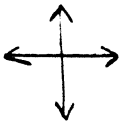
corresponding stencil.

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

1		1
	-4	
1		1

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

	1	
1	-4	1
	1	



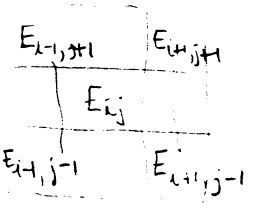
a popular estimate of the Laplacian is then

$\frac{2}{3}$ ⟷ + $\frac{1}{3}$ ⤫

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

1	4	1
4	-20	4
1	4	1

j ↑ i →



for the quadratic Laplacian

at $x = i+1$ $\frac{\partial E}{\partial y} \approx \frac{E_{i+1,j+1} - E_{i+1,j-1}}{2\epsilon}$ at $x = i-1$ $\frac{\partial E}{\partial y} \approx \frac{E_{i-1,j+1} - E_{i-1,j-1}}{2\epsilon}$

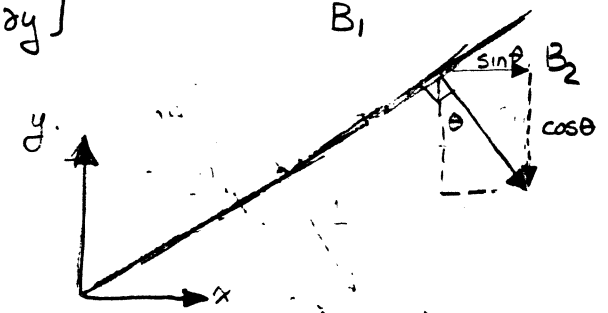
$$\frac{\partial E}{\partial x \partial y} = \frac{\left.\frac{\partial E}{\partial y}\right|_{i+1} - \left.\frac{\partial E}{\partial y}\right|_{i-1}}{2\epsilon} \approx \frac{E_{i+1,j+1} - E_{i+1,j-1} - E_{i-1,j+1} + E_{i-1,j-1}}{4\epsilon^2}$$

$$\frac{\partial E}{\partial x \partial y} = \frac{1}{4\epsilon^2}$$

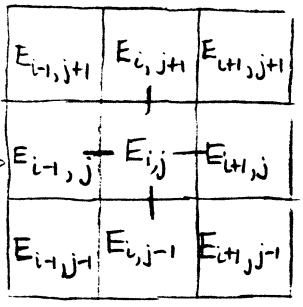
-1		1
1		-1

can also do other terms!

gradient $\begin{bmatrix} \frac{\partial E}{\partial x} \\ \frac{\partial E}{\partial y} \end{bmatrix} = (B_2 - B_1)\,\delta(x\sin\theta - y\cos\theta + \rho) \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix}$.



gradient points in direction of largest increase. !



approximating 2nd derivative requires at least a 3×3 mask.

$$\frac{\partial^2 E}{\partial x^2} = \frac{\frac{E_{i+1,j} - E_{i,j}}{\epsilon} - \frac{E_{i,j} - E_{i-1,j}}{\epsilon}}{\epsilon} = \frac{E_{i+1,j} - 2E_{i,j} + E_{i-1,j}}{\epsilon^2}$$

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

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

taking 2nd derivatives turns out rather nicely.

Laplacian should be symmetric

Look at what we get if we combine

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

these are weights

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

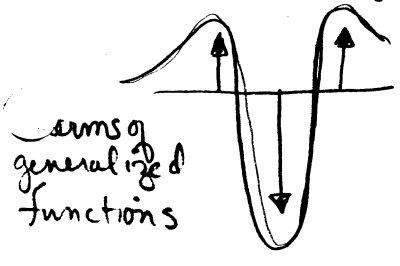
0	1	0
1	-4	1
0	1	0

this is the normalization I was referring to.

is a good approximation of a Laplacian

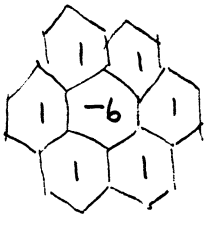
(hard to approximate symmetry with a square grid)

before we had to worry about which ~~edges~~ corners were connected we have the same problem in approximating the Laplacian do we include corners or not.



terms of generalized functions

$\frac{2}{3\epsilon^2}$



this is a better approximation as it is really symmetric

Steps in Edge Detection

1. filtering – the gradient is very sensitive to noise as it is only 1x2 or 2x2

   more filtering gives weaker edges

2. enhancement – emphasize pixels where there is a significant change in local intensity

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

3. detection – typically use local thresholding of magnitude

4. localization – locate the edge

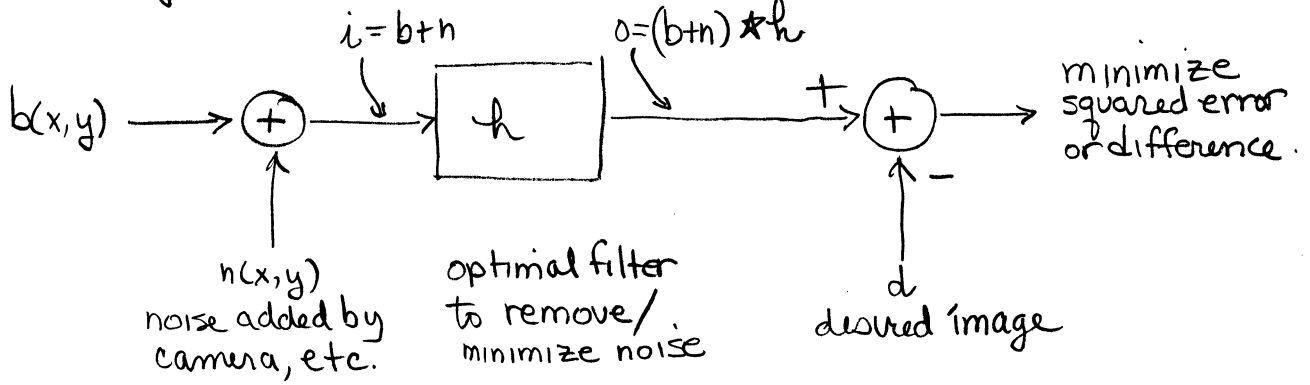
   estimate (upto subpixel) location and orientation

optional

Local operators and noise

A major problem with edge operators is that they are basically derivative operators which are very sensitive to noise.

Several ways to handle this:



See Chapter 6 of Horn. optimum filter given by $H_{opt} = \frac{\Phi_{id}}{\Phi_{ii}}$

Consider attempting to optimize Laplacian

Assume additive noise in frequency domain

Compute autocorrelation of i $\Phi_{ii} = \Phi_{bb} + \Phi_{bn} + \Phi_{nb} + \Phi_{nn}$ (the Φ_{bn} and Φ_{nb} terms $\to 0$)

cross-correlation with d in space domain $\phi_{id} = i \otimes d = i * \nabla^2 b = \nabla^2 (i * b)$ (⊗ = correlation) =

Now, Fourier Transform

$$\Phi_{id} = -(u^2+v^2)\,\Phi_{ib} = -(u^2+v^2)(\Phi_{bb} + \Phi_{bn})$$

($\Phi_{bn} \to 0$: noise & signal are uncorrelated!)

Optmum filter

$$H_{optimum} = -(u^2+v^2)\,\frac{\Phi_{bb}}{\Phi_{bb}+\Phi_{nn}} = -(u^2+v^2)\,\frac{1}{1+\frac{\Phi_{nn}}{\Phi_{bb}}}$$

Laplacian viewed as a filter

This is the optimization Optimal filter to remove/ reduce noise.

decreases gain in regions of high noise

Consider the region of support of the optimal filter we derived for estimating the Laplacian in the presence of additive, white noise.

$$H_{opt} = -(u^2+v^2)\,G$$

(u^2+v^2): Laplacian

G: optimal filter (which is a low-pass filter) usually

This is actually $g \otimes \nabla^2$

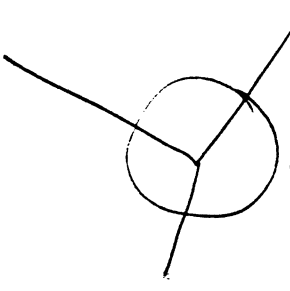
which has a larger region of support than ∇^2 alone. In general, you need to use larger regions to compute good edges.

comment on this.

In fact, for N pixels in the estimator (where each pixel is an independent random variable) with standard measurement error σ, the standard deviation of the average goes as $\frac{\sigma}{\sqrt{\frac{1}{2}N}}$

Conclusion, make N as large as possible without averaging over edge regions.

Corners are special cases of edges. Typically, you need to estimate the edges and intersect them to locate the corners.



region of support basically "blurs" the corner.

It makes sense to suppose that Φ_{bb} is band limited since b is usually low-pass filtered and/or windowed.

In cylindrical coordinates, suppose $\Phi_{bb} \sim \frac{S^2}{\rho^2}$ signal often goes as $\frac{1}{uv}$ or $\frac{1}{\rho^{3/2}}$ so this is ok.

and $\Phi_{nn} \sim N^2$ constant, white noise

Then, $$H_{optimum} = -\rho^2 \frac{\frac{S^2}{\rho^2}}{\frac{S^2}{\rho^2} + N^2} = -\rho^2 \frac{S^2}{S^2 + \rho^2 N^2}$$

for high frequencies OR large noise powers gain reduces to a constant

for mid frequencies or low noise power $H_{opt} \rightarrow -\rho^2$

the desired Laplacian

The region of support is very crucial to edge detection

Robert's original edge operator (MIT, 1965) was a simple gradient operator which detected brightness changes

Original Robert's operator:

$$R1 = \begin{pmatrix} -1 & 0 \\ 0 & \textcircled{1} \end{pmatrix}$$ $-45°$ direction

These are the (0,0) coordinates

$$R2 = \begin{pmatrix} 0 & 1 \\ -1 & \textcircled{0} \end{pmatrix}$$ $+45°$ direction

These are called compass directions

The Laplacian is, in the limit, a point operator