Computer Vision

MSc Informatics option GVR

James L. Crowley

Fall Semester 4 november 2008

Lesson 4
Gaussian Derivatives and Scale Space

Lesson Outline:

1 Gaussian Derivative Operators

1.1 The Sampled Gaussian Functions

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

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

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

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

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

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

The normalization factor $A = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} e^{-\frac{(i^2+j^2)}{2\sigma^2}} \approx 2\pi\sigma$

This factor will be slightly smaller if the Gaussian is evaluated over a finite window $W_N(i,j)$ of size NxN pixels.
This factor can be determined from the filter, and can be ignored in discussing the properties of the Gaussian.

Windowed Sampled 2D Gaussian (un-normalized) $G(i,j,\sigma) = W_N(i,j) \cdot e^{-\frac{i^2+j^2}{2\sigma^2}}$

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

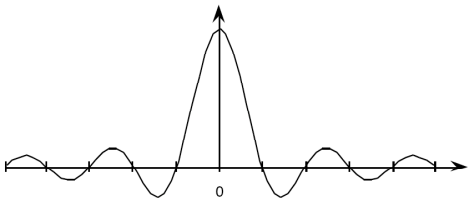
N = 2R+1

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

where : $W_N(u,v) = \frac{\sin(uN/2)\sin(vN/2)}{\sin(u/2)\sin(v/2)}$

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

$$W_N(n) = \begin{cases} 1 & \text{for } -R \le n \le R \\ 0 & \text{otherwise} \end{cases}$$



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

Typically: for R should be $\geq 3\sigma$ and Thus $N \leq 6R+1$

1.2 Properties of Gaussian Receptive Fields (or Wavelets)

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

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

Derivative Filters:

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

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

$$G_{xy}(i,j,\sigma) = \frac{ij}{\sigma^4} G(i,j,\sigma)$$

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

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

Diffusion Equation: $\nabla^2 G_x(i,j,\sigma) = G_{xx}(i,j,\sigma) + G_{yy}(i,j,\sigma) = \dfrac{\partial G(i,j,\sigma)}{\partial \sigma}$

As a consequence: $\nabla^2 G(i,j,\sigma) \approx G(i,j,\sigma_1) - G(i,j,\sigma_2)$

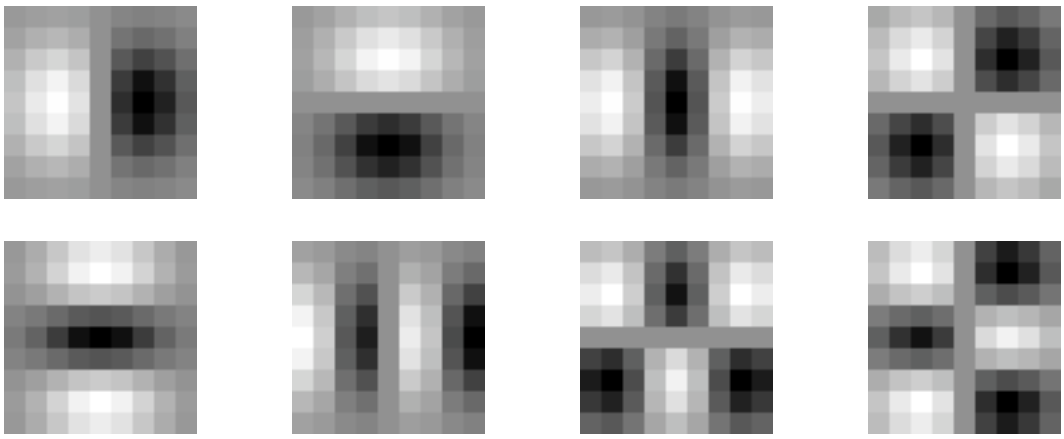
This typically requires $\sigma_1 \geq \sqrt{2}\ \sigma_2$

Thus it is common to use:

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

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

$\vec{G}(i,j,\sigma) = (G_x, G_y, G_{xx}, G_{xy}, G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy})$



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

These can be used to compute a local description known as the "local jet".

$$\vec{L}(i,j,\sigma) = P * \vec{G}(i,j,\sigma)$$

The Local jet can be steered to a normalization angle θ.

Steerability:

$$G_\theta^1(x,y,\sigma) = \cos(\theta)\cdot G_x(x,y,\sigma) + \sin(\theta)\cdot G_y(x,y,\sigma)$$

$$G_\theta^2(x,y,\sigma) = Cos(\theta)^2 G_{xx}(x,y,\sigma) + Cos(\theta)Sin(\theta)G_{xy}(x,y,\sigma) + Sin(\theta)^2 G_{yy}(x,y,\sigma)$$

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

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.

1.3 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: $\nabla^2 P(i,j) = P * \nabla^2 G(i,j,\sigma) = \begin{pmatrix} P * \nabla^2 G(i,j,\sigma) \\ P * \nabla^2 G(i,j,\sigma) \end{pmatrix} \approx P * \nabla^2 G(i,j,\sigma_1) - P * \nabla^2 G(i,j,\sigma_2)$

where typically $\frac{\sigma_1}{\sigma_2} \geq \sqrt{2}$

Because:

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

Thus:

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

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

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

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

2 Gaussian Scale Space

$$p(t,s) = x(t) * k(\frac{t}{s})$$

x(t) A signal (t is time or space)
k(t) A kernel function
p(t,s) A multi-scale representation of x(t) (s is scale)

Scaling x(t) translates p(t,s) in scale

However, to obtain "equvariant" scaling, we need to use Log(s).

$$p(t,s) = x(t) * k(\frac{t}{2^s})$$

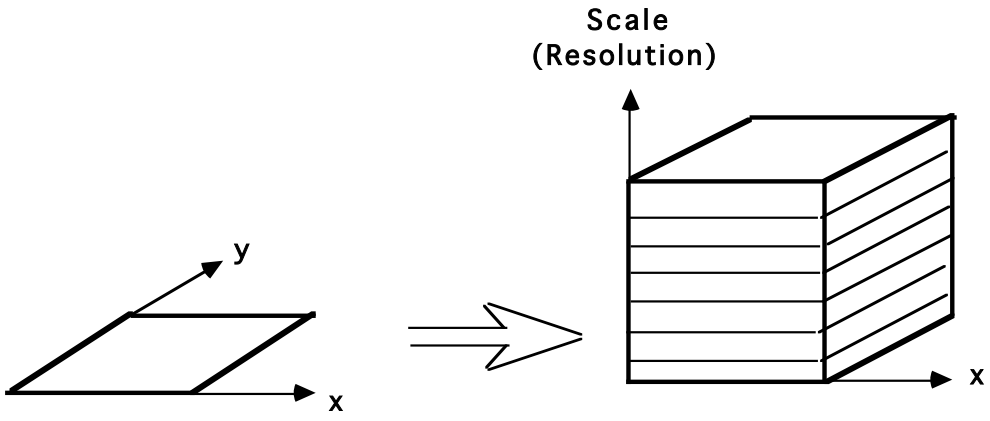
In this way, doubling the size of x(t) translates the signal by s+1.

2.1 Image Scale Space:

Continuous Case.

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

Continuous x, y, s: P(x, y, s) = P(x,y)* G(x, y, $2^{s/2}$)



The appearance of a pattern in the image results in a unique struction in P(x, y, s).
This structure is "equvariant" in position, scale and rotation.
Translate the pattern by Δx, Δy and the structure translates by Δx, Δy in P(x, y, s).
Rotate by θ in x,y and the structure rotates by θ in P(x, y, s).
Scale by a factor of 2^s, and the structure translates by s in P(x, y, s).

Scale space :

Separates global structure from fine detail.
Provides context for recognition.
Provides a description that is invariant to position, orientation and scale.

2.2 Discrete Scale Space

Let P(i,j) be an image of size MxM pixels

Let $G(i,j, 2^{k/2})$ be a kernel filter for $\sigma=2^{k/2}$ of size NxN were $N_k=2^{k/2+3}+1$

$R_k = 4\sigma_k \qquad N_k=2R_k+1 = 8\sigma_k +1 = 8\ (2^k)+1 = 2^{k+3}+1$

Then

$P(i,j,k) = P(i,j) * G(i,j, 2^{k/2})$

for $0 \leq k \leq M-4$

Cost of computing p(i,j,k) is

$C= O(M^2((N_0+1)^2+(N_1+1)^2+(N_2+1)^2+\ldots+(N_{M-4}+1)^2))$

if we use "seperable" convolution:

$P(i,j) * G(i,j, 2^{k/2}) = P(i,j) * G(i, 2^{k/2}) * G(j, 2^{k/2})$

then

$C= O(M^2 \cdot 2(N_0+N_1+N_2+N_3+\ldots+N_{M-4}+M-4+1)$

$C= O(M^2 \cdot 2(8+16+32+64+\ldots+ N_{M-4})+6).$

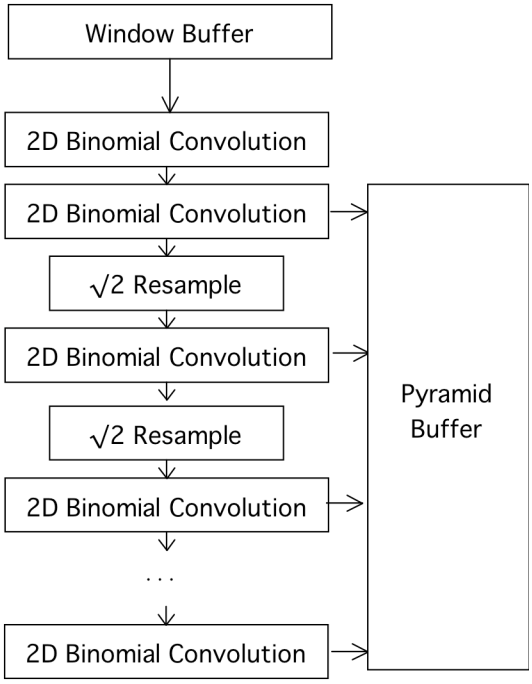
Practically, the computational cost is exorbitant.
We can use Pyramid Methods to reduce cost

2.3 Scale Invariant Pyramids

Some Bibliography:

1. M. D. Kelly. Edge detection by computer in pictures using planning. Machine Intelligence, 6:379-409, 1971.
2. S. L. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. Computer Graphics and Image Processing, 4:104-119, 1975.
3. H. P. Moravec. Towards automatic visual obstacle avoidance. In Proceedings of the 5th International Joint Conference on Artificial Intelligence, 1977.
4. J. L. Crowley, Analysis Synthesis and Evaluation of Linear Operators for Textures", IEEE Conference on P.R.I.P., Chicago, June, 1978.
5. J. L. Crowley, A Representation for Visual Information, Doctoral Dissertation, Carnegie Mellon University, CMU RI TR 82-7, Nov. 1981
6. P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code", IEEE Transactions on Communications, Vol 31, No. 4, 1983.
7. J. P. Crowley and R. M. Stern, "Fast Computation of the Difference of Low-Pass Transform", IEEE Transactions on PAMI, PAMI 6(2), March 1984.
8. J. J. Koenderink, A. J. van Doorn, "Representation of Local Geometry in the Visual System", Biological Cybernetics, pages 367-375, 1987
9. David G. Lowe, "Object Recognition from Local Scale-Invariant Features," iccv,pp.1150, Seventh International Conference on Computer Vision (ICCV'99) - Volume 2, 1999

Pyramid Algorithm:



The half-octave Gaussian pyramid for an NxN image is composed of up to K = $2Log_2(N)$ images. Each image, k [1, K] of the pyramid has been convolved with a Gaussian filter, $G(x,y,2^{k/2})$, and can be resampled with a sample distance of $2^{k/2}$, resulting in a constant ratio of scale over sample distance. The resulting "pyramid" represents the original NxN image with a sequence of K=2Log(N) images at a geometric progressions of scales $s_k=2^{k/2}$ each with half the number of samples of the previous, resulting in a total of 2xNxN samples.

The Gaussian pyramid for an M=NxN pixel image can be computed in O(M) operations using cascade convolution with resampling [Crowley-Stern84]. This algorithm involves alternatively convolving with a Gaussian support, and resampling the resulting image with a sample distance of √2. The effect of cascade convolution is to sum the variances of the filters, so that the cumulative variance is $s_k^2=2^k$ and the resulting standard deviation is $s_k=2^{k/2}$. Interleaving resampling with convolutions decreases the number of image samples while expanding the distance between samples. This has the effect of dilating the Gaussian support without increasing the number of samples used for the Gaussian, effectively increasing the scale. Aliassing is avoided (or minimized) by the fact that the image has been low-pass filtered by previous convolutions. The result is an algorithm with linear algorithmic complexity (i.s. O(M)) providing that gives a discrete representation of scale space with 2M total samples.

The image is initially convolved with a filter of $\sigma_o=1$ to produce an initial image P(x,y,0)

k=0: $P(x,y,0) = P(x,y) * G(x,y,1)$

where "$*$" is the convolution operator. The pyramid image (k=1) is produced by a convolution with the same low pass filter, resulting in a cumulative scale factor of $\sigma_1^2=4$ giving $\sigma_1=2$.

k=1: P(x,y,1) = P(x,y,0) * G(x,y,1)

Each successive image in the pyamid is computed by convolving an expanded Gaussian with a sampled image as described by the following recurrence equation:

$P(x,y,k) = S_{\sqrt{2}}\{P(x,y,k-1)\} * E_{\sqrt{2}}^{k}\{G(x,y,1)\}$

Where $S_{\sqrt{2}}\{\}$ is the "diagonal" resampling operator, shown in figure 1, and defined as:

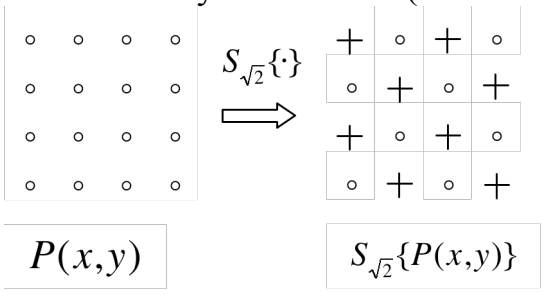
$$S_{\sqrt{2}^k}\{P\{x,y)\} = \begin{cases} P(x,y) & \text{if } (x+y) \text{ Mod } 2^{k/2} = 0 \\ 0 & \text{otherwise} \end{cases}$$

and $E_{\sqrt{2}}^{k}\{-\}$ is an diagonal expansion operator shown in figure 2 and defined as:

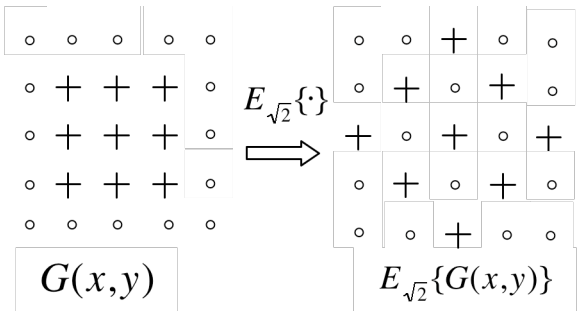
$$E_{\sqrt{2}^k}\{G\{x,y)\} = \begin{cases} G(\frac{x+y}{2},\frac{x-y}{2}) & \text{if } (x+y) \text{ Mod } 2^{k/2} = 0 \\ 0 & \text{otherwise} \end{cases}$$

The k=0 image may be discarded or used for estimating a Laplacian image for k=1 if required. Because the k=1 image has been smoothed with a Gaussian low-pass filter of scale $\sigma = 2$, resampling with a sample distance of $\sqrt{2}$ will result in an aliasing of less than 1% of signal energy.

The $\sqrt{2}$ resampling Operator, $S_{\sqrt{2}}\{\}$, selects even columns of even rows and odd columns of odd rows. For k even, diagonal sample operator eliminates every second column (starting with even columns on even rows and odd columns on odd rows). For k odd, resampling eliminates every second row (odd rows).

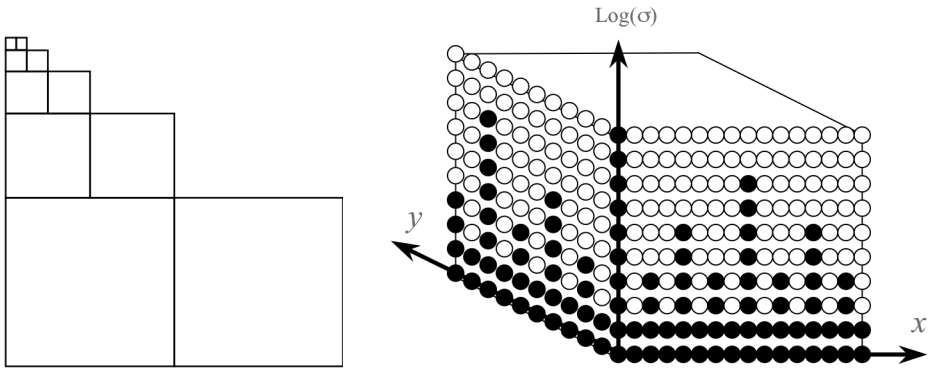


The $\sqrt{2}$ expansion operator, $E_{\sqrt{2}}\{\}$, maps rows of a filter onto diagonals, increasing sample distance by $\sqrt{2}$

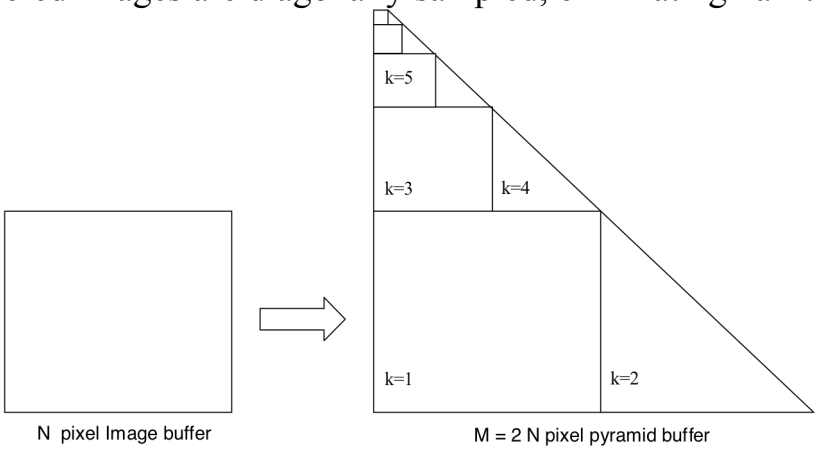


Because the sample distance scales with the cumulative variance, the effect is as an identical impulse response at every level of the pyramid, resulting in a perfectly scale invariant descriptor with a distance between samples of $s = 2^{\frac{k-1}{2}}$ for each image, p(x,y,k). Gaussian derivatives are easily calculated in the row and column directions from these images using differences of adjacent pixels

Data Structure



The even numbered images are diagonally sampled, eliminating half the pixels.



For an image of size NxN: Pyramid has PN^2 samples:

where $P = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots = 2$

2.4 Scale Invariant Interest Points

It is common to detect "keypoints" as maxima in :

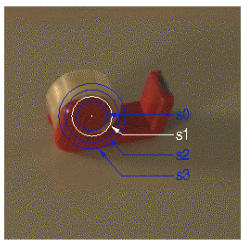
Recall

$$\nabla^2_\sigma P(i,j) = P * \nabla^2 G(i,j,\sigma) = \begin{pmatrix} P * \nabla^2 G(i,j,\sigma) \\ P * \nabla^2 G(i,j,\sigma) \end{pmatrix} \approx P * \nabla^2 G(i,j,\sigma_1) - P * \nabla^2 G(i,j,\sigma_2)$$

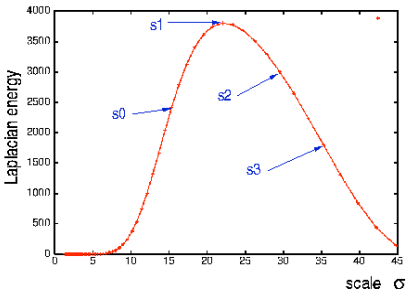
Thus for $\sigma_1/\sigma_2 = \sqrt{2}$

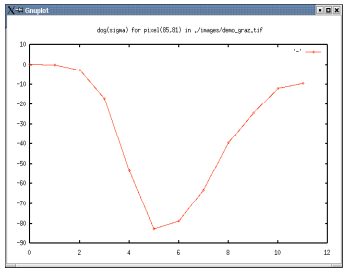
$$\nabla^2_{\sigma=2^{k/2}} P(i,j) \approx P(i,j,k) - P(i,j,k-1)$$

At every point in the image, there will be some value of σ for which $\nabla_\sigma^2 p(i,j)$ will be a maximum in σ.



zero crossing of Laplacian at s1





The scale σ_i is an "invariant" for the appearance at P(i,j).

$$\sigma_i = \arg\max_\sigma \{P * \nabla^2 G(i,j,\sigma)\}$$
$$\sigma_i = \arg\max_\sigma \{\nabla^2_{\sigma=2^k} P(i,j)\}$$
$$\sigma_i = \arg\max_k \{P(i,j,k) - P(i,j,k-1)\}$$

Maximally stable invariant points are found as :

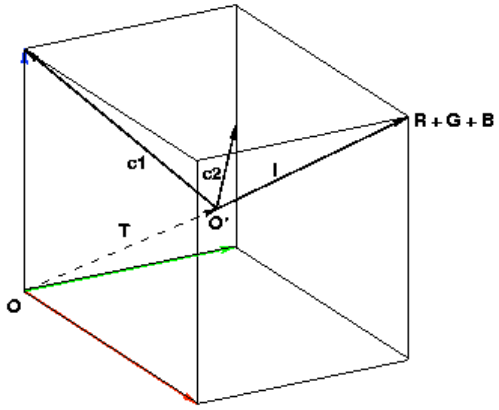
$$X(i,j,k) = \arg\max_{i,j,k} \{P(i,j,k) - P(i,j,k-1)\}$$

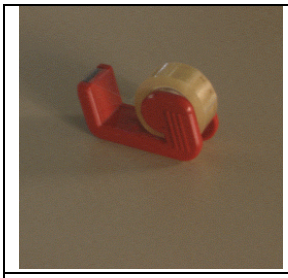
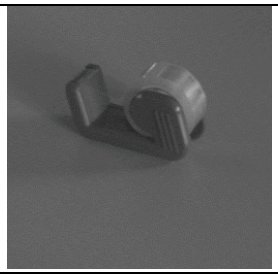
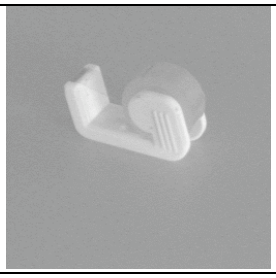
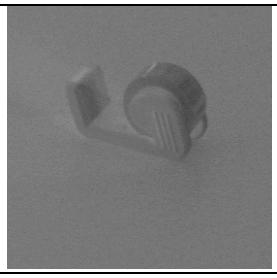
2.5 Color Opponent Scale Space

(R, G, B) ⇒ (L, C_1, C_2)

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.33 & 0.33 & 0.33 \\ -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & 0 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

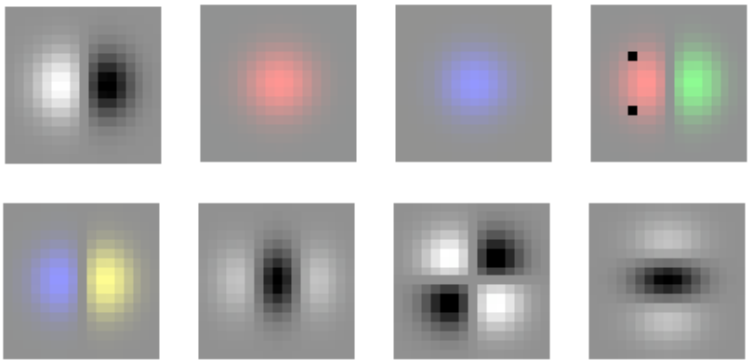
This representation separates luminance and chrominance.



				
RGB		B-W	R-G	R+G-B

This makes it possible to "steer" the chrominance to an illumination color

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.33 & 0.33 & 0.33 \\ -0.5 & -0.5 & 1 \\ 0.5 & -0.5 & 0 \end{pmatrix} \begin{pmatrix} \alpha_1 R \\ \alpha_2 G \\ \alpha_3 B \end{pmatrix}$$



$$\vec{\varphi} = \begin{bmatrix} \varphi_0 \\ \vdots \\ \varphi_k \end{bmatrix} = \begin{bmatrix} G_x^L \\ G^{C_1} \\ G^{C_2} \\ G_x^{C_1} \\ G_x^{C_2} \\ G_{xx}^L \\ G_{xy}^L \\ G_{yy}^L \end{bmatrix}$$