Computer Vision

MSc Informatics option GVR
James L. Crowley

Fall Semester 8 November 2012

Lesson 5

View Invariant Image Description

Lesson Outline:

1. Beyond Edges - Describing Local Appearance

Appearance is what you see.

To describe the local appearance in an image p(i, j), we “project” a local neighborhood (window) onto a set of feature functions, $f_k(x,y)$. These are referred to as "local image description functions" or simply "Local features".

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

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

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

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

Ideally the set of local image description functions should be orthogonal

$$\sum_{x=-R}^{R} \sum_{y=-R}^{R} f_m(x,y) f_n(x,y) = 0 \quad \text{if } n \neq m$$

to minimize the number of features. However, it is more important that the features be discriminant and robust to changes in position and scale.

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.

2. Image Description Using Gaussian Derivatives

2.1. 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.

2.2. 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:

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

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

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

2.3. 2D Gaussian functions

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

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

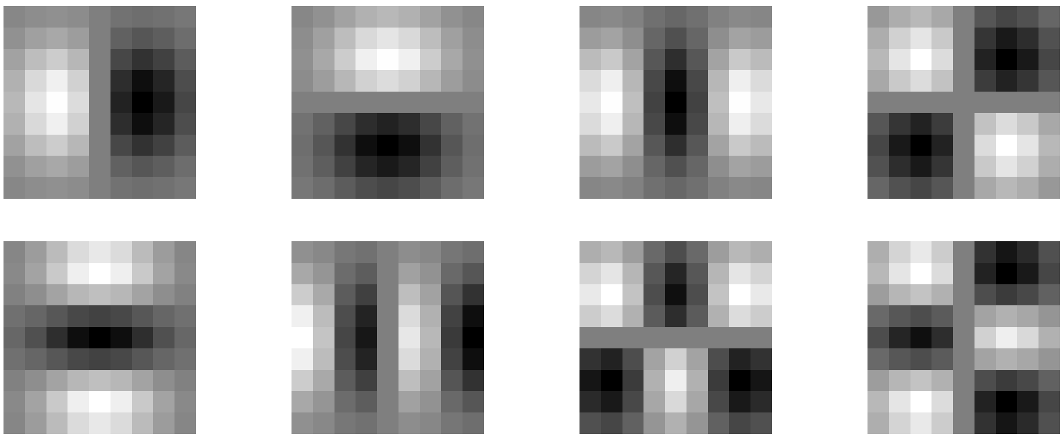
Separability: $$G(x,y,\sigma) = e^{-\frac{(x^2+y^2)}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} * e^{-\frac{y^2}{2\sigma^2}}$$

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

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

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

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



These are sometimes referred to as Gaussian receptive fields .

2.4. The Laplacian of the Gaussian

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

This is sometimes referred to as a "Mexican Hat" operator.

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

3. 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 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 $W_N(x\ ,y)$

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

Thus there are 3 parameters to Control:
1) Sample Distance Δx, Δy
2) Window size, $N = 2R+1$

These are based on the "scale" parameter of the Gaussian: σ

Sample Distance: Easy answer – Let $\Delta x = \Delta y = 1$ 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 $N \geq 6\sigma+1$

3.1. 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 $n\Delta x$.

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

This is modeled as multiplication by an infinite pulse chain.

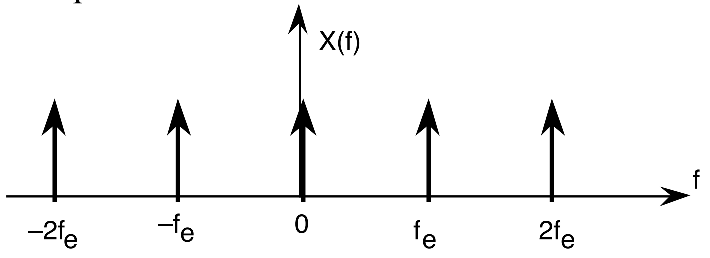
$$\begin{tikzcd}
G(x) \arrow[r] & \otimes \arrow[r] & G(n\Delta x) \\
& \delta_n(x) \arrow[u] &
\end{tikzcd}$$

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

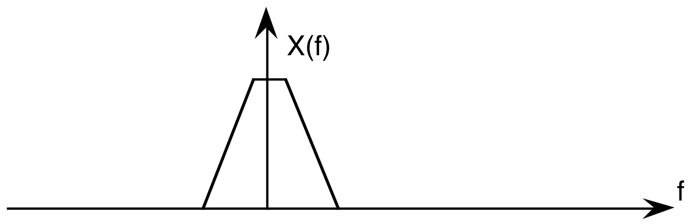
$$G(n,\sigma) = e^{-\frac{n^2}{2\sigma^2}}$$

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

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

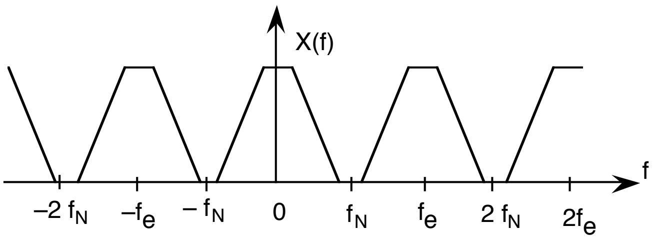


For a spectrum:



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 $\sigma \geq \Delta x$

3.2. 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 $W_N(n)$

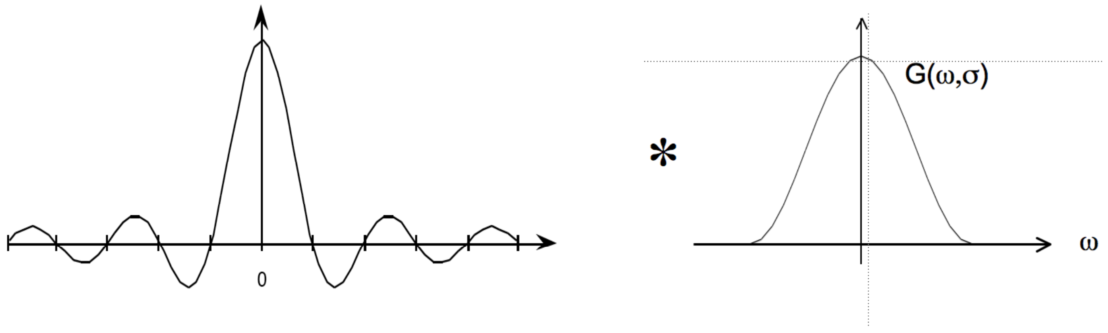
When we limit $G(x,\sigma)$ to a finite support, we multiply by a window

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

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

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

3.3. 1-D Sampled Gaussian Derivative Filters

The sampled Gaussian and its derivatives are:

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

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

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

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

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.

3.4. 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

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

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

Typically: for R should be $\geq 3\sigma$. Recommend $R=4\sigma$

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

3.5. 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.

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

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:

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

3.6. 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

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

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 $p_{xy}(i, j, \sigma)$

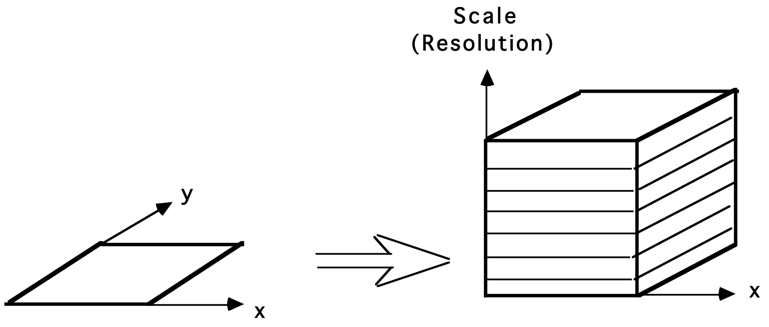
3.7. Intrinsic Orientation:

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



Our objective is to describe the appearance at each pixel of the image in a manner that does not change (or changes very little) with position, illumination color, distance, or viewing direction.
We will obtain invariance to position by using shift invariant filters
We will obtain invariance to illumination color by estimating the illumination color and correcting the image description in LC_1C_2.
We will obtain invariance to distance by working in a Scale Space

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)*

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

Note that the scale (s) axis is logarithmic. $s = \mathrm{Log}_2(\sigma)$

4.2. Discrete Scale Space

Suppose *p(x, y)* is an image array of size M x M pixels.

We will sample the scale axis with a step size of $\Delta\sigma = 2^{1/2}$: $\sigma_k \approx 2^{k/2}$

Because we use a low pass filter, $G(x, y, \sigma_k)$, 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 <u>invariance</u> of the impulse response of each image. The sample size Δx_k, Δy_k can grow exactly as σ_k.

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

What sample size is possible? It is possible to show that the sample step must be smaller than $\frac{1}{\sqrt{2}}\sigma$. $\Delta x \leq 2^{-1/2}\sigma$

If we choose k such that $\Delta x_k = 2^{k/2}$, then $\sigma_k = 2^{(k+1)/2}$

This defines a resampled scale space (a pyramid)

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

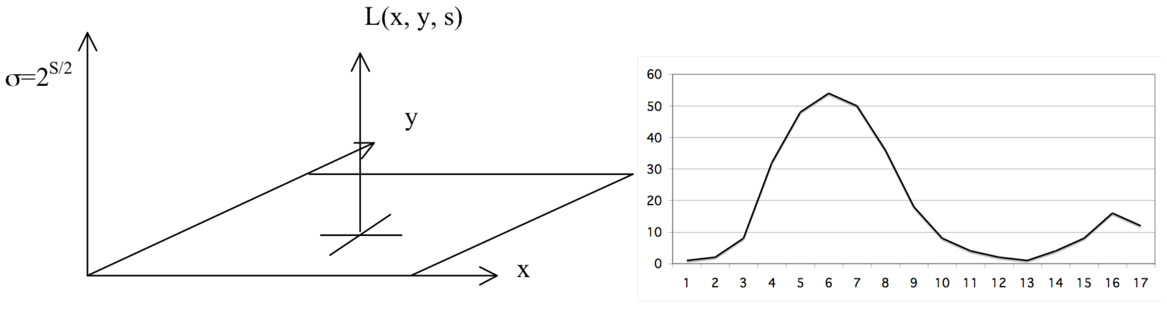
When $\sigma/\Delta x$ is held constant, every "level" k of p(i, j, k) has the same impulse response.

4.3. Laplacian Profile

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

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

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.



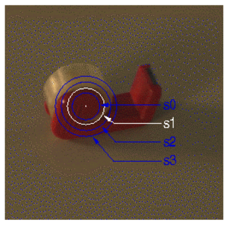
A change in viewing distance at x, y shifts the function L(x,y,s) in s.
The function remains the same. Thus the maximum is a local invariant.

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

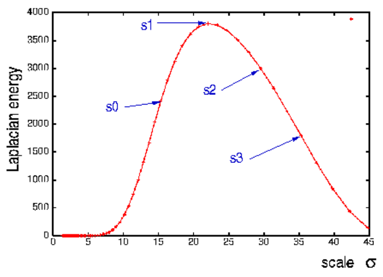
where:

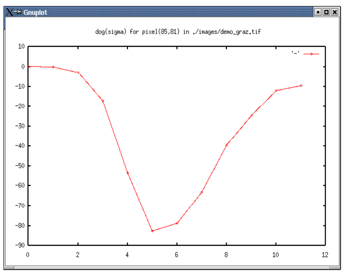
$$s_i = \arg\text{-}\max_s \{L(x,y,s)\}$$

Examples:



zero crossing of Laplacian at si





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

4.4. Scale Invariant Interest Points

Maximal points in the image derivatives provide keypoints.
In an image scale space, these points are scale invariant.

Example: maxima in the Lapacian as invariant "interest points"

Recall the Laplacian of the image :

$$\nabla^2 p(x,y,s) = \nabla^2 G(\sigma) * p(x,y) = G_{xx}(\sigma) * p(x,y) + G_{yy}(\sigma) * p(x,y)$$

from the diffusion equation, we know that for a Gaussian

$$\nabla^2 G(\sigma) = \frac{\partial^2 G(\sigma)}{\partial x^2} + \frac{\partial^2 G(\sigma)}{\partial y^2} = \frac{\partial G(\sigma)}{\partial \sigma} \approx \left(G(\sigma_1) - G(\sigma_2)\right)$$

This is referred to as a "Difference of Gaussian" or DoG.

The approximation works best at $\sigma_1 = \sqrt{2}\sigma_2$

Thus we can compute the Laplacian as a difference of Gaussians:

$$\nabla^2 G(\sigma) * p(x,y) \approx \left(G(\sigma_1) - G(\sigma_2)\right) * p(x,y) = G(\sigma_1) * p(x,y) - G(\sigma_2) * p(x,y)$$

and we can compute the Laplacian of the image as a difference of Gaussian smoothed images.

$$\nabla^2 p(i,j,\sigma) = \nabla^2 G(\sigma) * p(i,j) \approx G(\sqrt{2}\sigma) * p(i,j) + G(\sigma) * p(i,j)$$

Using an image Pyramid, the Laplacian is simply the difference at adjactent levels.

DoG: $L(i, j, k) = \nabla^2 p(i, j, k) = p(i, j, k) - p(i, j, k-1)$

We can detect scale invariant interest points as

$$X(i,j,k) = \underset{i,j,k}{local - \max}\{L(i,j,k)\}$$

4.5. Other popular interest point detectors.

Other popular detectors for scale invariant interest points include:

Gradient Magnitude: $A(i,j,k) = \underset{i,j,k}{Local - \max}\{\| P_x(i,j,k), P_y(i,j,k) \|\}$

and Determinant of the Hessian: $A(i,j,k) = \underset{i,j,k}{Local - \max}\left\{\det\begin{pmatrix} P_{xx}(i,j,k) & P_{xy}(i,j,k) \\ P_{xy}(i,j,k) & P_{yy}(i,j,k) \end{pmatrix}\right\}$

$$A(i,j,k) = \underset{i,j,k}{Local - \max}\left\{P_{xx}(i,j,k)P_{yy}(i,j,k) - P_{xy}(i,j,k)^2\right\}$$

and the Harris-Laplace.

let $b_2(i,j) = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

$$H_x^2 = b_2 * P_x^2$$
$$H_{xy} = b_2 * P_{xy}$$
$$H_y^2 = b_2 * P_y^2$$

$$H = \begin{pmatrix} H_x{}^2 & H_{xy} \\ H_{xy} & H_y{}^2 \end{pmatrix}$$

Harris interest points h(i,j,k) = arg-max{det(H)-Trace(H)}

5. HOG: Histogram of Oriented Gradients

A local histogram of gradient orientation provides a vector of features image appearance that is relatively robust to changes in orientation and illumination.

HOG gained popularity because of its use in the SIFT feature point detector (described next). It was subsequently explored and made popular by Navneet Dalal (M2R GVR 2003) and Bill Triggs.

Recall: The orientation of a gradient at pyramid sample (i,j,k) is:

$$\theta(i,j,k) = Tan^{-1}\left\{\frac{p_y(i,j,k)}{p_x(i,j,k)}\right\}$$

This is a number between 0 and π. We can quantize it to a value between 1 and N value by

$$a(i,j,k) = N \cdot Trunc\left\{\frac{\theta(i,j,k)}{\pi}\right\}+1$$

We can then build a local histogram for a window of size WxH, with upper left corner at i_o, j_o, k. We allocate a table of N cells: h(a). Then for each pixel i,j in our window:

$$\overset{W}{\underset{i=1}{\forall}}\overset{H}{\underset{j=1}{\forall}} h(a(i+i_o, j+j_o, k)) = h(a(i+i_o, j+j_o, k))+1$$

The result is a local feature composed of N values.
Recall that with histograms, we need around 8 samples per bin to have a low RMS error. Thus a good practice is to have N=W=H. For example N=4, W=4 and H=4. Many authors ignore this and use values such as N=8, W=4, H=4, resulting in a sparse histogram.

Remark: A fast version when N=4 replaces the inverse tangent by computing the diagonal derivatives with differences:

$$P_{\frac{\pi}{4}}(i,j,k) = P(i+1, j+1, k) - P(i-1, j-1, k)$$
$$P_{\frac{\pi}{2}}(i,j,k) = P(i, j+1, k) - P(i, j-1, k)$$
$$P_{\frac{3\pi}{4}}(i,j,k) = P(i+1, j-1, k) - P(i-1, j+1, k)$$
$$P_{\pi}(i,j,k) = P(i+1, j, k) - P(i-1, j, k)$$

To determine a(i,j,k) simply choose the maximum.

6. Scale Invariant Feature Transform (SIFT)

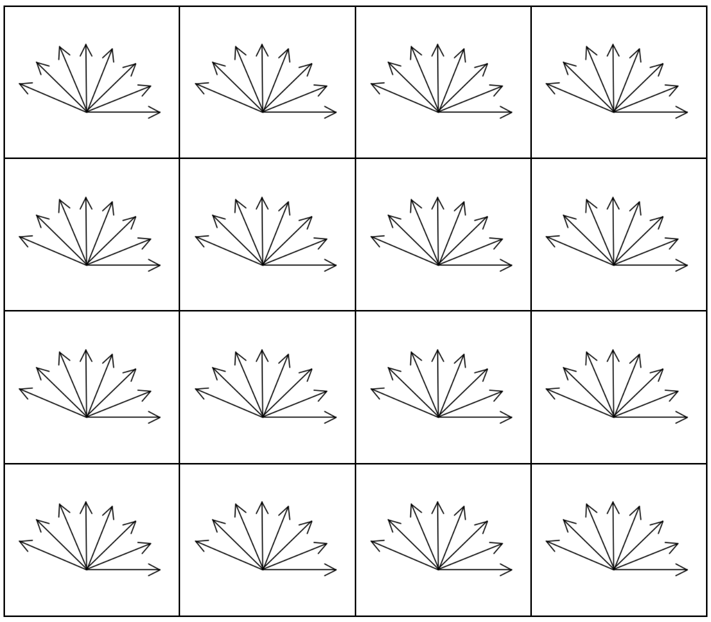
SIFT uses a scale invariant pyramid to compute a scale invariant DoG interest point detector to detect local scale-invariant interest points.

$$X(i,j,k) = \underset{i,j,k,R=2}{Local-\max}\{P(i,j,k) - P(i,j,k-1)\}$$

It then computes a U x V grid of HOG detectors with N=8, W=4, H=4 at the level k Typically U=V=4.

At level k, $\Delta i = \Delta j = 2^{k/2}$

This gives 16 x 16 = 128 features at each interest point.
This feature vector is invariant to changes in position and scale and very robust with changes in image plane rotation and illumination intensity.



Various authors experiment with other grid sizes.

For example, let the grid size be G.

G=4, W=4, H=4, N=4

Gives 64 features.