Computer Vision

James L. Crowley

M2R MoSIG option GVR — Fall Semester

18 October 2012

Lesson 4

Describing Local Appearance with Derivatives

Lesson Outline:

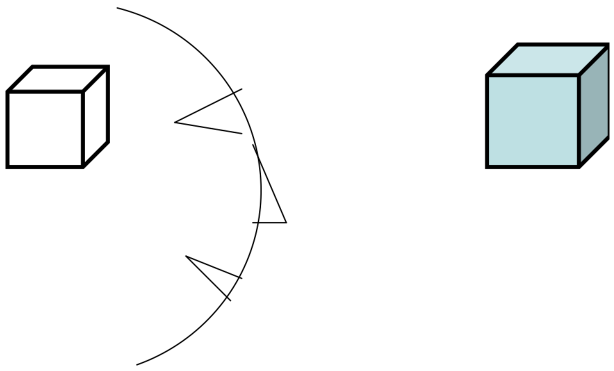
1 Describing Image Contrast with Derivatives

An image is simply 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 (chrominance) provides information about regions of constant pigment.
Contrast (change in intensity) provides information about 3D shape.
Contours of high contrast are referred to as edges.

Objects composed of planar surfaces (polyhedric objects) have straight line edges. Such edges are visual invariants to view angle. For this reason, edges (straight edge contours) became popular as an invariant image description during the 1960s and 1970s.



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

$$\begin{tikzcd}
\text{image} \arrow[r] & \boxed{\text{Filter}} \arrow[r] & \text{image} \arrow[r] & \boxed{\text{Detection}} \arrow[r] & \text{Edge Description}
\end{tikzcd}$$

A classic contrast detection operator is the Sobel edge detector.
Modern approaches use Gaussian Derivatives.

1.1 The Sobel Edge Detector

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

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

$m_c(i,j)$: detects contrast in column direction. *$m_r(i,j)$* : detects contrast in row direction

Convolution (or filtering) gives an edge vector: for n = r, c (row or column)

$$E_n(i,j) = m_n * p(i,j) \sum_{k=-1}^{1} \sum_{l=-1}^{1} m_n(k,l) p(i-k, j-l) \qquad \vec{E}(i,j) = \begin{pmatrix} E_c(i,j) \\ E_r(i,j) \end{pmatrix} = \begin{pmatrix} m_c(i,j) \\ m_r(i,j) \end{pmatrix}$$

(here we introduce the subscript as a derivative notation).

The contrast is the edge energy: $E(i,j) = \|\vec{E}(i,j)\| = \sqrt{E_r(i,j)^2 + E_c(i,j)^2}$

The direction of maximum contrast is the phase angle $\varphi(i,j) = Tan^{-1}\left(\frac{E_c(i,j)}{E_r(i,j)}\right)$

Sobel's edge filters can be seen as a composition of an image derivative and a smoothing filter.

$$m_c(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} = b_c(i,j) * d_r(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} b_c(k,l) d_r(i-k, j-l)$$

$$m_r(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} = b_r(i,j) * d_c(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} b_r(k,l) d_c(i-k, j-l)$$

The filter $d_c(i,j) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$ is a form of image derivative in the column direction

The filter $b_c(i,j) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ is a binomial smoothing filter in the column direction

1.2 Difference Operators: Derivatives for Sampled Signals

For the function, s(x) the derivative can be defined as a one sided dervative or a two sided derivative:

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

A two sided (symmetric) derivative is $\frac{\partial s(x)}{\partial x} = \lim_{\Delta x \to 0} \left\{ \frac{s(x+\Delta x) - s(x-\Delta x)}{\Delta x} \right\}$

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

For the two sided derivative: $\frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n-1)}{1} = s(n) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$

This is the operator used by Sobel.

Δn = 1 : $\frac{\Delta s(n)}{\Delta n} = \frac{s(n+1) - s(n-1)}{1} = s(n) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$

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

1.3 Fourier Analysis of Derivative Operators

Note that a derivative is equivalent to a 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} {–jω} 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].

The Fourier Transform of a discrete signal is periodic in frequency with period π.

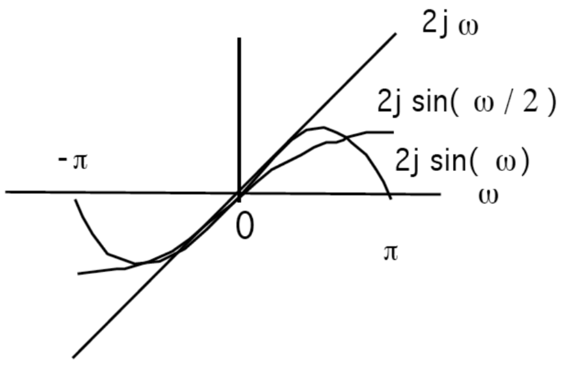
$$F\{s(n)\} = S(\omega) = \sum_{n=-\infty}^{\infty} s(n)e^{-j\omega n}$$

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.

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



The derivative used by Sobel attenuates aliasing noise in high frequencies.

1.4 Smoothing: The Binomial Low pass filter.

Sobel uses a filter b(m) = [1, 2, 1] to smooth.
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.

Les coefficients du suite binomial sont générés par le triangle de Pascal :

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.

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

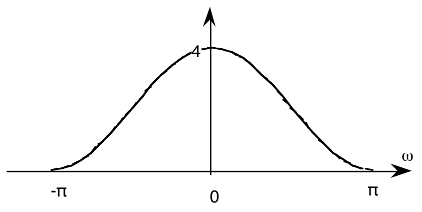
Gain : $$s_n = \sum_{m=1}^{n} b_n(m) = 2^n$$

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

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

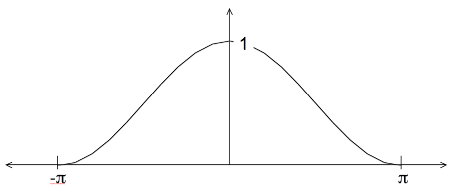
$$B_2(\omega) = \sum_{m=-1}^{1} b_2(m) e^{-j\omega m}$$
$$B_2(\omega) = 1e^{-j\omega(-1)} + 2e^{-j\omega 0} + 1e^{-j\omega(1)}$$
$$B_2(\omega) = 2 + e^{j\omega} + e^{-j\omega}$$
$$B_2(\omega) = 2 + 2\cos(\omega)$$



If we normalize the gain: $b_2(m) = (1/4)[1, 2, 1]$

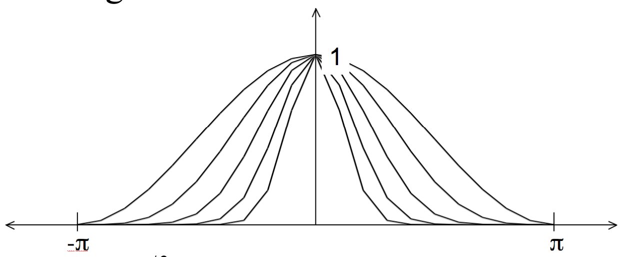
$$B_2(\omega) = \frac{1}{2} + \frac{1}{2}\cos(\omega)$$

Which is a cosine on a platform



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

Repeated convolution makes generates



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

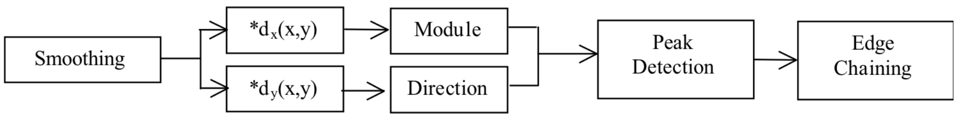
The binomial coefficients provide a series of low pass filters with no ripples.
In 2D, the filters provide separable filters that are nearly circularly symmetric

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

2 Describing Contrast with Edges

Contrast is "change in image intensity".
Classically, contrast was described by detecting "edges".

2.1 Edge Detection using integer coefficient filters



Gradient of the image is a vector: $\vec{\nabla} P(i,j) = \begin{pmatrix} p_i(i,j) \\ p_j(i,j) \end{pmatrix}$

where $p_i(i,j) = \dfrac{\Delta p(i,j)}{\Delta i} = p(i,j) * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ $\quad p_j(i,j) = \dfrac{\Delta p(i,j)}{\Delta j} = p(i,j) * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

Gradient Magnitude indicates edge strength $E(i,j) = \|\vec{\nabla} P(i,j)\|$

Gradient direction is the direction of maximum contrast

As an angle: $\vartheta(i,j) = Tan^{-1}\left(\dfrac{p_j(i,j)}{p_i(i,j)}\right)$

As a vector: $\vec{d}(i,j) = \begin{pmatrix} Sin(\vartheta(i,j)) \\ Cos(\vartheta(i,j)) \end{pmatrix} = \dfrac{1}{\|\nabla P(i,j)\|}\begin{pmatrix} p_i(i,j) \\ p_j(i,j) \end{pmatrix} = \begin{pmatrix} \dfrac{p_i(i,j)}{\|\nabla P(i,j)\|} \\ \dfrac{p_j(i,j)}{\|\nabla P(i,j)\|} \end{pmatrix}$

We can get the direction of direction of maximum gradient, $\vartheta(i,j)$, by normalizing the derivatives.

2.2 Non-maximum suppression.

Contrast points C(i, j) are local maxima in $\vec{\nabla} P(i,j)$ in the direction of maximum gradient. $\vartheta(i,j)$

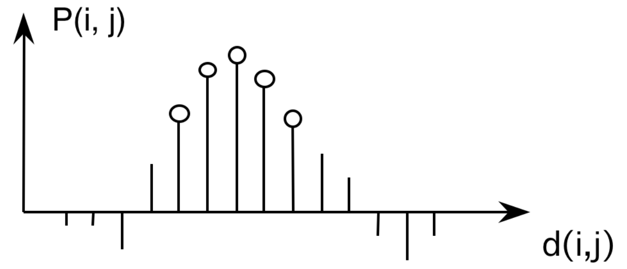
$$\vec{\nabla} P(i,j) = \begin{pmatrix} p_i(i,j) \\ p_j(i,j) \end{pmatrix} \qquad \vartheta(i,j) = Tan^{-1}\left(\frac{\Delta_j p(i,j)}{\Delta_i p(i,j)}\right) = Tan^{-1}\left(\frac{E_j(i,j)}{E_i(i,j)}\right)$$

For each edge pixel :

1) Determine the direction of maximum gradient:

$$\Delta i = \cos(\vartheta(i,j)) = \frac{P_i(i,j)}{\left\|\vec{\nabla} P(i,j)\right\|} \qquad \Delta j = \sin(\vartheta(i,j)) = \frac{P_j(i,j)}{\left\|\nabla P(i,j)\right\|}$$

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



$$c(i,j) = \begin{cases} 1 & \text{IF } E(i-\Delta i, j-\Delta j) \leq E(i,j) \geq E(i+\Delta i, j+\Delta j) \\ & \text{and IF } E(i-2\Delta i, j-2\Delta j) < E(i,j) > E(i+2\Delta i, j+2\Delta j) \\ 0 & \text{Otherwise} \end{cases}$$

Alternatively, we can consider $E(i,j) = \left\|\vec{\nabla} P(i,j)\right\|$ as "evidence" for contrast at all points

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)

3.1 The Algorithm

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 : $h(c, \theta)$.

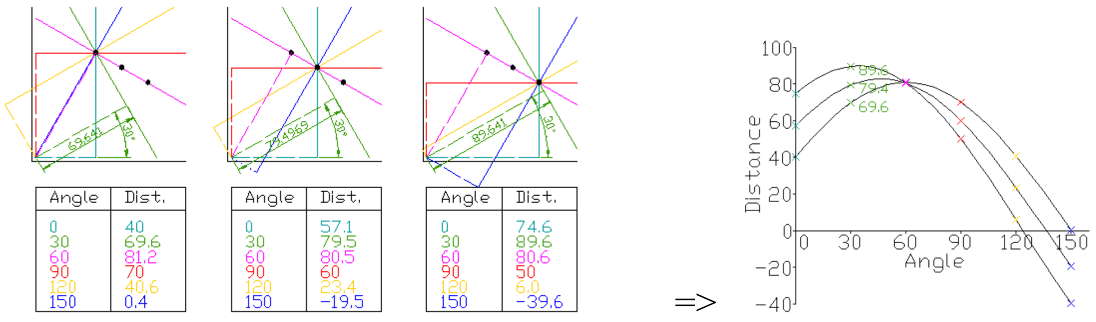
Let the c be an integer $c \in [0, D]$ where D is the "diagonal distance of the image.
Let θ be an integer $\theta \in [0, 179]$

Algorithm:

```
allocate a table $h(c, \theta)$ initially set to 0.
For each x, y of the image
    for $\theta$ from 0 to 179
        $c = -x \cos(\theta) - y\sin(\theta)$
        $h(c, \theta) = h(c, \theta) + E(x, y)$
    End
End
```

The resulting table accumulates contrast.

Peaks in h(c, θ) correspond to line segments in the image.



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6

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

3.2 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 $a_{max}$
            $b = -y - \mathrm{sqrt}(r^2 - (x - a)^2)$
            $h(a,b,r) = h(a,b,r) + E(x,y)$.
        End
    End
End
```

4 Beyond Edges - Describing Local Appearance

Appearance is what you see. Consider an image p(i, j).

We seek a set of K local basis functions, $f_k(x,y)$ to describe "appearance" around a point in an image. These are referred to as "local image description functions" or simply "Local features". (x and y are integers).

Each function, $f_k(x,y)$ gives an image "feature", a_k, describing appearance in the neightborhood of the image position p(i, j).

$$a_k(i,j) = \sum_{x=-R}^{R} \sum_{y=-R}^{R} p(i-x, j-y) f_k(x,y)$$

Projection of the image neighborhood *p(i,j)* onto this set of functions gives a

"feature" vector for appearance, $\vec{A}(i,j) = \begin{pmatrix} a_1 \\ a_2 \\ \ldots \\ a_K \end{pmatrix}$ around position *p(i,j).*

Ideally the set of local image description functions should be orthogonal

$$\sum_{x=-R}^{R} \sum_{y=-R}^{R} p(i-x, j-y) f_k(x,y) = 0 \quad \text{if } n \neq m \quad \text{for all } (i, j)$$

to minimize the number of features. However, it can be shown that a function cannot be both orthogonal and shift invariant. Because shift invariance is important for robust view invariant recognition, we compromise on orthoganality.

For a variety of reasons, derivatives of the Gaussian function have been found to be very useful as local image features. Chief among these are invariance to affine transformations.

5 Image Description Using Gaussian Derivatives

5.1 The Gaussian Function

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

The Gaussian function is invariant to affine transformations.

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

For example, a change in scale is an affine transform:

$$T_s\{G(x, y, \sigma)\} = G(T_s\{x\}, T_s\{y\}, T_s\{\sigma\}) = G(sx, sy, s\sigma)$$

This is just one of the many interesting properties of the Gaussian function when used as the basis for an image descriptor.

5.2 Gaussian Derivatives Operators

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

Fourier Transform: $F\{e^{-\frac{x^2}{2\sigma^2}}\} = G(\omega,\sigma) = \sqrt{2\pi}\ \sigma e^{-\frac{1}{2}\sigma^2\omega^2}$

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

Derivatives:

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

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

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

5.3 2D Gaussian functions

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

Fourier Transform: $F\{e^{-\frac{x^2+y^2}{2\sigma^2}}\} = \frac{\pi}{2\sigma^2} e^{-\frac{1}{2}\sigma^2(u^2+v^2)}$

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

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

Derivatives:

$$\frac{\partial G(x,y,\sigma)}{\partial x} = -\frac{x}{\sigma^2} G(x,y,\sigma) = G_x(x,y,\sigma)$$
$$\frac{\partial G(x,y,\sigma)}{\partial y} = -\frac{y}{\sigma^2} G(x,y,\sigma) = G_y(x,y,\sigma)$$
$$\frac{\partial^2 G(x,y,\sigma)}{\partial x^2} = \frac{x-\sigma^2}{\sigma^4} G(x,y,\sigma) = G_{xx}(x,y,\sigma)$$
$$\frac{\partial^2 G(x,y,\sigma)}{\partial x \partial y} = \frac{xy}{\sigma^4} G(x,y,\sigma) = G_{xy}(x,y,\sigma)$$
$$\frac{\partial^3 G(x,y,\sigma)}{\partial x^3} = \frac{x^3 - x\sigma^2}{\sigma^6} G(x,y,\sigma) = G_{xxx}(x,y,\sigma)$$

The Laplacian of the Gaussian: $\nabla^2 G(x,y,\sigma) = G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma)$

Diffusion Equation: $\nabla^2 G(x,y,\sigma) = \frac{\partial^2 G(x,y,\sigma)}{\partial x^2} + \frac{\partial^2 G(x,y,\sigma)}{\partial y^2} = \frac{\partial G(x,y,\sigma)}{\partial \sigma}$

As a consequence: $\nabla^2 G(x,y,\sigma) \approx \left(G(x,y,\sigma_1) - G(x,y,\sigma_2)\right)$

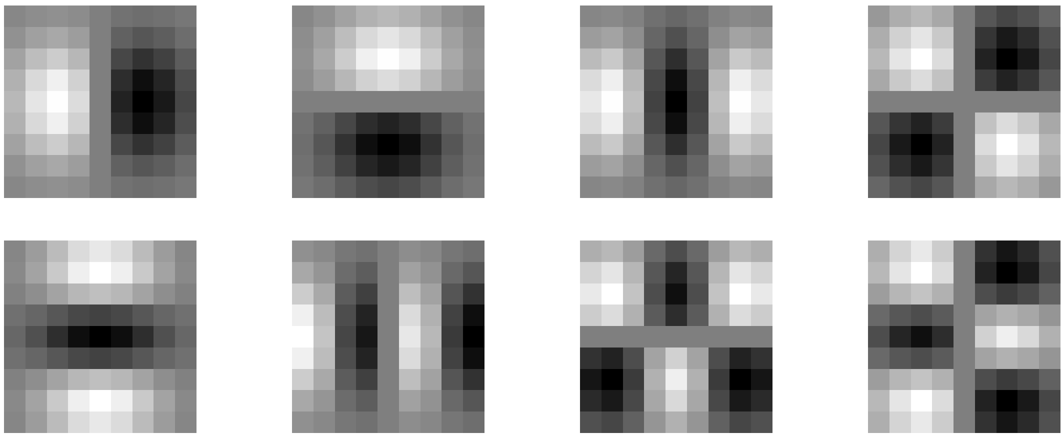
This is called a Difference of Gaussian (DoG) and typically requires $\sigma_1 \geq 1.4\, \sigma_2$
It is common to use: $\nabla^2 G(x,y,\sigma) \approx G(x,y,\sqrt{2}\sigma) - G(x,y,\sigma)$

But note that from the scale property : $G(x,y,\sqrt{2}\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma)$

so that $\nabla^2 G(x,y,\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma) - G(x,y,\sigma)$

We can use these functions to create a basis set of receptive fields for appearance

$G = (G_x, G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy})$



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

5.4 Using the Gaussian to compute image derivatives

For an image *p(i,j)*, the derivatives can be approximated by convolution with Derivatives of Gaussians.

$$\frac{\partial p(i,j)}{\partial x} * G(x,y) = \frac{\partial}{\partial x} * p(i,j) * G(x,y) = \frac{\partial}{\partial x} * G(x,y) * p(i,j) = \frac{\partial G(x,y)}{\partial x} * p(i,j)$$

Thus we can approximate an image derivative as $P_x(i,j) \approx G_x * P(i,j)$
However to compute G_x, it is NECESSARY to specify σ.
Small σ is not necessarily best.

$$p_x(i,j,\sigma) \approx G_x(x,y,\sigma) * p(i,j)$$

or more simply $p_x(i,j,\sigma) \approx G_x(\sigma) * p(i,j)$

Simarly:
$$p_y(i,j,\sigma) \approx G_y(\sigma) * p(i,j)$$
$$p_{xx}(i,j,\sigma) \approx G_{xx}(\sigma) * p(i,j)$$
$$p_{xy}(i,j,\sigma) \approx G_{xy}(\sigma) * p(i,j)$$
$$p_{yy}(i,j,\sigma) \approx G_{yy}(\sigma) * p(i,j)$$

The Gradient of the image $\vec{\nabla} p(i,j)$ is calculated by $\vec{\nabla} G(\sigma) * p(i,j)$

where $\vec{\nabla} G(\sigma) = \begin{pmatrix} G_x(\sigma) \\ G_y(\sigma) \end{pmatrix}$ This gives:

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

Laplacien:
$$\nabla^2 p(i,j,\sigma) = \nabla^2 G(\sigma) * p(i,j) = p_{xx}(i,j,\sigma) + p_{yy}(i,j,\sigma) \approx G_{xx}(\sigma) * p(i,j) + G_{yy}(\sigma) * p(i,j)$$

To use Gaussian functions to describe images we need to sample the Gaussian an limit its extent. That is, we must define Gaussian Filters.

6 Gaussian Funtion as a digital filters

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

To obtain a digital Gaussian filter we must perform two operations:
1) Sample the spatial axis x, y at a rate of Δx, and Δy
2) Limit the spatial extent with a window $W_N(x,y)$

6.1 Sampling

Let us consider the case of a 1-D Gaussian.

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

To sample we replace x with nΔx.

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

This is modeled as multiplication by an infinite pulse chaine.

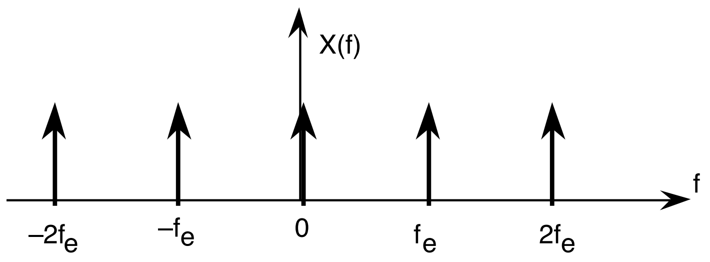
$$\begin{tikzcd}
G(x) \arrow[r] & \otimes \arrow[r] & G(n\Delta x) \\
& \delta_n(x) \arrow[u] &
\end{tikzcd}$$

We can set the sample size to $\Delta x = 1$. This gives a sampled function

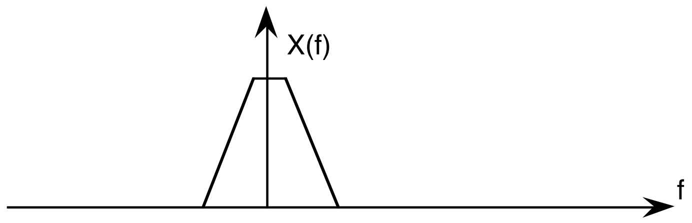
$$G(n,\sigma) = e^{-\frac{n^2}{2\sigma^2}}$$

$$x_{ei}(t) = x(t) \bullet T_e\, \delta_{Te}(t) = T_e \sum_{n=\infty}^{\infty} x(t)\, \delta(t - nT_e)$$

Sampling converts the Fourier Transform from an infinite function to a periodic function. The ideal sample function is a

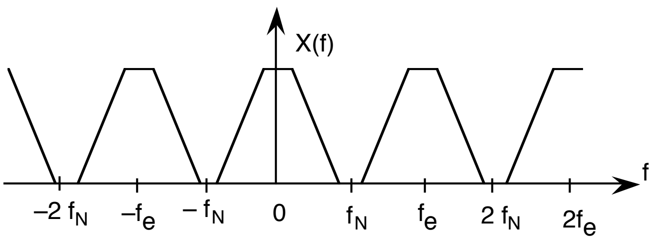


For a spectrum:



Sampling creates multiple copies intervals of f_e.

The Nyquist frequency is $f_n = \frac{f_e}{2} = \frac{1}{2\Delta x}$



The fourier Transform of the Gaussian is

$$F\{e^{-\frac{x^2}{2\sigma^2}}\} = G(\omega,\sigma) = \sqrt{2\pi}\ \sigma e^{-\frac{1}{2}\sigma^2(2\pi f)^2}$$

The tail of the Gaussian beyond $f_N = ½\Delta x$ will be converted to noise.
We need to insure that the integral from f_n to infinite is small.

Rule of thumb : assure that $\Delta x \leq \sigma$

6.2 Setting the Spatial Extent (Window Size).

To represent this in a computer we must also specify the spatial extent (number of samples), N of the filter. We set N = 2R + 1 where R is the "radius" of the function.

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 $W_N(n)$

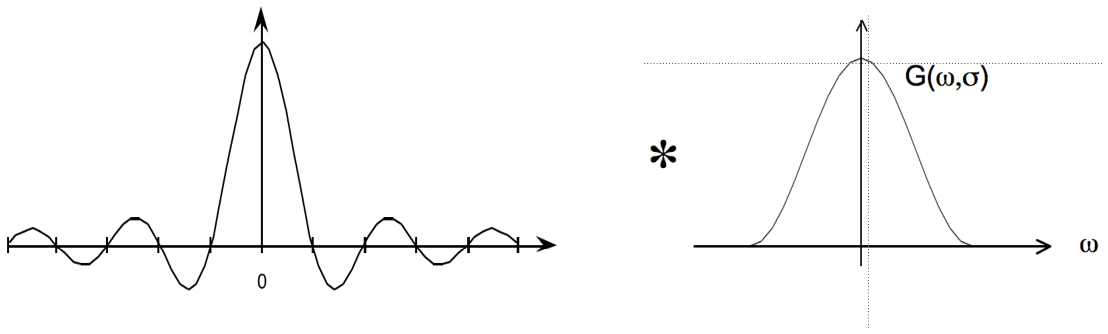
When we limit G(x,σ) to a finite support, we multiply by a window

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

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

$$F\{G(n,\sigma)\cdot w_N(n)\} = G(\omega,\sigma) * W_N(\omega)$$

where $W_N(\omega)=\frac{\sin(\omega N/2)}{\sin(\omega/2)}$ and $G(\omega,\sigma)=\sqrt{2\pi}\,\sigma\, e^{-\frac{1}{2}\sigma^2\omega^2}$



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 σ/Δx < 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.

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

The sampled Gaussian and its derivatives are:

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

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

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

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

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

Note the scale parameter σ determines the "resolution" of the derivatives.
You MUST specify σ. The smallest σ is not always the best.
Many computer vision algorithms give unpredictable results because the researchers forget to specify the scale σ at which the algorithm was validated.