Computer Vision

MSc Informatics option GVR

James L. Crowley

Fall Semester 21 October 2009

Lesson 5

Describing Local Appearance with Gaussian Derivatives and Scale Space

Lesson Outline:

1 Describing Local Appearance

Appearance is what you see.

We seek a set of K local basis functions, $d_k(x,y)$ to describe "appearance" around a point in an image. Each basis function, $d_k(x,y)$ gives an image "feature", x_k, describing appearance at the pixel (x_o, y_o).

$$x_k = \sum_{x=-R}^{R} \sum_{y=-R}^{R} p(x_o + x, y_o + y) d_k(x,y)$$

Projection of the image neighborhood $P(x_o,y_o)$ onto this set of functions gives a "feature" vector for appearance, $\vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_K \end{pmatrix}$

Ideally these functions should be orthogonal

$$\sum_{x=-R}^{R} \sum_{y=-R}^{R} d_m(x_o + x, y_o + y) d_n(x,y) = 0 \quad \text{if } n \neq m$$

to minimize the number of features. This is not strictly necessary.
Such a basis can be obtained by a Taylor series representation of the image function P(x) around a point, a:

$$f(x) = f(a) + \frac{1}{1!} f_x (x-a) + \frac{1}{2!} f_{xx} (x-a)^2 + \frac{1}{3!} f_{xxx} (x-a)^3 + \dots$$

The basis set for a taylor series is the series of local derivatives.
This is called the "local jet". The local jet is a basis for describing appearance.

Because the image is a sampled signal, we need a kernel filter to compute the local Jet. The Gaussian function provides a Kernel with many useful properties.

2 Gaussian Derivative Operators

2.1 The Sampled Gaussian Functions

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

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

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

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!)

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

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.

2.2 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.

$G(x, \sigma)$

Because we can control scale with σ, we can set the sample size to $\Delta x = \Delta y = 1$.
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 $W_N(n)$

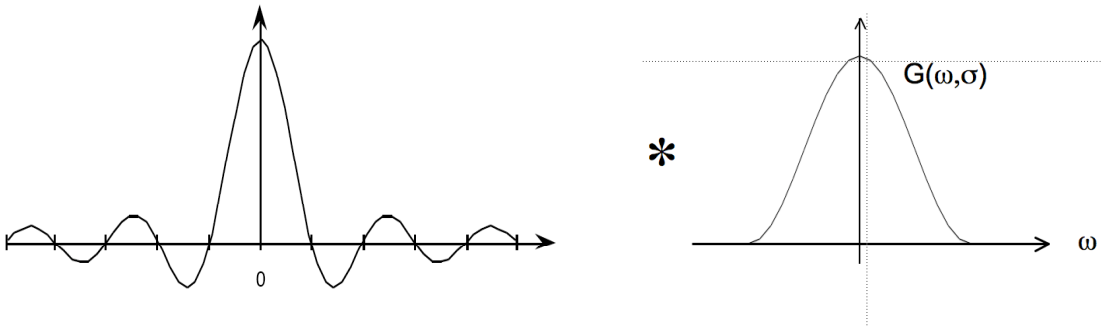
When we limit $G(x,\sigma)$ to a finite support, we multiply by a window

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

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

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

2.3 Properties of Gaussian Receptive Fields (or Wavelets)

Separability:

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

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

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

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

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

This typically requires $\sigma_1 \geq 1.4\,\sigma_2$

Thus it is common to use:

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

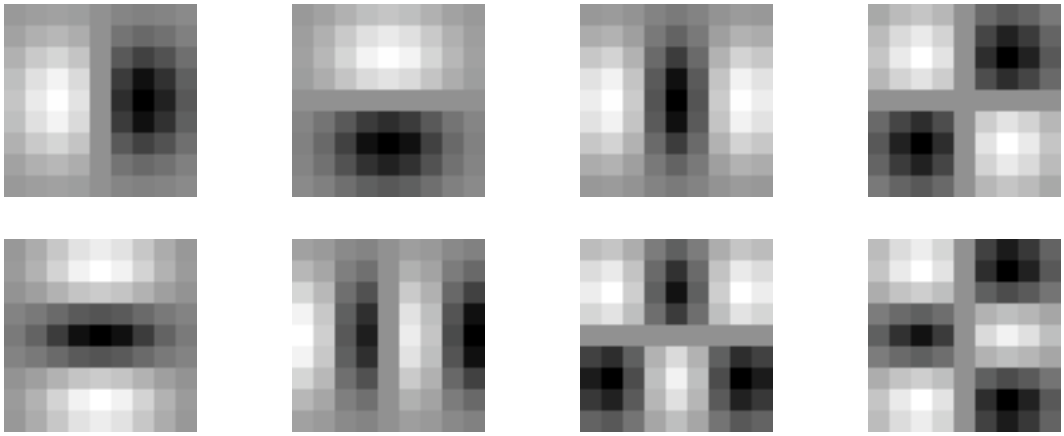
But note that :

$$G(x,y,\sqrt{2}\sigma) \approx G(x,y,\sigma) * G(x,y,\sigma)$$

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 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.

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.

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

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

2.5 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

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

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.

3 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).

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

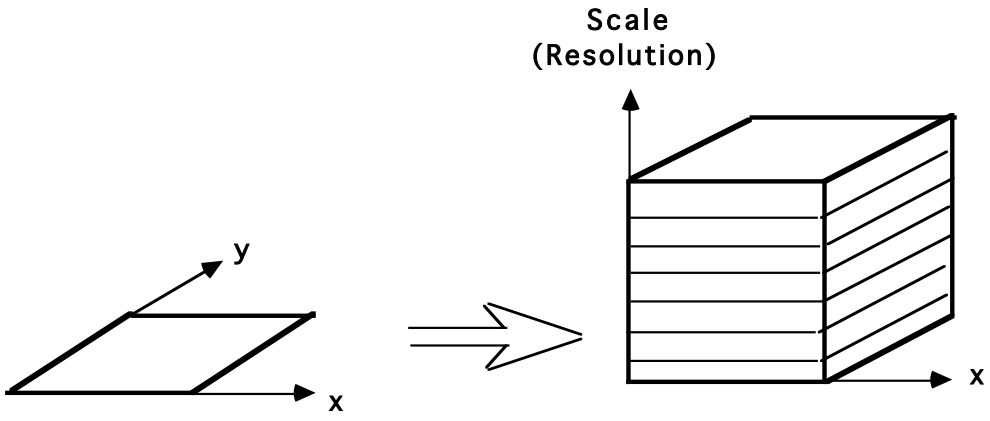
In this way, doubling the size of x(t) translates the signal by s+1.

3.1 Image Scale Space:

Continuous Case.

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

Continuous x, y, s: P(x, y, s) = P(x,y)* G(x, y, 2^s)



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.

3.2 Discrete Scale Space

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

We propose to sample scale a $\Delta\sigma = 2^{1/2}$ so that $\sigma_k=2^{k/2}$

Let G(i,j,k) be a kernel filter for $\sigma_k=2^{k/2}$

The filter support is $R_k = 4\sigma_\kappa$ so that $N_k = 2R_k +1 = 2(2^2)2^{k/2}+1$

$N_k=2^{k/2+3}+1$

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

for $1 \leq k \leq M-4$

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

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:

$\nabla^2 P(i,j,k) = \langle p(i,j), \nabla^2 G(i,j,\sigma_k) \rangle \approx P(i,j,k) - P(i,j,k-1)$

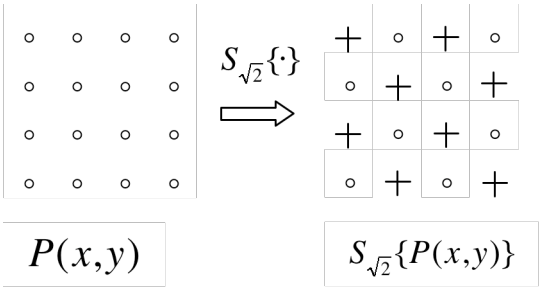
3.3 Resampled Pyramid

Discrete scale space is highly redundant. The low resolution images may be represented by a subset of pixels.

Because G(i,j,k) is a low pass filter, we can resample each pyramid image, P(i,j,k) at $\Delta x_k = \sigma_k/2=2^{(k-1)/2}$ with aliasing of less than 1% of signal energy.

for k odd, $\Delta x_k = \{1, 2, 4, 8\ldots\}$
for k even, $\Delta x_k = \{\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 8\sqrt{2}, \ldots\}$

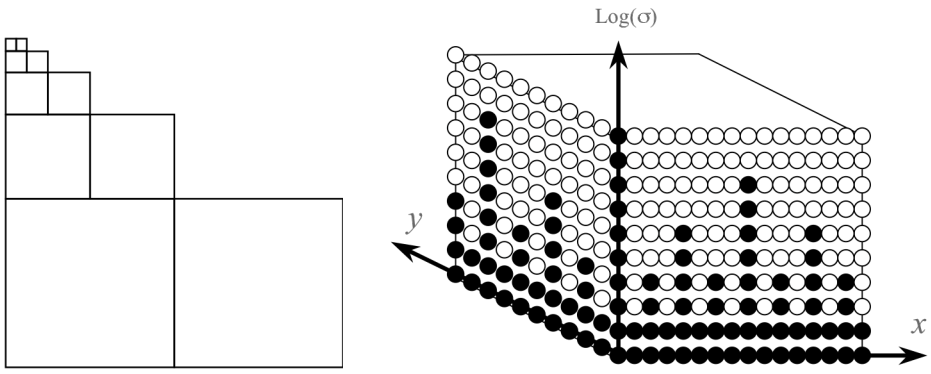
How ? with the diagonal sampling operator $S_{\sqrt{2}}\{\}$



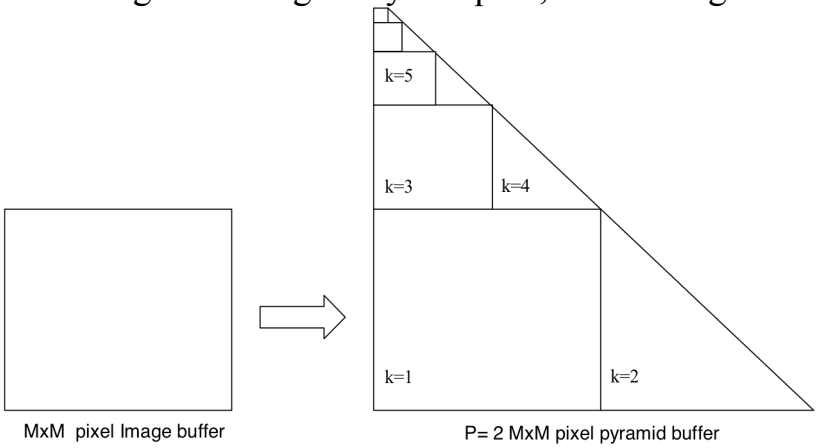
For k even, the √2 resampling operator, $S_{\sqrt{2}}{}^{k}\{\}$, selects even columns of even rows and odd columns of odd rows.
For k odd, diagonal sample operator eliminates every second column (starting with even columns on even rows and odd columns on odd rows). For k odd, resampling eliminates every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

$$S_{\sqrt{2}^k}\{P\{x,y)\} = \begin{cases} P(x,y) & \text{if } (x+y)^2 \text{ Mod } 2^{k-1} = 0 \\ 0 & \text{otherwise} \end{cases}$$

Data Structure



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



For an image of size MxM, Number of pixels is

$$P = MxM \times (1 + ½ + ¼ + \ldots) = 2M^2$$

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 Cascade Convolution Methods to reduce cost

Within such a structure, the derivatives can be approximated as differences:

$$P_x(i,j,k) =< P(i,j), G_x(i,j,\sigma) \approx P(i+\Delta x_k, j, k) - P(i-\Delta x_k, j, k)$$

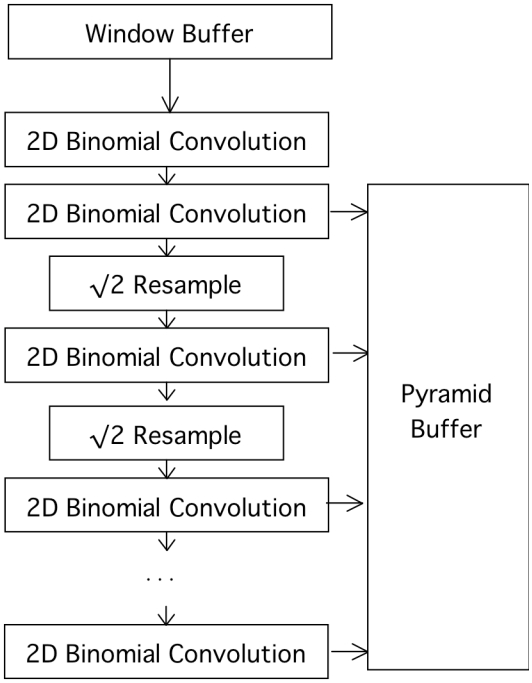
$$P_y(i,j,k) =< P(i,j), G_y(i,j,\sigma) \approx P(i, j+\Delta x_k, k) - P(i, j+\Delta x_k, k)$$

$$P_{xx}(i,j,k) =< P(i,j), G_{xx}(i,j,\sigma) \approx P(i+\Delta x_k, j, k) - P(i,j,k) + P(i-\Delta x_k, j, k)$$

$$\begin{aligned} P_{xy}(i,j,k) &=< P(i,j), G_{xx}(i,j,\sigma) \\ &\approx P(i+\Delta x_k, j+\Delta x_k, k) - P(i-\Delta x_k, j+\Delta x_k, k) \\ &\quad - P(i+\Delta x_k, j-\Delta x_k, k) + P(i-\Delta x_k, j-\Delta x_k, k) \end{aligned}$$

$$P_{yy}(i,j,k) =< P(i,j), G_{yy}(i,j,\sigma) \approx P(i, j+\Delta x_k, k) - P(i,j,k) + P(i, j+\Delta x_k, k)$$

3.4 Cascade Convolution Pyramid Algorithm:

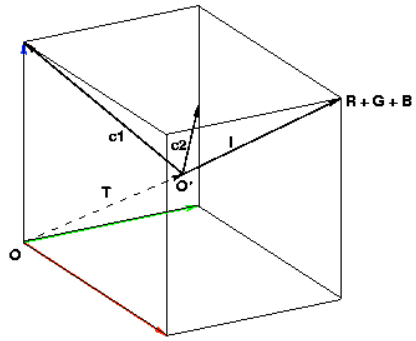


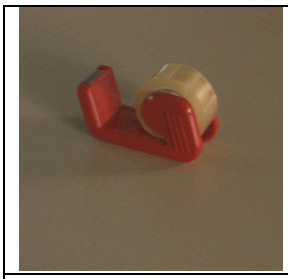
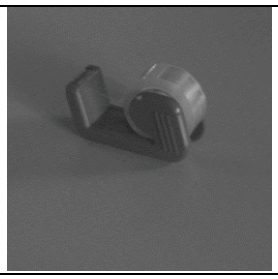
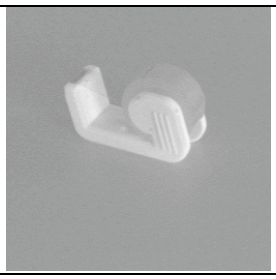
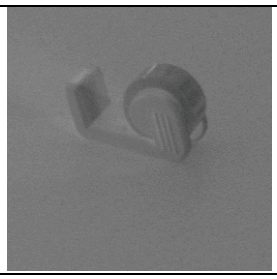
3.5 Color Opponent Scale Space

$(R, G, B) \Rightarrow (L, C_1, C_2)$

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

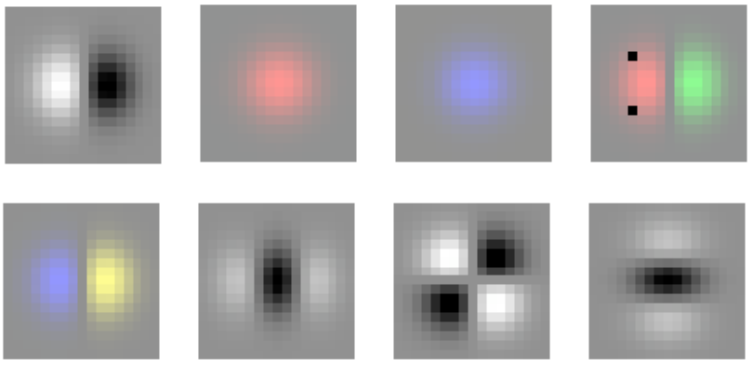
This representation separates luminance and chrominance.



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

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

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



We then compute 3 pyramids : L(i,j,k), C_1(i,j,k), and (i,j,k),

This gives us a feature vector for appearance:

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

3.6 Scale Invariant Interest Points

It is common to detect "keypoints" as maxima in the lapacian.

Recall

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

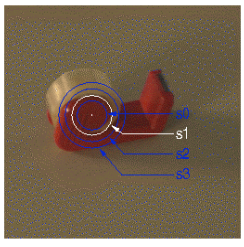
Thus for $\sigma_1/\sigma_2 = \sqrt{2}$

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

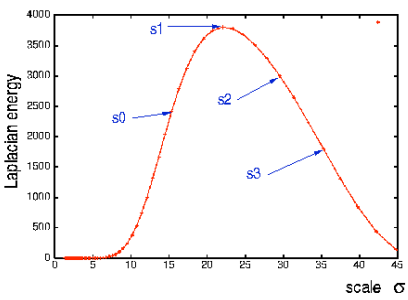
We can detect scale invariant keypoints as

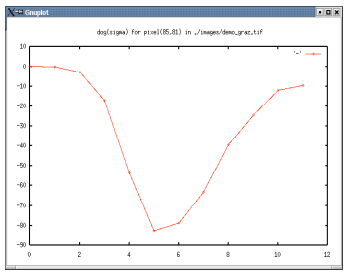
$$(i,j,k)_n = \arg\min_{i,j,k} \{\nabla^2 P(i,j,k)\}$$

Examples:



zero crossing of Laplacian at si





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 :

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

Such points are used for tracking, for image registration, and as feature points for recognition.

3.7 Scale Invariant Feature Transform (SIFT)

Notes not yet available

3.8 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