Computer Vision

MR2 Informatics Option GVR

James L. Crowley

Fall Semester 10 November 2016

Lesson 4

Describing Local Appearance with Gaussian Derivatives

Lesson Outline:

1. Image Description Using Gaussian Derivatives

1.1. 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.

1.2. Gaussian Derivatives Operators

The Gaussian function is: $G(x,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$

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:

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

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

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

1.3. 2D Gaussian functions

2D Gaussian Kernel: $$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

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: $$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} * \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}$$

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

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

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

The 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: $$\nabla^2 G(x,y,\sigma) \approx G(x,y,\sqrt{2}\sigma) - G(x,y,\sigma)$$

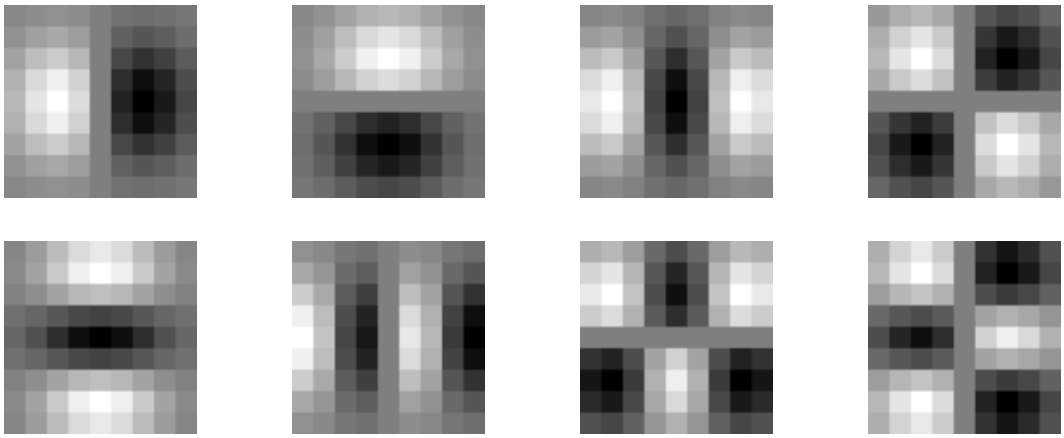
But note that from the scale property: $$G(x,y,\sqrt{2}\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma)$$

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

(note - this requires that *G(x,y,σ)* to be normalized to sum to 1.

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 $G_x, G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy}$.

1.4. A vector for describing local appearance.

Derivatives of the Gaussian function have been found to be very useful as local image features. Chief among these are invariance to affine transformations.

To describe the local appearance in an image p(i, j), we "project" a local neighborhood (window) onto a set of feature functions, $G_k(x,y,\sigma)$.

These are referred to as "local image description functions" or "Local features".

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

For example:

$$\vec{G}^{\sigma} = G_k^{\sigma} = (G_x, G_y, G_{xx}, G_{yy}, G_{xy})$$

Note that you must specify σ. σ=1 is not necessarily best.

Each function, $G_k(x,y,\sigma)$ gives an image "feature", a_k, describing appearance in the neighborhood of the image position p(i, j). (let x and y be integers).

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

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

"feature" vector for appearance, $\vec{A}(i,j) = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_K \end{pmatrix}$ at each pixel *p(i,j)*.

We can use this feature vector much as we used color features - to detect objects based on their appearance.

1.5. 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 and limit its extent. That is, we must define Gaussian Filters.

1.6. The Laplacian of the Gaussian and the DoG

The Laplacian of Gaussian is a scalar value:

$$\nabla^2 G(x,y,\sigma) = G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma) = \frac{\partial G(x,y,\sigma)}{\partial \sigma}$$

Because it is the derivative with respect to s, it can be approximated by a difference of Gaussians (DoG) :

$$\nabla^2 G(x,y,\sigma) \approx G(x,y,\sigma_1) - G(x,y,\sigma_2)$$

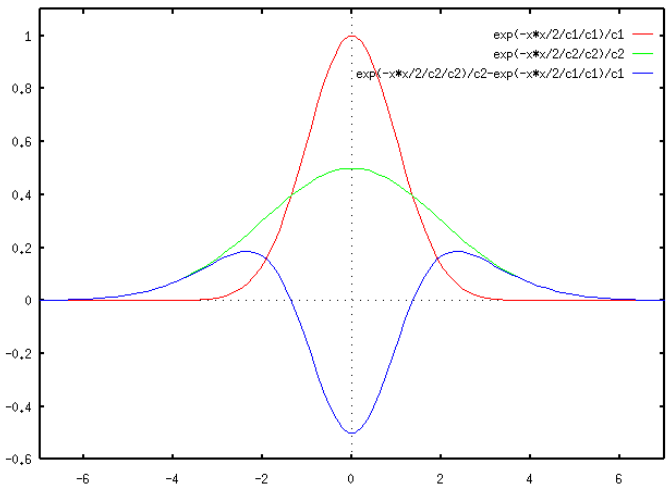
This is called a Difference of Gaussian and typically requires $\sigma_1 \geq 1.4\ \sigma_2$

It is common to use: $\nabla^2 G(x,y,\sigma) \approx G(x,y,\sqrt{2}\sigma) - G(x,y,\sigma)$
Because of the scale property: $G(x,y,\sqrt{2}\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma)$

We can easily compute a DoG as

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

In 1D:



2. Using the Gaussian function as a low-pass digital filter

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 $W_N(x,y)$

$$G(x,y;\sigma) \rightarrow G(i,j;\sigma) \equiv W_N(i,j)G(i\Delta x, j\Delta y;\sigma)$$

Thus there are 2 parameters to Control:
1) Sample Distance Δx
2) Window size, $N = 2R+1$

These are both determined by "scale" parameter of the Gaussian: σ

Sample Distance: Easy answer – Let $\Delta x = 1$ and control σ.
This is valid, provided that $\sigma \geq \Delta x$ or that $\sigma/\Delta x \leq 1$

Window Size: $R \geq 3\sigma$ Thus $N \geq 6\sigma+1$

Note that $R = 3\sigma$ is a lower limit that can leave some windowing noise in the function.

2.1. Sampling (Optional advanced subject)

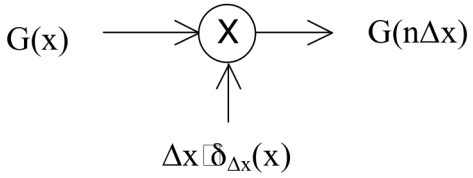
Let us consider the case of a 1-D Gaussian.

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

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 chain.



where:

$$\delta_{\Delta x}(x) = \sum_{n=-\infty}^{\infty} \delta_{\Delta x}(n\Delta x)$$

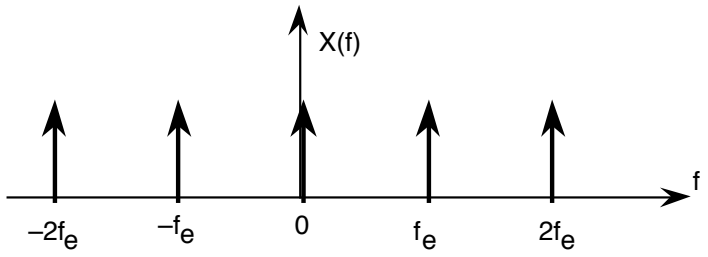
So that

$$G(n) = G(x)\cdot \Delta x \delta_{\Delta x}(x) = \sum_{n=-\infty}^{\infty} G(x)\cdot \delta_{\Delta x}(x - n\Delta x)$$

Multiplication in Space is a Convolution in Frequency. The Fourier transform of the sampling function is:

$$F(\delta_{\Delta x}(x)) = \Delta x \sum_{n=-\infty}^{\infty} \delta\ (nf_{\Delta x})$$

The ideal sample function is a



Where $f_{\Delta x}$ is the "Nyquist" frequency $f_{\Delta x} = \frac{1}{2\Delta x}$

In the Frequency domain, sampling converts the Fourier Transform of the Gaussian into an infinite sequence of Gaussians.

Transform of the Gaussian is

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

Sampling creates multiple copies intervals of $G(\omega)$ at intervals of $f_{\Delta x} = \frac{1}{2}\Delta x$

The tail of the Gaussian beyond $f_{\Delta x} = \frac{1}{2}\Delta x$ will be converted to noise.
We need to insure that the integral from f_n to infinite is small.

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

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

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

2.2. Setting the Window Size (Optional advanced subject)

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 $W_N(n)$

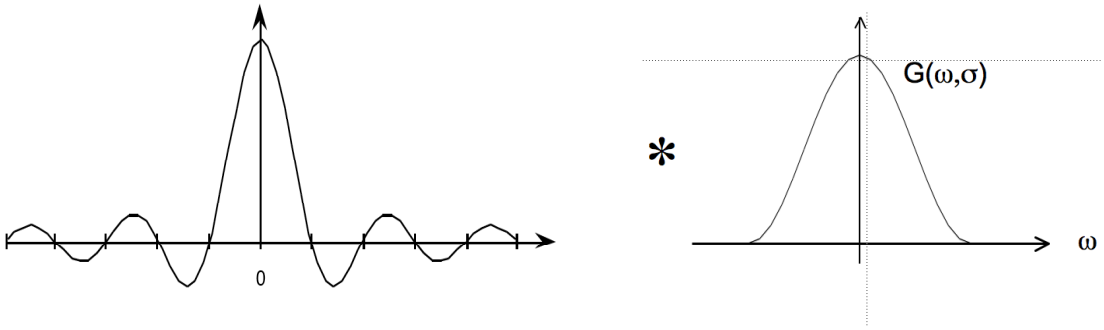
When we limit $G(x,\sigma)$ to a finite support, we multiply by a window

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

(note N = 2R+1). 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 $\sigma/\Delta x < 1$, 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 size 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.

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

2.3. 1-D Sampled Gaussian Derivative Filters

The sampled Gaussian and its derivatives are:

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

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

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

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

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.

2.4. The 2D Sampled Gaussian Function

The 2D Gaussian Receptive Field is : $G(i,j,\sigma) = \frac{1}{B}W_N(i,j)\cdot e^{-\frac{(i^2+j^2)}{2\sigma^2}}$

where

$$w_N(i,j) = \begin{cases} 1 & \text{for } -\mathrm{R} \le \mathrm{i} \le \mathrm{R} \text{ and } -\mathrm{R} \le \mathrm{j} \le \mathrm{R} \\ 0 & \text{otherwise} \end{cases}$$

Finite window, $w_N(i, j)$ has $N^2 = (2R+1)^2$ coefficients

Typically: for R should be ≥ 3σ . Recommend R=4σ

The normalization factor $B = \sum_{x=-R}^{R}\sum_{y=-R}^{R} e^{-\frac{(i^2+j^2)}{2\sigma^2}} \approx 2\pi\sigma$

3. Using the Gaussian to compute image derivatives

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

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

Where $G(\sigma) = G(i,j,\sigma)$.
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. Information exists at ALL values of σ.

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

Similarly:

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

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

Laplacien:

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

3.1. Steerability of Gaussian Derivatives.

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. The weights are given by sine and cosine functions.

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

Higher order derivatives can also be steered.

Thus:

1st order $p_x^{\theta}(i,j,\sigma) = Cos(\theta)p_x(i,j,\sigma) + Sin(\theta)p_y(i,j,\sigma)$

2nd order $p_{xx}^{\theta}(i,j,\sigma) = Cos(\theta)^2 p_{xx}(i,j,\sigma) + 2Cos(\theta)Sin(\theta)p_{xy}(i,j,\sigma) + Sin(\theta)^2 p_{yy}(i,j,\sigma)$

3rd order

$$p_{xxx}^{\theta}(i,j,\sigma) = Cos(\theta)^3 p_{xxx}(i,j,\sigma) + 3\cdot Cos(\theta)^2 Sin(\theta)p_{xxy}(i,j,\sigma) + 3\cdot Cos(\theta)Sin(\theta)^2 p_{xyy}(i,j,\sigma) + Sin(\theta)^3 p_{yyy}(i,j,\sigma$$

By steering the derivatives to the local orientation, we obtain an "invariant" measure of local contrast. We can also "steer" in scale to obtain invariance to size.

Note, we can NOT steer the mixed derivatives, i.e $p_{xy}(i, j, \sigma)$

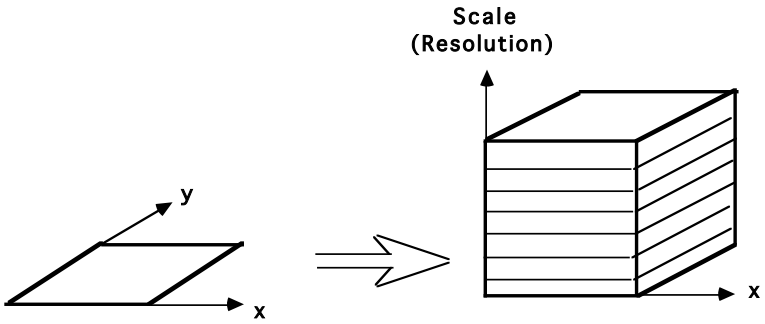
3.2. Intrinsic Orientation:

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

Note that local orientation depends on σ!

4. Image Scale Space



Scale space represents a 2D image in a 3D space, where the 3rd dimension is "resolution" or scale. This can be used to make image descriptions invariant to changes in size.

4.1. Continuous Scale Case.

Let p(x, y) be the image.
Let G(x, y, σ) by a Gaussian function of scale $\sigma=2^{s/2}$

The image Scale Space is a 3D continuous space *p(x, y, s)*

$p(x, y, s) = p(x,y)* G(x, y, 2^s)$

Note that the scale (s) axis is logarithmic. $s = Log_2(\sigma)$

4.2. Image Pyramids - Discrete Scale Space

Let *p(x, y)* is an image array of size WxH pixels, where (x, y) are integers,

We propose to sample scale with a step size of $\Delta\sigma = 2$ so that $\sigma_k=2^k$
For k=0, to K.

$\sigma_0=1$ is the smallest scale that we can represent.

At k=0 : $\sigma_0=2^0=1$.

let M = min(W, H)
K is the largest scale possible: $K=Log_2(M)$

at k=K, $\sigma_K=2^K=2^{\log(M)}=M=\min(W,H)$

For k > K the scale parameter σ is larger than the image.

4.3. Spatial Resampling and Image Pyramids

Because the Gaussian, $G(x, y, \sigma_k)$, is a low pass filter, as σ_k grows it becomes possible to resample the image with a larger step size without loss of information.

Such resampling has the benefit of assuring an invariance of the impulse response of each image. The sample size Δx_k, Δy_k can grow exactly as σ_k.

What sample size is possible? It is possible to show that the sample step must be smaller than σ. For example, let $\Delta x = \sigma/2$

Thus $\Delta x_k = \Delta y_k = 2^{k-1}$ with only minimal aliasing.

Resampling selects every Δx image sample:
for integer values of i, j:

$$p(i, j, k) = p(i\Delta x_k, j\Delta y_k, k) = p(x/\Delta x_k, y/\Delta x_k, k)$$

The position in the original image is $x = i\Delta x_k$ and $y = j\Delta y_k$

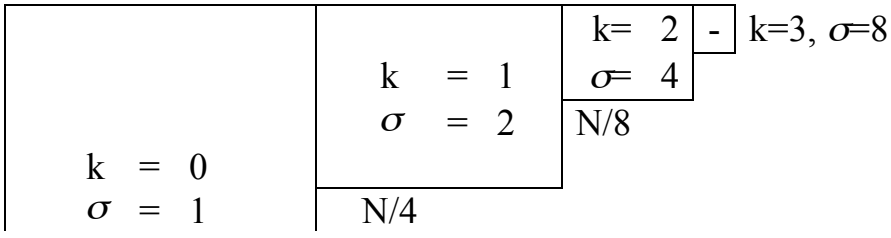
Resampling at $\Delta x_k = \sigma_k = 2^k$ results an identical impulse response at each level.
This property is called "scale invariance". (The impulse response is scale invariant).

A resampled scale space (a scale invariant pyramid), with a scale step of one "octave".

$$p(i, j, k) = p(x/\Delta x_k, y/\Delta x_k, k) \quad \text{such that } \Delta x_k = 2^{k+1} \text{ and } \sigma_k = 2^{k+1}$$

Note that it is possible to build a scale invariant pyramid with a step size of $\Delta\sigma = 2^{k/2}$ however this involves a very complicated resampling scheme that is beyond the scope of this class.

This can be drawn as a set of images:

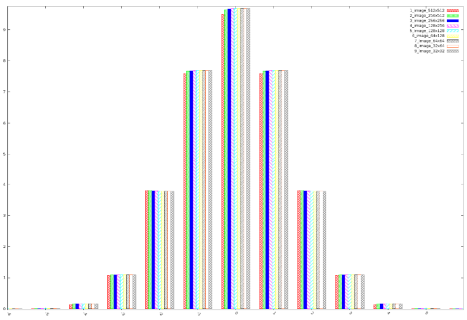


N

Let the image have M=NxN pixels. The total number of pixels is

$$P = M+M/4+M/8+M/16+M/64+...= 1.333... \text{ Pixels.}$$

When $\sigma/\Delta x$ is held constant, every "level" k of p(i, j, k) has the same impulse response.



In fact, a step size of $\sigma_k = 2^k$ is too large. We need at least $\Delta\sigma_k = 2^{1/2}$
A more reasonable pyramid is given by

$$\sigma_k = 2^{(k+1)/2} \text{ and } \Delta x = 2^{k/2}$$

Example:

This involves some tricky resampling, but gives a pyarmid of size

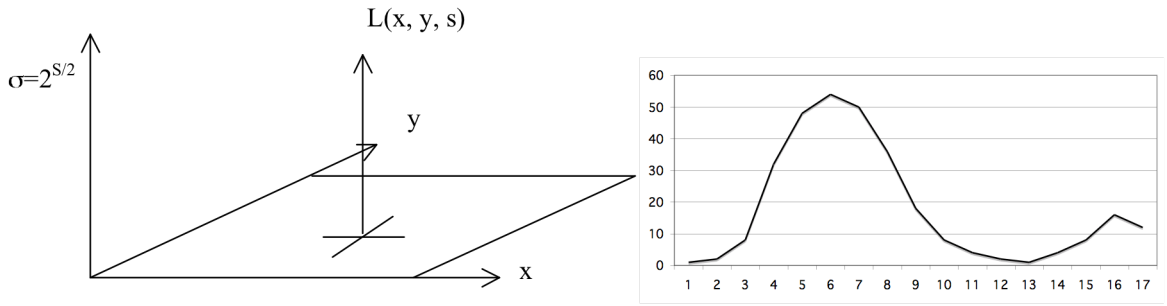
P = M+M/2+M/4+M/8+..... = 2M samples,

4.4. Laplacian Profile

At every image point, the Laplacian profile is the Laplacian of the image computed over a continuous (exponential) range of scales.

$$L(x, y, s) = P(x, y) * \nabla^2 G(x, y, 2^s)$$

The Laplacian profile is invariant to rotation and translation and equivariant to changes in scale. Since scale is proportional to distance, the profile is invariant to viewing distance.

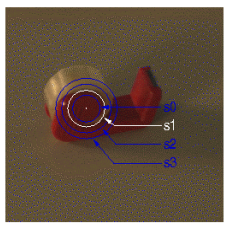


A change in viewing distance at x, y shifts the function L(x,y,s) in s.
The function remains the same. Thus the maximum is a local invariant.

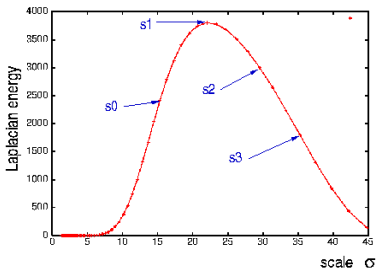
The "intrinsic" scale at a point (x, y) is $\sigma_i = 2^{s_i}$

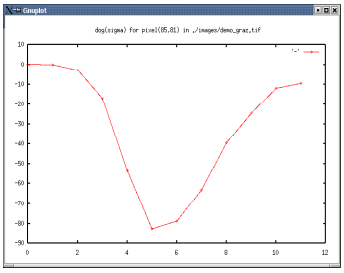
such that $s_i = \arg\max_s \{L(x,y,s)\}$

Examples:



zero crossing of Laplacian at si





The scale of the maximal Laplacian is an invariant at ALL image points.