Computer Vision

MSc Informatics option GVR

James L. Crowley

Fall Semester 23 October 2008

Lesson 3

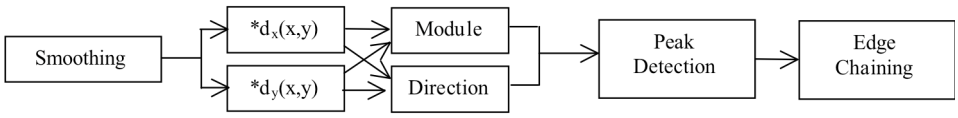
Describing Contrast with Gaussian Derivatives

Lesson Outline:

1 Describing Contrast (continued)

1.1 Edge Detection using integer coefficient filters:

The detection of edges (contours of high contrast) typically involves the following steps



Edges are detected as local maxima in the image gradient magnitude.
The gradient is

$$\vec{\nabla}P(i,j) = \begin{pmatrix} \frac{\Delta P(i,j)}{\Delta i} \\ \frac{\Delta P(i,j)}{\Delta j} \end{pmatrix} = \begin{pmatrix} m_1 * p(i,j) \\ m_2 * p(i,j) \end{pmatrix} = \begin{pmatrix} E_1(i,j) \\ E_2(i,j) \end{pmatrix}$$

The gradient magnitude is:

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

The direction of maximum gradient is:

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

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 local maxima whose grandient magnitgude is above a (usually fixed) threshold. Elimination of chains where one of the points is not above a second threshold.
5) Polygonal approximation. (Multiple algorithms exist: recursive Line fitting, leaste squares estimation, the Hough Transform).

For smoothing, it is necessary to specify a smoothing factor. This is typically specified as a smoothing "scale", given by the Standard Deviation for a Guassian smoothing window, σ. We will develop this more fully below.

As we saw in the last lecture, a fast smoothing can be obtained by repeated convolution with a 2D binomial filter.

$$\text{2-D}\quad b_2(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

From the last lecture, for n convolutions of [1,1].

$$\sigma^2 = \frac{n}{4} \quad \text{so}$$

for a desired σ, $n = 4\sigma^2$

We can greatly speed this up using pyramid methods. This will be developed below.

since [1,1] * [1,1] = [1,2,1]

n convolutions of [1,1] is n/2 convolutions of [1,2,1],
n convolutions of [1,1] is also n/4 convolutions of [1,4,6,4,1]

thus to obtain a smoothing with scale σ
it is sufficient to convolve the image with [1,4,6,4,1] k=n/4 = σ^2 times.

1.2 Non-maximum suppression.

Contrast points are local maxima in E(i, j) such that the gradient magnitude is above a threshold.

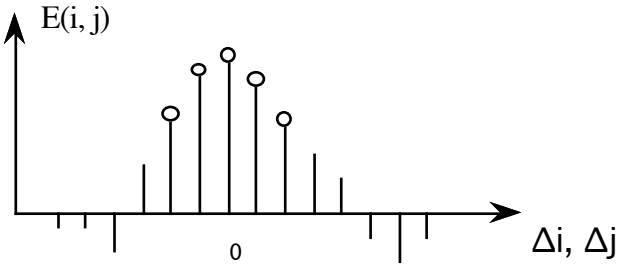
Contrast points : C(i, j) are determined from the gradient magnitude E(i, j) and orientation θ(i, j)

For each point :

1) Determine the direction of maximum gradient:

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

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



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

Because i,j are integers, it is necessary to round off Δi and Δj

Construct a list of connected points for which E(i,j) $\geq T_1$.
The threshold T_1 is generally dependent on the noise energy of the image.
In most cases, a value of $T_1 = 4$ is sufficient to eliminate most noise points.

Multiple algorithms exist to detect the points that are considered edge points:
1) Line scan edge chaining algorithm
2) Edge following

Alternatively we can estimate line equations with a Hough Transform.

2 The Hough Transform

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

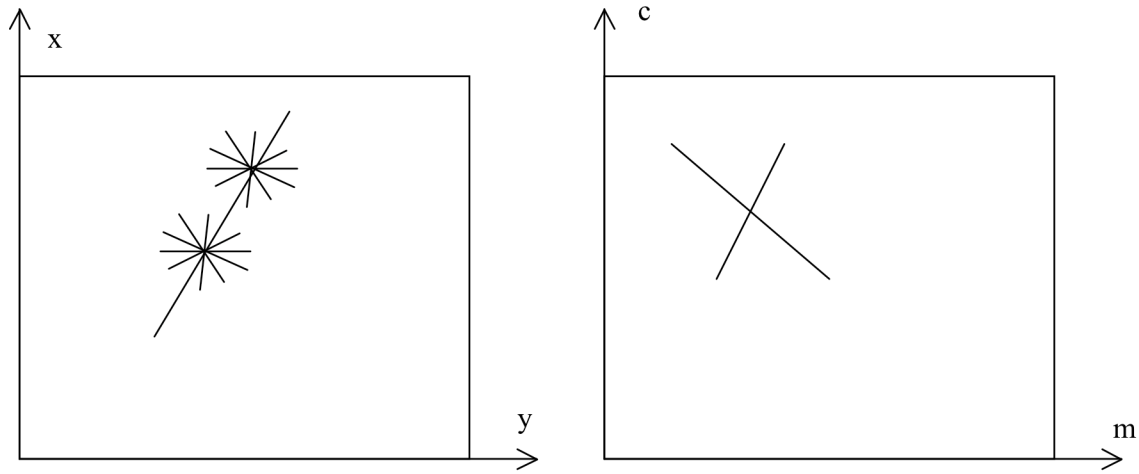
This transform was invented by
P.V.C. Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

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)

Earliest versions of the Hough transform used a line equation of $y=mx+c$.
This is easier to explain, but has problems for vertical lines.
A better line equation is $x \cos(\theta) + y \sin(\theta) + c = 0$

Consider the line equation $y=mx+c$.
Any point x,y belongs to an continuous manifold (continuous set) of lines m, c.
A point in x,y specifies a line in m,c
A point in m,c corresponds to a line in x,y



y=mx+c c= –mx+y

In the Hough transform, we will create a dual space in which (m,c) are the free parameters.

$$c = -mx+y$$

We will create an accumulator array h(m,c) for discrete values of (m,c).

(this poses the problem of how to "discretize" m, c.

For each point x, y, which is a local maxima in the gradient (for which c(x,y) ≠ 0), we will evaluate the set of possible values of m and c. For each possible value we will increment h(m,c).

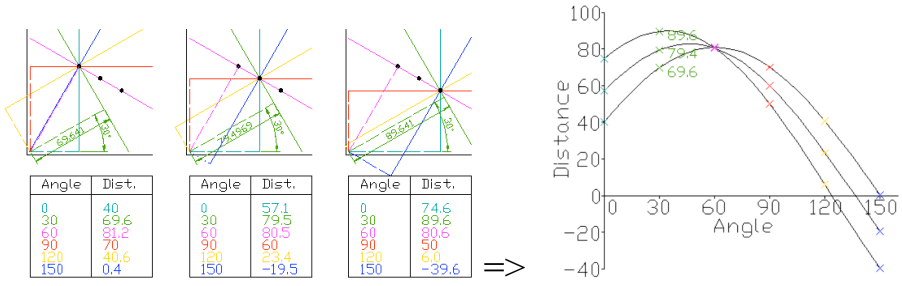
For all x, y
    if C(x, y) ≠ 0 then
    for all m
        c= –mx+y
        h(m,c) = h(m,c) + 1.

We will estimate lines as peaks in this dual space. To find peaks we build an accumulator array : h(c, θ).

Let the c be an integer c ∈ [0, D] where D is the "diagonal distance of the image.
Let θ be an integer θ ∈ [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, θ) correspond to line segments in the image.



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

2.1 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

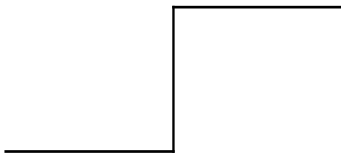
Algorithm:

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

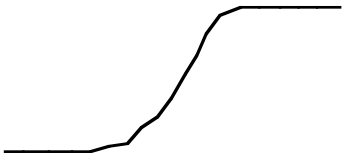
3 Second Derivatives.

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

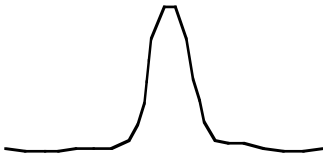
Contrast :



Smoothing :



1st derivative



Second derivative

The second derivative is a form of Laplacian operator:

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

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

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

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

3.1 Zero Crossings in the Second Derivative.

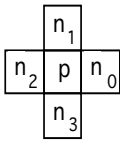
In theory

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:



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

alternatively :

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

4 Image Description Using Gaussian Derivatives

4.1 Gaussian Derivatives Operators

The Gaussian Function is $G(x,\sigma) = e^{-\frac{x^2}{2\sigma^2}}$

The Gaussian function is invariant to affine transformations.

$$T_a\{G(x, \sigma)\} = G(T_a\{x\}, T_a\{\sigma\})$$

Recall from lesson 2 we saw that $x_r = x_c \frac{F}{z_c}$

The apparent size of an object is inversely proportional to its distance

A change in size (or scale) is a special case of an affine transform:

$$T_S\{G(x, \sigma)\} = G(T_S\{x\}, T_S\{\sigma\}) = G(sx, s\sigma)$$

This is just one of the many interesting properties of the Gaussian function.

4.2 Gausian Derivative Operators

The Gaussian function and its derivatives are:

$$G(x,\sigma) = e^{-\frac{x^2}{2\sigma^2}}$$

$$G_x(x,\sigma) = \frac{\partial G(x,\sigma)}{\partial x} = -\frac{x}{\sigma^2} G(x,\sigma)$$

$$G_{xx}(x,\sigma) = \frac{\partial^2 G(x,\sigma)}{\partial x^2} = \frac{x^2 - \sigma^2}{\sigma^4} G(x,\sigma)$$

$$G_{xxx}(x,\sigma) = \frac{\partial^3 G(x,\sigma)}{\partial x^3} = -\frac{x^3 - x\sigma^2}{\sigma^6} G(x,\sigma)$$

When sampled and as filter, the Guassian function should have a unit "gain" (amplification).. The gain introduced by a filter is its integral:

$$A = \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx = \sqrt{2\pi}\sigma$$

4.3 Differential operators based on the Gaussian

For a 2D signal, the first and second order derivatives are the Gradient and the Laplacian.

The gradient of a 2D function is a vector: $\vec{\nabla} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix}$

The Laplacian is a scalar function : $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

For the 2D function: s(x,y) :

The Gradient is : $\vec{\nabla} s(x,y) = \begin{pmatrix} \frac{\partial s(x,y)}{\partial x} \\ \frac{\partial s(x,y)}{\partial y} \end{pmatrix}$

The laplacian is: $\nabla^2 s(x,y) = \frac{\partial^2 s(x,y)}{\partial x^2} + \frac{\partial^2 s(x,y)}{\partial y^2}$

if s(x,y) is convolved with G(x,y;σ): $s * G(x,y,\sigma) = \int_{u=-\infty}^{\infty} \int_{v=-\infty}^{\infty} s(u,v) G(x-u, y-v, \sigma) du dv$

The gradient can be evaluated as: $\vec{\nabla}(s * G(x,y,\sigma)) = \begin{pmatrix} \frac{\partial (s * G(x,y,\sigma))}{\partial x} \\ \frac{\partial (s * G(x,y,\sigma))}{\partial y} \end{pmatrix}$

The Laplacian can be evaluated as $\nabla^2 (s * G(x,y,\sigma)) = \frac{\partial^2 (s * G(x,y,\sigma))}{\partial x^2} + \frac{\partial^2 (s * G(x,y,\sigma))}{\partial y^2}$

4.4 Gaussian Digital Filters

Computers represent image as 2D sampled digitized signals. Because they are sampled, processing requires convolution with a sampled filter.

To obtain a 2D digital Gaussian filter we must sample the function at a rate of Δx, and Δy over some fine support of radius R.

G(x, σ)

Because we can control scale with σ, we can set the sample size to Δx=Δy=1.
This gives a sampled function

$$G(n,\sigma) = e^{-\frac{n^2}{2\sigma^2}}$$

To represent this in a computer we must also specify the finite support (number of samples), N.
We set N = 2R + 1.

This gives us 2 parameters to control:

1) The scale of the Gaussian $\sigma/\Delta x$
2) the size of the support N = 2R+1

Truncating a function to a finite support is equivalent to multiply by a window $W_N(n)$

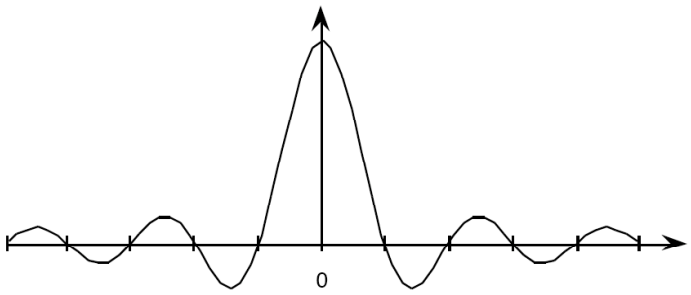
When we limit $G(x,\sigma)$ to a finite support, we multiply by a window

$G(n, \sigma) = G(n, \sigma) \cdot w_N(n)$ where $w_N(n) = \begin{cases} 1 & \text{for } -R \le n \le R \\ 0 & \text{otherwise} \end{cases}$

Multiplying by a finite window is equivalent to convolving with the fourier transform of the finite window:

$$G(\omega,\sigma) = G(\omega,\sigma) * W_N(\omega)$$

where $W_N(\omega) = \dfrac{\sin(\pi N/2)}{\sin(\pi/2)}$



For N < 7, the ripples in WN(w) dominate the spectrum and corrupt the resulting Gaussian.
At N=7 the effect is tolerable but significant.
At N≥ 9 the effect becomes minimal

In addition for $\sigma/\Delta x < 1$, the phenomenon of aliasing folds a significant amount of energy at the nyquist frequency, corrupting the quality (and the invariance) of the Gaussian function.

Finally, it is necessary to assure that the "gain" of the Gaussian filter is 1. This can be assured by normalizing so that the sum of the coefficients is 1. If the Gaussian were infinite in extent, then

$$\sum_{x=-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} = \sqrt{2\pi}\sigma$$

However, because we truncate the Gaussian to an extent n ±R, we must calculate the sum of the coefficients, A:

$$A = \sum_{n=-R}^{R} e^{-\frac{n^2}{2\sigma^2}}$$

The Gaussian filter is thus normalized by dividing by A to give a unit gain Receptive Field.

$$G(i, j, \sigma) = \frac{1}{A} e^{-\frac{(j^2 + j^2)}{2\sigma^2}}$$

4.5 The Sampled Gaussian Functions

The sampled Gaussian and its derivatives are:

$$G(n,\sigma) = e^{-\frac{n^2}{2\sigma^2}}$$

$$G_x(n,\sigma) = -\frac{n}{\sigma^2}G(x,\sigma) = -\frac{n}{\sigma^2}e^{-\frac{n^2}{2\sigma^2}}$$

$$G_{xx}(n,\sigma) = \frac{n^2-\sigma^2}{\sigma^4}G(n,\sigma) = \frac{n^2-\sigma^2}{\sigma^4}e^{-\frac{n^2}{2\sigma^2}}$$

$$G_{xxx}(n,\sigma) = -\frac{n^3-n\sigma^2}{\sigma^6}G(n,\sigma) = -\frac{n^3-n\sigma^2}{\sigma^6}e^{-\frac{n^2}{2\sigma^2}}$$

For the 2D Sampled Gaussian $G(i,j,\sigma) = e^{-\frac{i^2+j^2}{2\sigma^2}}$

(Attention: i, j are INTEGERS, not complex numbers!)

Continuous Gaussian Kernel: $G(x,y,\sigma) = e^{-\frac{(x^2+y^2)}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} * e^{-\frac{y^2}{2\sigma^2}}$

Discrete form: $G(i,j,\sigma) = e^{-\frac{(j^2+j^2)}{2\sigma^2}} = e^{-\frac{i^2}{2\sigma^2}} * e^{-\frac{j^2}{2\sigma^2}}$

Scale property $G(x,y,\sqrt{2}\sigma) = G(x,y,\sigma) * G(x,y,\sigma)$

Discrete form: $G(i,j,\sqrt{2}\sigma) = G(i,j,\sigma) * G(i,j,\sigma)$

Derivative Features:

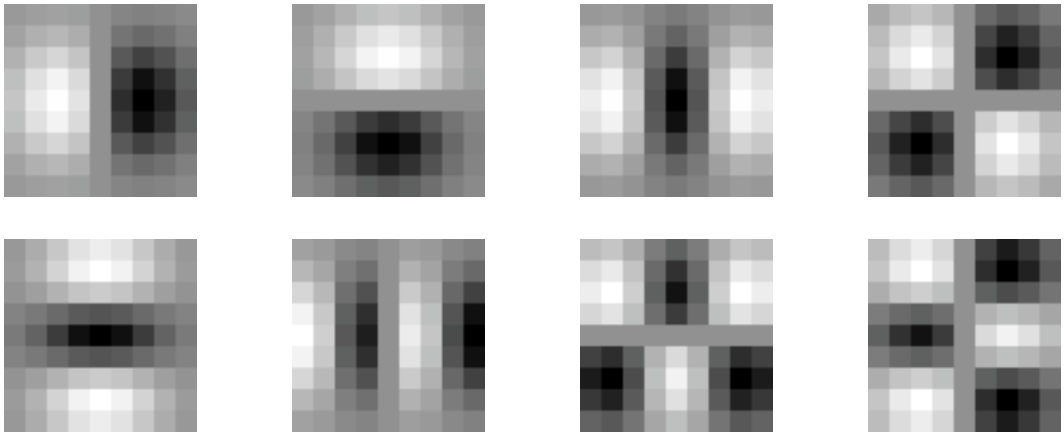
$$G_x(i,j,\sigma) = -\frac{i}{\sigma^2}G(i,j,\sigma)$$

$$G_{xx}(i,j,\sigma) = \frac{i-\sigma^2}{\sigma^4}G(i,j,\sigma)$$

$$\nabla^2 G_x(i,j,\sigma) = G_{xx}(i,j,\sigma) + G_{yy}(i,j,\sigma)$$

We can use these sampled functions to create a basis set of receptive fields:

$\vec{G}_a = (G_{x,} G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy})$



The Gaussian receptive fields $G_{x,} G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy}$.

Note that there is only one parameter: σ. This determines the limit of the resolution for the position of a contrast point.

4.6 Using the Gaussian to compute image derivatives

Consider an image: P(i, j),

The Gradient $\vec{\nabla}P(i,j)$ is calculated by $P(i,j)*\vec{\nabla}G(i,j,\sigma)$

where $\vec{\nabla}G(i,j,\sigma)=\begin{pmatrix}G_x(i,j,\sigma)\\G_y(i,j,\sigma)\end{pmatrix}$

Gradient: $\vec{\nabla}P(i,j)\approx\vec{\nabla}(P*G(i,j,\sigma))=P*\vec{\nabla}G(i,j,\sigma)=\begin{pmatrix}P*G_x(i,j,\sigma)\\P*G_y(i,j,\sigma)\end{pmatrix}$

Laplacien: $\nabla^2P(i,j)\approx P*\nabla^2G(i,j,\sigma)=\begin{pmatrix}P*\nabla^2G(i,j,\sigma)\\P*\nabla^2G(i,j,\sigma)\end{pmatrix}$

Note, you must specify σ!!! Many computer vision researchers neglect this and obtain unpredictable results in their experiments. Some times it works, some times it does not.

4.7 Steerability of Gaussian Derivatives.

For each pixel, one can calculate the orientation of maximal gradient. This orientation is equivariant with rotation. One can use this as an "intrinsic" orientation to normalize the receptive fields at any point in the image.

Local orientation: $\theta_i(x,y,\sigma)=Tan^{-1}(\frac{G_y\cdot P(x,y,\sigma)}{G_x\cdot P(x,y,\sigma)})$

It is possible to synthesize an oriented derivative at any point as a weighted sum of derivatives in perpendicular directions. The weights are given by sine and cosine functions.

$$G_1^{\theta}(x,y,\sigma)=\cos(\theta)\cdot G_x(x,y,\sigma)+\sin(\theta)\cdot G_y(x,y,\sigma)$$

By steering the Gaussian response to the local orientation, we obtain an "invariant" measure of local contrast:

Thus:

$$P_1(x,y;\theta,\sigma)=Cos(\theta)<G_x(x,y;\sigma)\cdot P(x,y)>+Sin(\theta)<G_y(x,y;\sigma)\cdot P(x,y)>$$

a similar measure can be obtained for the second and third derivatives:

$$G_2^{\theta}(x,y,\sigma)=Cos(\theta)^2G_{xx}(x,y,\sigma)+Cos(\theta)Sin(\theta)G_{xy}(x,y,\sigma)+Sin(\theta)^2G_{yy}(x,y,\sigma)$$
$$G_3^{\theta}(x,y,\sigma)=Cos(\theta)^3G_{xxx}(x,y,\sigma)+Cos(\theta)^2Sin(\theta)G_{xxy}(x,y,\sigma)+Cos(\theta)Sin(\theta)^2G_{xxy}(x,y,\sigma)+Sin(\theta)^3G_{yyy}(x,y,\sigma)$$

thus

$$P_{xx}(x,y;\theta,\sigma) = Cos(\theta)^2 < G_{xx}(x,y;\sigma)\cdot P(x,y) > + Sin(\theta)^2 < G_{yy}(x,y;\sigma)\cdot P(x,y) > \\ +2Cos(\theta)Sin(\theta) < G_{xy}(x,y;\sigma)\cdot P(x,y) >$$

and

$$P_{xxx}(x,y,\sigma,\theta) = Cos(\theta)^3 < G_{xxx}(x,y,\sigma) + Cos(\theta)^2 Sin(\theta) G_{xxy}(x,y,\sigma) + Cos(\theta)Sin(\theta)^2 G_{xxy}(x,y,\sigma) + Sin(\theta)^3 G_{yyy}(x,$$