Computer Vision

MSc Informatics option GVR

James L. Crowley

Fall Semester 21 October 2008

Lesson 2 (version 2)
Image Formation and Description

Lesson Outline:

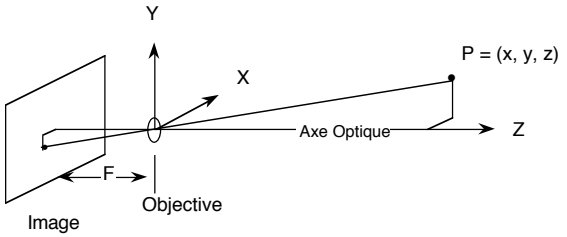
1 The Pinhole Camera Model 2
1.1 Homogeneous Coordinates and Tensor Notation 3
1.2 Coordinate Transforms : From Scene to Camera Coordiantes 6
1.3 Projective Transforms: from the scene to the retina 7
1.4 From the Retina to Digitized Image 8
2 Describing Contrast 12
2.1 Roberts Cross Edge Detector 13
2.2 The Sobel Detector 14
2.3 Difference Operators: Derivatives for Sampled Signals 15
2.4 Smoothing: The Binomial Low pass filter. 18
2.5 Edge Detection using integer coefficient filters: 20
2.6 Non-maximum suppression. 22
3 The Hough Transform 23
3.1 Generalisation of the Hough Transform 24
4 Second Derivatives. 25
4.1 Zero Crossings in the second derivative. 27

1 The Pinhole Camera Model

A "camera" is a closed box with an aperture (a "camera obscura"). Photons are reflected from the world, and pass through the aperture to form an image on the retina. Thus the camera coordinate system is defined with the aperture at the origin.

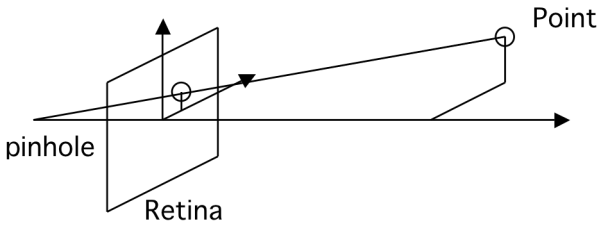
The Z (or depth) axis runs perpendicular from the retina through the aperture.
The X and Y axes define coordinates on the plane of the aperture.

Physical Camera Model



Points in the scene are projected to an "up-side down" image on the retina.

The field of computer vision often uses the "Central Projection Model" for the camera. In the Central Projection Model, the retina is placed in Front of the projective point.



We will model the camera as a projective transformation from scene coordinates, S, to image coordinates, i.

$$\vec{\mathrm{Q}}^{\mathrm{i}} = \mathbf{M}_s^i \vec{\mathrm{P}}^{\mathrm{s}}$$

This transformation is expressed as a 3x4 matrix:

$$\mathrm{M}_{\mathrm{s}}^{\mathrm{i}} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix}$$

composed from 3 transformations between 4 reference frames.
When expressed in homogeneous coordinates, these transformations are composed as matrix multiplications.

$$\vec{Q} = M_s^i \vec{P} = C_r^i P_c^r T_s^c \vec{P}$$

We will use "tensor notation" to keep track of our reference frames.

1.1 Homogeneous Coordinates and Tensor Notation

Homogeneous coordinates allow us to express translation, rotation, scaling, and projection all as matrix operations. The principle is to add an extra dimension to each vector.

For example, points on a plane are expressed as:

$$\vec{\mathrm{P}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Similarly, points in 3D space become

$$\vec{\mathrm{Q}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

The line equation, ax+by+c=0 can be expressed as a simple product:

$$\vec{L}^T\vec{P} = \begin{pmatrix} a & b & c \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0 \qquad \text{where} \qquad \vec{\mathrm{L}}^{\mathrm{T}} = \begin{pmatrix} a & b & c \end{pmatrix}$$

Similarly, for a plane equation: *ax+by+cz+d=1:*

$$\vec{S}^T\vec{P} = \begin{pmatrix} a & b & c & d \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = 0 \qquad \text{where} \qquad \vec{\mathrm{S}} = \begin{pmatrix} a & b & c & d \end{pmatrix}$$

Note that in Homogeneous coordinates, all scalar multiplications are equivalent.

$$a\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = b\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Any vector can be expressed in "canonical" form by dividing by the last coefficient

$$\begin{pmatrix} ax \\ ay \\ a \end{pmatrix} = \begin{pmatrix} ax/a \\ ay/a \\ a/a \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Tensor Notation:

In tensor notation, the sign " $\vec{}$ " is remplaced by subscripts and superscripts.
A super-script signifies a column vector.
For example the point $\vec{P}$ is P^i

$$P^i = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

The line $\vec{L}^T$ is $L_i = (l_1, l_2, l_3)$

The indices can be used to indicate the refence frame. For example: "i" for image.

Our camera model has the form of a 3 x 4 matrix

$$M_s^i = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & 1 \end{pmatrix}$$

such that

$$\begin{pmatrix} q_1/q_3 \\ q_2/q_3 \\ 1 \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \vec{Q} = M_s^i \vec{P} = M_s^i = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & 1 \end{pmatrix} \begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix}$$

The sub/super scripts indicate the source and destination reference frames.
M_s^i is a transformation from "S" (Scene) to "i" (image).
The matrix M_s^i is a line of columns or a column of rows

Einstein summation convention:

The summation symbol is implicit when a superscript and subscript have the same letter.

$$L_i P^i = l_1 p^1 + l_2 p^2 + l_3 p^3$$

for a matrix and vector, this gives a new vector:

$$P^j = T_i^j P^i$$

This example transforms the point in reference i to a point in reference j.

Similarly $\vec{Q}^i = M_s^i \vec{P}^s$ evaluates to a column vector by summation of rows

$$\vec{Q}^i = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} m_1^1 & m_2^1 & m_3^1 & m_4^1 \\ m_1^2 & m_2^2 & m_3^2 & m_4^2 \\ m_1^3 & m_2^3 & m_3^3 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = M_s^i \vec{P}^s$$

or

$$\begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = \begin{pmatrix} wi \\ wj \\ w \end{pmatrix} = \begin{pmatrix} m_1^1 & m_2^1 & m_3^1 & m_4^1 \\ m_1^2 & m_2^2 & m_3^2 & m_4^2 \\ m_1^3 & m_2^3 & m_3^3 & 1 \end{pmatrix} \begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix}$$

$$i = \frac{wi}{w} = \frac{m_1^1 x_s + m_2^1 y_s + m_3^1 z_s + m_4^1}{m_1^3 x_s + m_2^3 y_s + m_3^3 z_s + 1}$$

$$j = \frac{wj}{w} = \frac{m_1^2 x_s + m_2^2 y_s + m_3^2 z_s + m_4^2}{m_1^3 x_s + m_2^3 y_s + m_3^3 z_s + 1}$$

Notice that this gives 11 coefficients.

<u>Extrinsic and Intrinsic camera parameters:</u>

The camera model can be expressed as a function of 11 parameters.
These are often separated into 6 "extrinsic" parameters and 5 "intrinsic" parameters:
Thus the "Extrinsic" parameters of the camera describe the camera position and orientation in the scene. These are the six parameters:

Extrinsic Parameters = (x, y, z, θ, φ, γ)

The intrinsic camera parameters express the projection to the retina, and the mapping to the image. These are :

F : The "focal" length
C_x, C_y : the image center (expressed in pixels).
D_x, D_y : The size of pixels expressed in pixels/mm.

This transformation can be decomposed into 3 basic transformations between 4 reference frames. The reference frames are:

Coordinate Systems:

Scene Coordinates:

Point Scène: $P^s = (x_S, y_S, z_S, 1)^T$

Camera Coordinates:

external world: $P^c = (x_C, y_C, z_C, 1)^T$

Retina: $Q^r = (x_r, y_r, 1)^T$

Image Coordinates

Image: $Q^i = (i, j, 1)^T$

The transformations are represented by Homogeneous projective transformations.

$$\vec{Q}^i = C_r^i \vec{Q}^r \qquad \vec{Q}^r = P_c^r \vec{P}^c \qquad \vec{P}^c = T_s^c \vec{P}^s$$

These express
1) A translation/rotation from scene to camera coordinates: T_s^c
2) A projection from scene points in camera coordinates to the retina: P_c^r
3) Sampling scan and A/D conversion of the retina to give an image: C_r^i

1.2 Coordinate Transforms : From Scene to Camera Coordiantes

$$P^c = T_s^c P^s$$

Homogeneous coordinates makes it possible to express translations, rotations, scalings, similarity transforms, affine transforms and perspective projection as matrix multiplication.

For example, the following matrix represents a translation Δx, Δy, Δz and a rotation R.

$$T_s^c = \begin{pmatrix} & & & \Delta x \\ & R & & \Delta y \\ & & & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

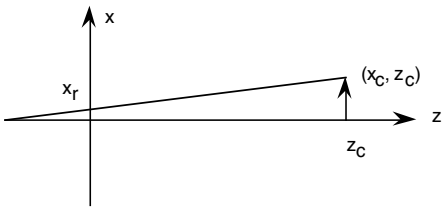
The transformation is composed by expressing the position of the source reference frame in the destination reference frame.

The rotation part is a 3x3 matrix that can be decomposed into 3 smaller rotations.

$$\mathbf{R} = \mathbf{R}_z(\gamma)\mathbf{R}_y(\varphi)\mathbf{R}_x(\theta)$$

1.3 Projective Transforms: from the scene to the retina

Projection through an aperture is a projective transformation
Consider the central projection model for a 1D camera:



In camera coordinates:

$P^c = (x_c, y_c, z_c, 1)$ is a scene point in camera (aperture centered) coordinates
$Q^r = (x_r, y_r, 1)$ is a point on the retina.

By similar triangles:

$$\frac{x_r}{F} = \frac{x_c}{z_c} \qquad \Rightarrow x_r = \frac{F}{z_c} x_c \quad \Rightarrow \frac{z_c}{F} x_r = x_c \quad \Rightarrow w x_r = x_c$$

$$\frac{y_r}{F} = \frac{y_c}{z_c} \qquad \Rightarrow y_r = \frac{F}{z_c} y_c \quad \Rightarrow \frac{z_c}{F} y_r = y_c \quad \Rightarrow w y_r = y_c$$

where $w = \frac{z_c}{F}$

As a matrix:
$$\begin{pmatrix} w x_r \\ w y_r \\ w \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{F} & 0 \end{pmatrix} \begin{pmatrix} y_c \\ y_c \\ y_c \\ 1 \end{pmatrix}$$

from which $x_r = \frac{w x_r}{w}$ and $y_r = \frac{w y_r}{w}$

Expressed in tensor notation, the transformation from Scene points in camera coordinates to retina points is:

$$Q^r = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{F} & 0 \end{pmatrix} \begin{pmatrix} p^1 \\ p^2 \\ p^3 \\ 1 \end{pmatrix} = P_c^r \vec{P}^c$$

and $\begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix} = \begin{pmatrix} q_1/q_3 \\ q_2/q_3 \\ 1 \end{pmatrix}$

thus:

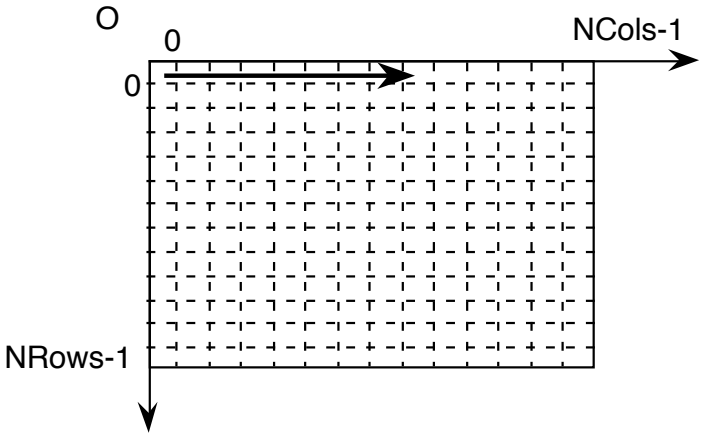
$$P_c^r = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{F} & 0 \end{pmatrix}$$

Note that P_c^r is not invertable.

1.4 From the Retina to Digitized Image

The "intrinsic parameters of the camera are F and C_x, C_y, D_x, D_y

The image frame is composed of pixels (picture elements)



Note that pixels are not necessarily square.

Typical image sizes 1024 x 768, 512x512

The mapping from retina to image can be expressed with 4 parameters:

C_x, C_y : the image center (expressed in pixels).
D_x, D_y : The size of pixels expressed in pixels/mm.

$i = x_r D_i$ (mm · pixel/mm) + C_i (pixel)

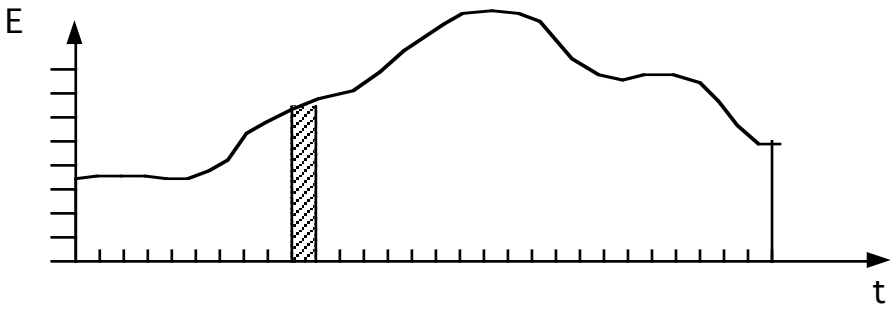
$j = y_r D_j$ (mm · pixel/mm) + C_j (pixel)

Transformation from retina to image :

$$Q^i = \mathbf{C}_r^i \; Q^r$$

$$\begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = \begin{pmatrix} wi \\ wj \\ w \end{pmatrix} = \begin{pmatrix} D_i & 0 & C_i \\ 0 & D_j & C_j \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix}$$

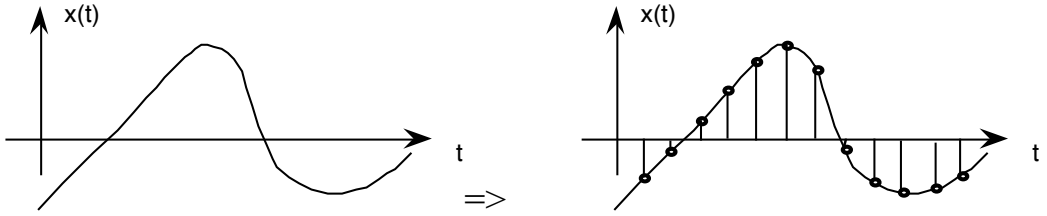
Sampling and A/D Conversion.



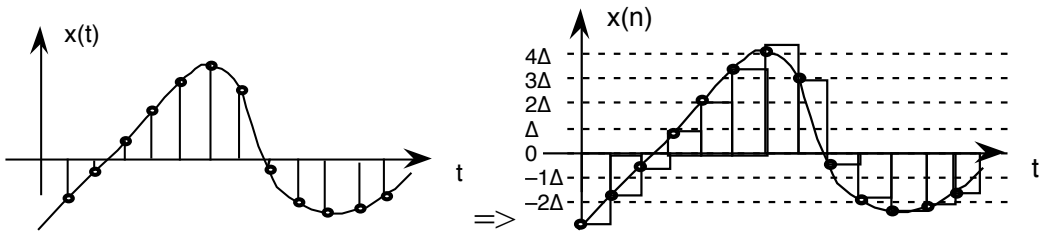
The transformation from Retina to Image adds two kinds of noise:

Sampling Noise (or aliasing)

Sampling:

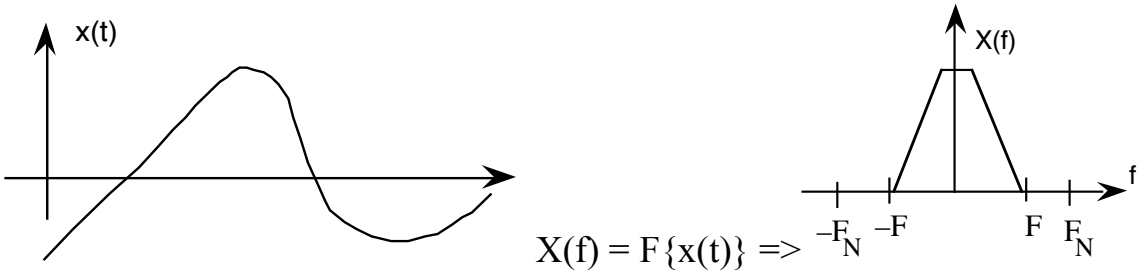


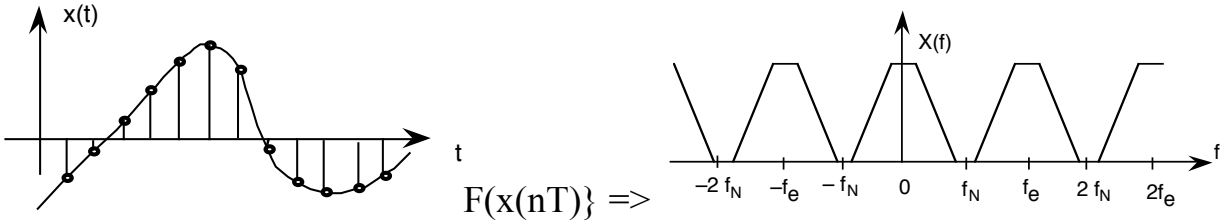
Quantification:



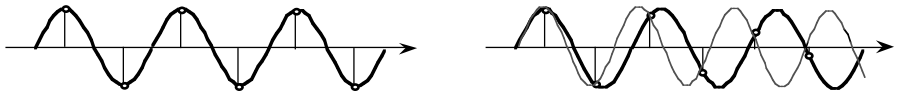
Sampling Noise:

Suppose that our analog signal has the following spectrum:



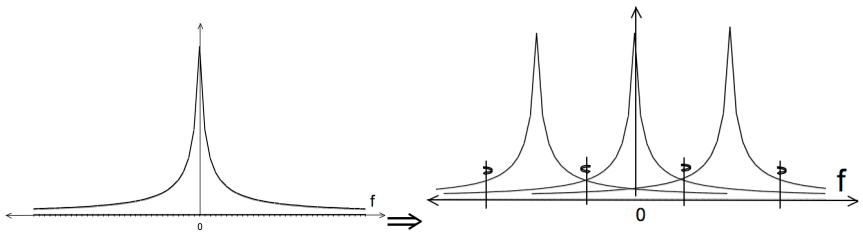


Multiple Cosines become indistinguishable (aliased).



$$F \leq f_N = \frac{f_e}{2}$$

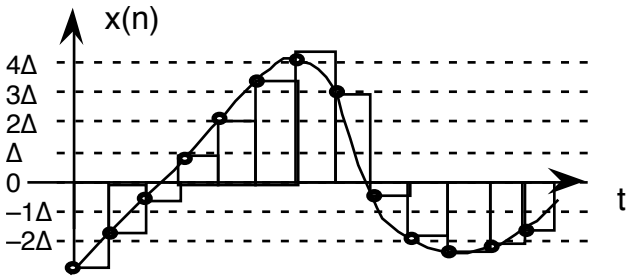
In reality, the spectrum can not be zero for more than an isolated point.



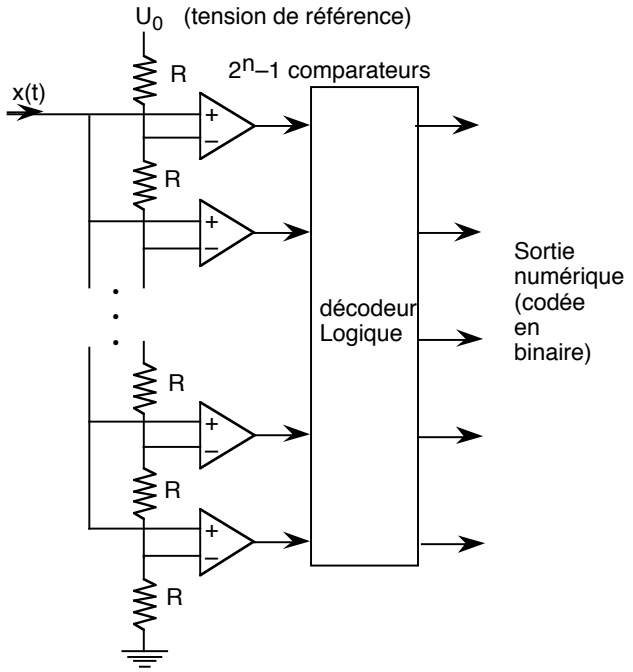
Sampling adds repeated copies of the spectrum at periods of two times the nyquist frequency $2F_n = 2/T$.

Thus, when a signal s(x) is sampled high frequency (local) sampling noise is necessarily introduced

Quantification Noise



A/D Conversion:



Teh signal is expressed in $2^B = \frac{V}{q}$ intervals of size q, by

$$x_q(n) = \underset{k}{\text{Max}} \{(x_e(nT_e) - k \cdot q + \frac{q}{2}) < 0\}$$

If the variance (energy) of the original signal is σ_x^2, the quantized signal has a logarithmic signal to noise ratio of

$$\xi_{qdB} = 6B + 10.8 - 20 \log_{10} \frac{V}{\sigma_x} \text{ dB}$$

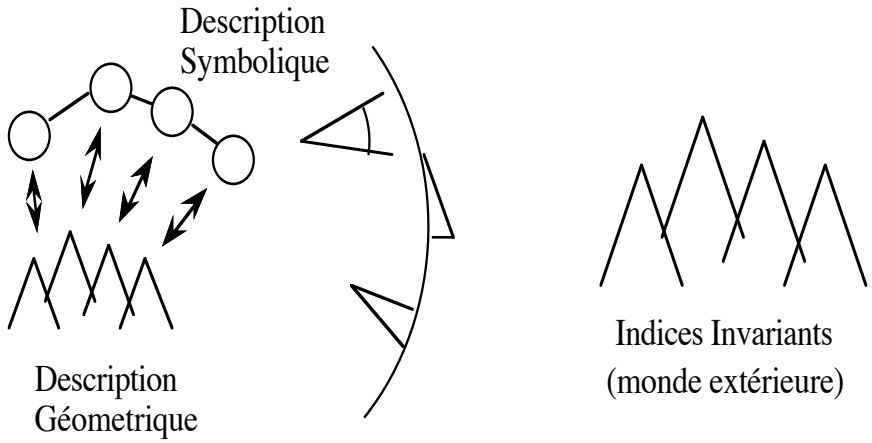
Lesson: Each extra bit DOUBLES the cost, and halves the noise.

Common practice : Digitize to 10 bits per color , save 8 bits per color after "processing" => Image is RGB (3 x 8 bits).

2 Describing Contrast

An image is a large table of numerical values (pixels).
The "information" in the image may be found in the colors of regions of pixels, and the variations in intensity of pixels (contrast).

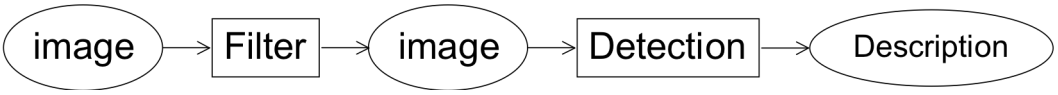
Extracting information from an image requires organizing these values into patterns that are "invariant" to changes in illumination and viewing direction.



Color provides information about regions of constant pigment.
Contrast provides information about 3D shape, as well as surface markings.

Contours of high contrast are referred to as "edges".

Edge detection is typically organized in two steps
1) contrast filtering
2) edge point detection and linking.



Two classic contrast detection operators are:

1) Roberts Cross Operator, and
2) The Sobel edge detector.

2.1 **Roberts Cross Edge Detector**

One of the earliest methods fore detecting image contrast (edges) was proposed by Larry Roberts in his 1962 Stanford Thesis.

Note, in this same thesis, Roberts introduced the use of homogeneous coordinates for camera models, as well as wire frame scene models. Roberts subsequently went to work for DARPA where he managed the program that created the Arpanet (now known as the internet).

Roberts Cross operator employs two simple image filters:

$$m_1(i,j) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad m_2(i,j) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

These two operators are used as filters. They are convolved with the image.

Convolution (or filtering) : for n = 1, 2

$$E_n(i, j) = m_n * p(i,j) = \sum_{k=0}^{1}\sum_{l=0}^{1} m_n(k,l)\,p(i-k, j-l)$$

The contrast is the module of each pixel :

$$E(i,j) = \left\|\vec{E}(i,j)\right\| = \sqrt{E_1(i,j)^2 + E_2(i,j)^2}$$

The direction of maximum contrast is the phase

$$\varphi(i,j) = Tan^{-1}\left(\frac{E_2(i,j)}{E_1(i,j)}\right) + \frac{\pi}{4}$$

Because of its small size and simplicity, the Roberts detector is VERY sensitive to high spatial-frequency noise. This is exactly the noise that is most present in images.

To reduce such noise, it is necessary to "smooth" the image with a low pass filter.
We can better understand the Roberts operators by looking at their Fourier Transform.

$$M_n(u,v) = \sum_{k=0}^{1}\sum_{l=0}^{1} m_n(k,l)e^{-j(ku+lv)}$$

$$M_1(u,v) = (+1)\cdot e^{-j(-(0)u+-(0)v)} + (-1)\cdot e^{-j(u+v)} = 2j\,Sin(0.5u+0.5v)$$

$$M_2(u,v) = (+1)\cdot e^{-j(-(0)u+-(1)v)} + (-1)\cdot e^{-j(-(1)u+-(0)v)} = 2j\,Sin(0.5u-0.5v)$$

2.2 The Sobel Detector

Invented by Irwin Sobel in his 1964 Doctoral thesis, this edge detector was made famous by the the text book of R. Duda adn P. Hart published in 1972.

It is perhaps the most famous and widely used edge detector:

$$m_1(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad m_2(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Convolution (or filtering) : for n = 1, 2

$$E_n(i, j) = m_n * p(i,j) = \sum_{k=-1}^{1}\sum_{l=-1}^{1} m_n(k,l)\,p(i-k, j-l)$$

The contrast is the module of each pixel :

$$E(i,j) = \left\|\vec{E}(i,j)\right\| = \sqrt{E_1(i,j)^2 + E_2(i,j)^2}$$

The direction of maximum contrast is the phase

$$\varphi(i,j) = Tan^{-1}\left(\frac{E_2(i,j)}{E_1(i,j)}\right)$$

Sobel's Edge Filters can be seen as a composition of a image derivative and a smoothing filter.

$$m_1(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

$$m_2(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

The filter $\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ is a form of image derivative.

The filter $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ is a binomial smoothing filter.

2.3 Difference Operators: Derivatives for Sampled Signals

For the function, s(x) the derivative can be defined as :

$$\frac{\partial s(x)}{\partial x} = \lim_{\Delta x \to 0}\left\{\frac{s(x+\Delta x) - s(x)}{\Delta x}\right\}$$

For a sampled signal, s(n), an the equivalent is $\frac{\Delta s(n)}{\Delta n}$

the limit does not exist, however we can observe

$$\Delta n = 1: \quad \frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n)}{1} = s(n) * [-1 \quad 1]$$

$$\Delta n = 0: \quad \frac{\Delta s(n)}{\Delta n} = \frac{0}{0}$$

This is the operator used by Roberts.

If we use a Symmetric definition for the derivative:

$$\frac{\partial s(x)}{\partial x} = \lim_{\Delta x \to 0}\left\{\frac{s(x+\Delta x) - s(x-\Delta x)}{\Delta x}\right\}$$

then

$$\Delta n = 1: \quad \frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n-1)}{1} = s(n) * [-1 \quad 0 \quad 1]$$

This is the operator used by Sobel.

Note that a derivative is equivalent to convolution!
We can define derivation in the fourier domain as follows:

$$F\left\{\frac{\partial s(x)}{\partial x}\right\} = -j\omega \cdot F\{s(x)\}$$

and thus

$$\frac{\partial s(x)}{\partial x} = F^{-1}\{-j\omega\} * s(x)$$

If we can determine d(x) = $F^{-1}\{-j\omega\}$ then we have our derivative operator. If we "sample" d(x) to produce d(n) we have a sampled derivative operator.

Unfortunately, $F^{-1}\{-j\omega\}$ has an infinite duration in x, and thus d(n) is an infinite series. However, the first term of d(n) is [-1 0 1].

Thus we can define the first "difference" operator as a first order approximation for the derivative of a discrete signal.

$$\Delta_i p(i,j) = \Delta p(i,j)/\Delta i = p(i,j) * [-1, 0, 1]$$

$$\Delta_i p(i,j) = \frac{\Delta p(i,j)}{\Delta i} = p(i,j) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\Delta_j p(i,j) = \frac{\Delta p(i,j)}{\Delta j} = p(i,j) * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

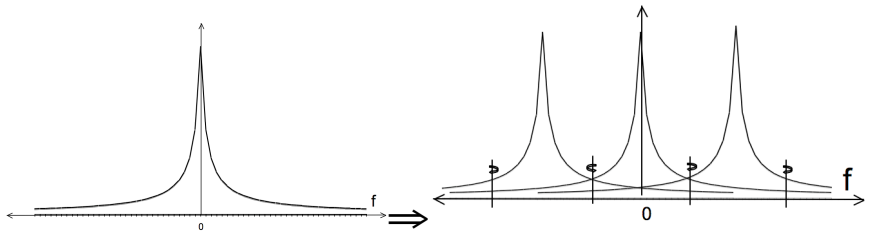
$$\nabla P(i,j) = \begin{pmatrix} \Delta_i p(i,j) \\ \Delta_j p(i,j) \end{pmatrix}$$

$$E(i,j) = \left\| \nabla P(i,j) \right\|$$

$$\vartheta(i,j) = Tan^{-1}\left(\frac{\Delta_j p(i,j)}{\Delta_i p(i,j)} \right)$$

This works fine, except that such a derivative operator amplifies sampling noise.

When a signal s(x) is sampled to create s(n), sampling noise is introduced



Sampling adds repeated copies of the spectrum at periods of two times the nyquist frequency $2F_n = 2/T$. The result amplifies high frequency noise.

The first difference filter $d_1(n) = [1, 0, -1]$ has a Fourier transform:

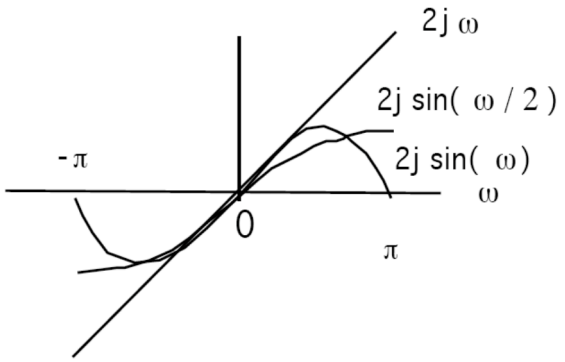
$$D(\omega) = \sum_{n=-1}^{1} d(n) e^{-j\omega n}$$
$$D(\omega) = 1e^{-j\omega(-1)} + 0e^{-j\omega 0} + (-1)e^{-j\omega(1)}$$
$$D(\omega) = e^{j\omega} - e^{-j\omega}$$
$$D(\omega) = -2j\sin(\omega)$$

Calculation of a derivative is the same as convolution with the filter [1, 0, -1], which is the same as multiplication of the spectrums.

$$d(n) * s(n) \quad \Leftrightarrow D(\omega) \cdot S(\omega)$$

The filter d(n) = [1, -1] is even worse. Its Fourier transform is

$$D(\omega) = -2j\sin(\omega/2)$$



Sobel uses an "optimal" local derivative filter : [1, 0, -1]

2.4 Smoothing: The Binomial Low pass filter.

Sobel uses a filter [1, 2, 1] to smooth. This is also an optimal filter.
It is part of a family of filters generated by the binomial series.

The binomial series is the series of coefficients of the polynomial:

$$(x + y)^n = \sum_{m=0}^{n} b_{m,n} x^{n-m} y^m$$

The coefficients can be computed as $b_{m,n} = b_n(m) = [1, 1]^n$

These are the coefficients of Pascal's Triangle.

n	sum = 2^n	$\mu = n/2$	$\sigma^2 = n/4$	$\sigma = \sqrt{n/2}$	Coefficients
0	1	0	0	0	1
1	2	0.5	0.25		1 1
2	4	1	0/5		1 2 1
3	8	1.5	0.75		1 3 3 1
4	16	2	1	1	1 4 6 4 1
5	32	2.5	1.25		1 5 10 10 5 1
6	64	3	1.5		1 6 15 20 15 1
7	128	3.5	1.75		1 7 21 35 35 21 7 1
8	256	4	2	$\sqrt{2}$	1 8 29 56 70 56 29 8 1

These coefficients provide a family of low pass filters with remarkable properties.
Notably, these are the best approximation for a Gaussian filter of finite extent.
They also happen to have integer coefficients.

$b_n(m) = b_1(m)^{*n} = [1, 1]^{*n}$ = n convolutions of [1, 1]

Gain : $s_n = \sum_{m=1}^{n} b_n(m) = 2^n$

Center of gravity is

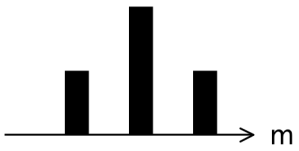
$$\mu_n = \frac{1}{s_n} \sum_{m=1}^{n} b_n(m) \cdot m = \frac{n}{2}$$

The variance is:

$$\sigma_n^2 = \frac{1}{s_n} \sum_{m=1}^{n} b_n(m) \cdot (m - \mu)^2 = \frac{n}{4}$$

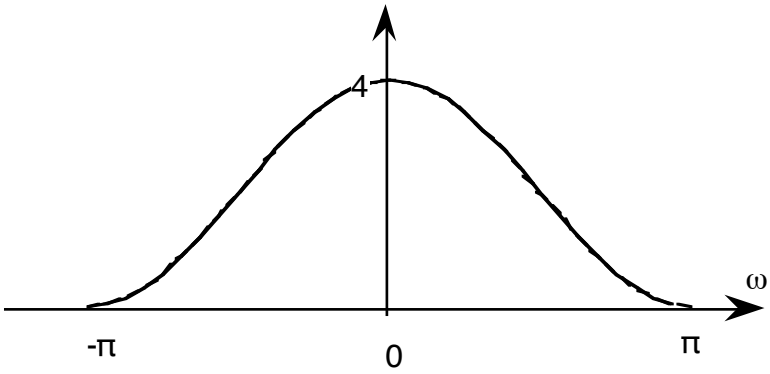
Consider $b_2(m)$:

$$b_2(m) = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$



The Fourier transform for $b_2(m) = [1, 2, 1]$ is

$$B_2(\omega) = \sum_{m=-1}^{1} b_2(m) e^{-j\omega m}$$
$$B_2(\omega) = 1e^{-j\omega(-1)} + 2e^{-j\omega 0} + 1e^{-j\omega(1)}$$
$$B_2(\omega) = 2 + e^{j\omega} + e^{-j\omega}$$
$$B_2(\omega) = 2 + 2\cos(\omega)$$



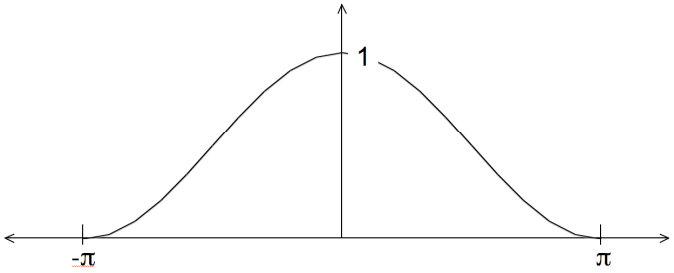
This is a cosine on a platform

If we normalize the gain to 1:

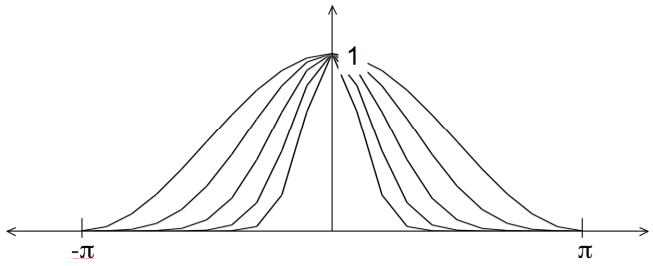
$b_2(m) = (1/4)[1, 2, 1]$

Then we obtain an low pass filter with no ripples

$$B_2(\omega) = \frac{1}{2} + \frac{1}{2}\cos(\omega)$$



Repeated convolution makes generates



$$b_{2n}(m) = b_n(m) * b_n(m)$$

$$B_{2n}(\omega) = B_n(\omega) \cdot B_n(\omega) = \left(\frac{1}{2} + \frac{1}{2}\cos(\omega)\right)^n$$

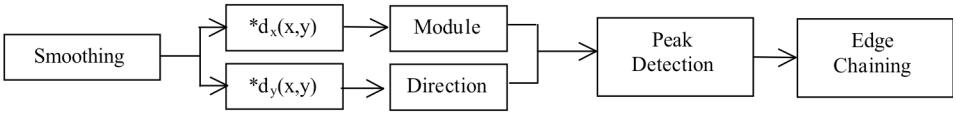
$$B_n(\omega) = \left(\frac{1}{2} + \frac{1}{2}\cos(\omega)\right)^{n/2}$$

The binomial coefficients provide a series of low pass filters with no ripples.

In 2D, the filters provide separable filters that are nearly circularly symmetric

$$\text{2-D} \quad b2(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \boxed{1\ 2\ 1}$$

2.5 Edge Detection using integer coefficient filters:



$$\nabla P(i,j) = \begin{bmatrix} \frac{\Delta p(i,j)}{\Delta i} \\ \frac{\Delta p(i,j)}{\Delta j} \end{bmatrix} = \begin{bmatrix} m_{1*}p(i,j) \\ m_{2*}p(i,j) \end{bmatrix} = \begin{bmatrix} E_1(i,j) \\ E_2(i,j) \end{bmatrix}$$

Gradient:

$$E(i,j) = \left\| \vec{E}(i,j) \right\| = \sqrt{E_1(i,j)^2 + E_2(i,j)^2}$$

Direction of maximum contrast

$$\varphi(i,j) = Tan^{-1}\left(\frac{E_2(i,j)}{E_1(i,j)}\right)$$

Steps:
1) Smoothing - Suppress high frequency noise
2) Gradient - Compute first derivatives in row and column
3) Detection - Non-maximum suppression with double threshold
4) Chaining - Assembly of connected points above threshold. Elimination of chains where one of the points is not above a second threshold.
5) Polygonal approximation. (multiple algorithms exist).

2.6 Non-maximum suppression.

Contrast points are local maxima in E(i, j).

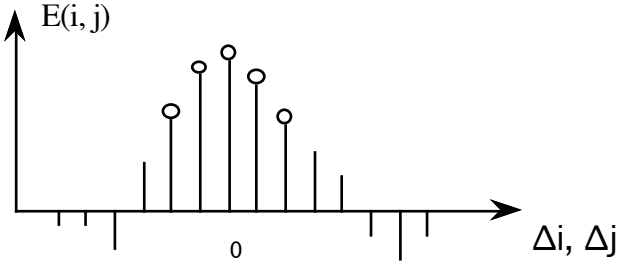
les points de contraste : C(i, j)
pour le gradient de la magnitude E(i, j) et orientation Φ(i, j)

For each point :

1) Determine the direction of maximum gradient:

$$\Delta i = \frac{\Delta_i P(i,j)}{\|\nabla P(i,j)\|} \qquad \Delta j = \frac{\Delta j P(i,j)}{\|\nabla P(i,j)\|}$$

2) Compare the gradient to its neighbors in this direction.



$$c(i,j) = \begin{cases} E(\mathrm{i},\mathrm{j}) \text{ if } \mathrm{E}(\mathrm{i}-\Delta\mathrm{i},\mathrm{j}-\Delta\mathrm{j}) \le \mathrm{E}(\mathrm{i},\mathrm{j}) \ge \mathrm{E}(\mathrm{i}+\Delta\mathrm{i},\mathrm{j}+\Delta\mathrm{j}) \\ 0 \text{ Otherwise} \end{cases}$$

Construct a list of connected points for which E(i,j) ≠ 0.

Techniques:

1) Line scan edge chaining algorithm
2) Edge following
3) Hough Transform

3 The Hough Transform

The Hough transform is an "optimal" statistical detector for estimating parametric functions from discrete samples. This method was invented for interpreting bubble chamber images in particle physics. It is based on "voting" for possible parameters.

This transform was invented by
P.V.C. Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

It was patented in a crude form by IBM in 1962 using $y = mx+c$.

It was made popular by Duda and Hart :
Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11–15 (January, 1972)

Consider the line equation

$$x \cos(\theta) + y \sin(\theta) + c = 0$$

In the image, for each x,y (free parameters) we need to determine (c, θ)

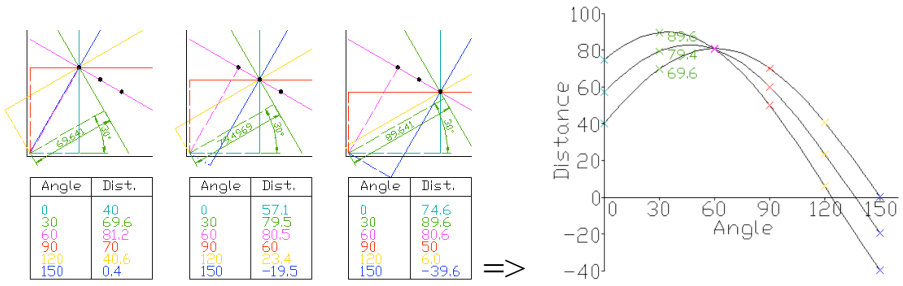
In the Hough transform, we will create a dual space in which (c, θ) are free parameters.
We will estimate lines as peaks in this dual space. To find peaks we build an accumulator array : $h(c, \theta)$.

Let the c be an integer $c \in [0, D]$ where D is the "diagonal distance of the image.
Let θ be an integer $\theta \in [0, 179]$

Algorithm:

```
allocate a table  h(c, θ) initially set to 0.
For each x, y of the image
        for  θ from 0 to 179
                c = –x cos(θ) – ysin(θ)
                h(c, θ) = h(c, θ) + E(x, y)
        End
End
```

The resulting table accumulates contrast.
Peaks in $h(c, \theta)$ correspond to line segments in the image.



Because we know θ(x, y), we can limit the evaluation to θ(x, y)+/- Δθ

3.1 Generalisation of the Hough Transform

We can represent a circle with the equation:

$$(x - a)^2 + (y - b)^2 = r^2$$

We can use this to create a Hough space h(a, b, r) for limited ranges of r.

The ranges of a and b are the possible positions of circles.

Algorithm

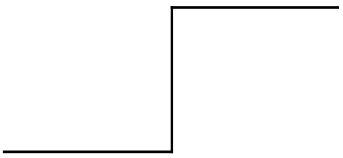
Algorithm:

allocate a table h((a, b, r) initially set to 0.
For each x, y of the image
for r from r_{min} to r_{max}
for a from 0 to a_{max}
b = –y–sqrt(r^2 – (x - a)^2)
h(a,b,r) = h(a,b,r) + E(x,y).
End
End
End

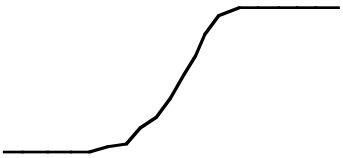
4 Second Derivatives.

An alternative to the gradient is to detect edges as zero crossings in the second derivative.

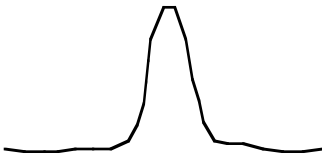
Contrast :



Smoothing :



1st derivative



Second derivative

The second derivative is a form of Laplacian operator:

$$\text{Laplacien: } \nabla^2 p(i,j) = \frac{\Delta^2 P(i,j)}{\Delta i^2} + \frac{\Delta^2 P(i,j)}{\Delta j^2}$$

$$\Delta_i^2 = [\ 1\ \ -1] * [1\ \ -1] = [\ -1\ \ 2\ \ -1]$$

$$\Delta_i^2 = \boxed{-1\ 2\ -1} \qquad \Delta_j^2 = \boxed{\begin{matrix} -1 \\ 2 \\ -1 \end{matrix}}$$

Laplacian : $\nabla^2 p(i,j) = \Delta_i^2 * p(i,j) + \Delta_j^2 * p(i,j)$

There are several possible discrete forms:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} + \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

$L(u,v) = 4 - 2\cos(u) - 2\cos(v)$
The best is :

$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix} + 2 \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

Gradient : $\| \nabla p(i, j) \| = \sqrt{(\frac{\partial P(i,j)}{\partial i})2 + (\frac{\partial P(i,j)}{\partial j})2}$

Laplacien : $\nabla^2 p(i,j) = \frac{\partial^2 P(i,j)}{\partial i^2} + \frac{\partial^2 P(i,j)}{\partial j^2}$

4.1 Zero Crossings in the second derivative.

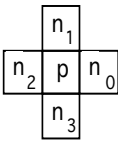
In theory

1) Zero crossings give closed contours
2) Zero crossings can be easily interpolated for high precision.

In practice

Zero crossings detect many small unstable contours.

Neigborhood test:



$$C(i,j) = \begin{cases} 1 \text{si } (\text{signe}(n_0) \neq \text{signe}(n_2)) \text{ et } |n_0\, n_2| > 0| \\ \text{ou } (\text{signe}(n_1) \neq \text{signe}(n_3)) \text{ et } |n_1\, n_3| > 0 \\ 0 \text{Sinon} \end{cases}$$

alternatively :

$$C(i,j) = \begin{cases} 1 \text{si } (\text{signe}(n_0) \neq \text{signe}(n_2)) \\ \text{et } |n_0\, n_2| > 0| \\ \text{et } |n_0 - n_2| > \text{seuil} \\ \text{ou } (\text{signe}(n_1) \neq \text{signe}(n_3)) \\ \text{et } |n_1\, n_3| > 0 \\ \text{et } |n_1 - n_3| > \text{seuil} \\ 0 \text{Sinon} \end{cases}$$