Formation et Analyse d'Images

James L. Crowley

ENSIMAG 3 Premier Sémestre 2009/2010

Lesson 8 7 décembre 2009

Describing Local Appearance with Gaussian Derivatives and Scale Space

Lesson Outline:

1 Scale and Rotation Invariant Image Description 2
1.1 The Sampled Gaussian Functions 2
1.2 Using the Gaussian to compute image derivatives 3
1.3 Image Scale Space: 3
1.4 Discrete Scale Space - Scale invariant impulse response. 4
1.5 Diagonal, Square root of two Sampling 6
1.6 Cascade Convolution Pyramid Algorithm: 8
1.7 Scale Invariant Interest Points 10
1.8 Color Opponent Scale Space 12

1 Scale and Rotation Invariant Image Description

1.1 The Sampled Gaussian Functions

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

where

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

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

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

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

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

1.2 Using the Gaussian to compute image derivatives

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

$$P_x(i,j) \approx P * G_x(i,j,\sigma)$$
$$P_y(i,j) \approx P * G_y(i,j,\sigma)$$
$$P_{xx}(i,j) \approx P * G_{xx}(i,j,\sigma)$$
$$P_{xy}(i,j) \approx P * G_{xy}(i,j,\sigma)$$
$$P_{yy}(i,j) \approx P * G_{yy}(i,j,\sigma)$$

Note: it is NECESSARY to specify σ. Small σ is not necessarily best.

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: $\vec{\nabla}P(i,j) = \begin{pmatrix} P_x(i,j) \\ P_y(i,j) \end{pmatrix} \approx \vec{\nabla}(P * G(i,j,\sigma)) = P * \vec{\nabla}G(i,j,\sigma) = \begin{pmatrix} P * G_x(i,j,\sigma) \\ P * G_y(i,j,\sigma) \end{pmatrix}$

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

Gaussian Derivatives are Steerable:

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

Thus:

1st order $P_1^{\theta}(i,j) = Cos(\theta) P_x(i,j) + Sin(\theta) P_y(i,j)$

2nd order $P_1^{\theta}(i,j) = Cos(\theta)^2 P_{xx}(i,j) + Sin(\theta)^2 P_{yy}(i,j) + 2Cos(\theta)Sin(\theta) P_{xy}(i,j)$

3rd order

$$P_3^{\theta}(i,j) = Cos(\theta)^3 P_{xxx}(i,j) + Cos(\theta)^2 Sin(\theta) P_{xxy}(i,j) + Cos(\theta)Sin(\theta)^2 P_{xyy}(i,j) + Sin(\theta)^3 P_{yyy}(i,j)$$

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 $P_{xy}(i,j)$

1.3 Image Scale Space:

Continuous Case.

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

Image Scale space is a 3D continuous space P(x,y,s)

$P(x, y, s) = P(x,y)* G(x, y,\sigma)$

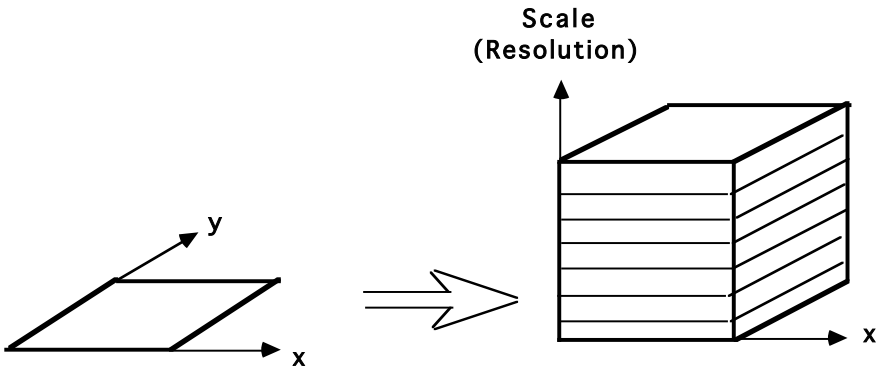
Note that the scale axis (s) is logarithmic. $s = Log_2(\sigma) = Log_2(2^s)$

Scale invariance: If a shape in an image is made larger by $D = 2^d$

$p(x,y) \rightarrow p(x2^d, y2^d)$

Then the scale space projection of appearance is shifted by s

$P(x,y,s+d) = p(x2^d, y2^d) * G(x2^d, y2^d, 2^d)$



The appearance of a pattern in the image results in a unique structure in P(x, y, s).
This structure is "equivariant" 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.

1.4 Discrete Scale Space - Scale invariant impulse response.

In a computer, we need to discretize (sample) x, y, and s.

Let P(i,j) be a discrete representation for $P(x,y) = P(i\Delta x, j\Delta x)$
Suppose P(i,j) is an image array of size M x M pixels.

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

Note that scale space "dilates" the Gaussian impulse response by 2^s.

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

As the Gaussian impulse response dilates, the sample density can also dilate.

$$P(x,y,s) = p(i\,\Delta x_k, j\,\Delta x_k, k) \quad \text{such that } \Delta x_k = 2^{k/2}$$

For a Gaussian Kernel filter $G(i,j,k) = G(x, y, \sigma_k = 2^{k/2})$

The image pyramid becomes :

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

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

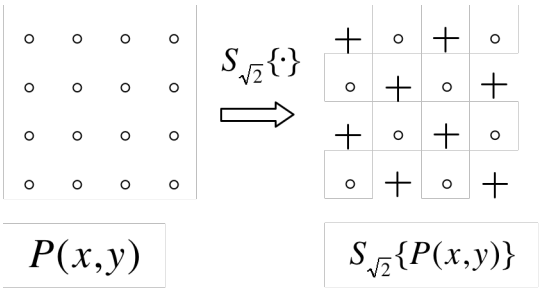
($M = \text{Log}_2(\max\{\text{width, height}\})$

1.5 Diagonal, Square root of two Sampling

Problem : How can we sample an image for for odd k? $\Delta x = 2^{k/2} = 2^{(k-1)/2}\sqrt{2}$

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 $S_{\sqrt{2}}\{\}$

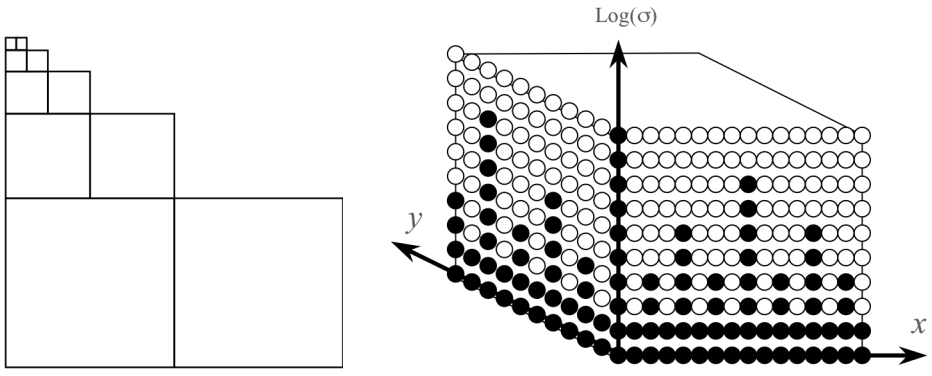


For k even, the √2 resampling operator, $S_{\sqrt{2}}{}^{k}\{\}$, selects even columns of even rows and odd columns of odd rows.

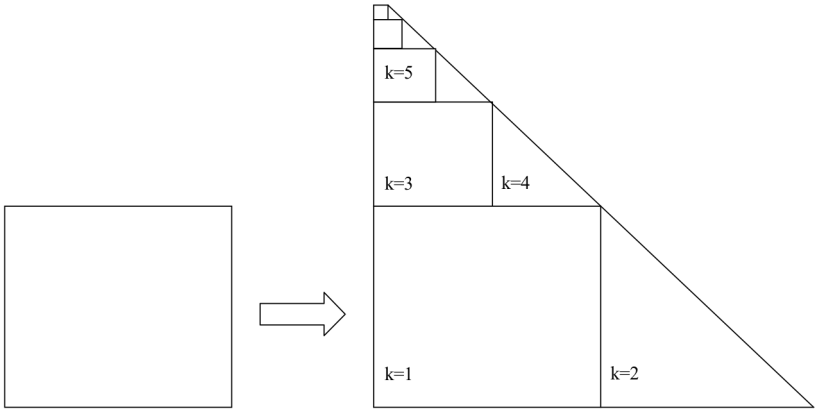
For k odd, diagonal sample operator eliminates every second column (starting with even columns on even rows and odd columns on odd rows). For k odd, resampling eliminates every second row (odd rows).

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

Data Structure



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



For an image of size MxM, number of pixels is

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

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

$$P_x(i,j,k) =< P(i,j), G_x(i,j,2^{k/2}) \approx P(i+1,j,k) - P(i-1,j,k)$$

$$P_y(i,j,k) =< P(i,j), G_x(i,j,2^{k/2}) \approx P(i,j+1,k) - P(i,j-1,k)$$

$$P_{xx}(i,j,k) =< P(i,j), G_{xx}(i,j,2^{k/2}) \approx P(i+1,j,k) - 2P(i,j,k) + P(i-1,j,k)$$

$$P_{yy}(i,j,k) =< P(i,j), G_{yy}(i,j,2^{k/2}) \approx P(i,j+1,k) - 2P(i,j,k) + P(i,j-1,k)$$

$$P_{xy}(i,j,k) =< P(i,j), G_{xy}(i,j,2^{k/2}) >$$
$$\approx P(i+1,j+1,k) - P(i-1,j+1,k) - P(i+1,j-1,k) + P(i-1,j-1,k)$$

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:

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

1.6 Cascade Convolution Pyramid Algorithm:

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:

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

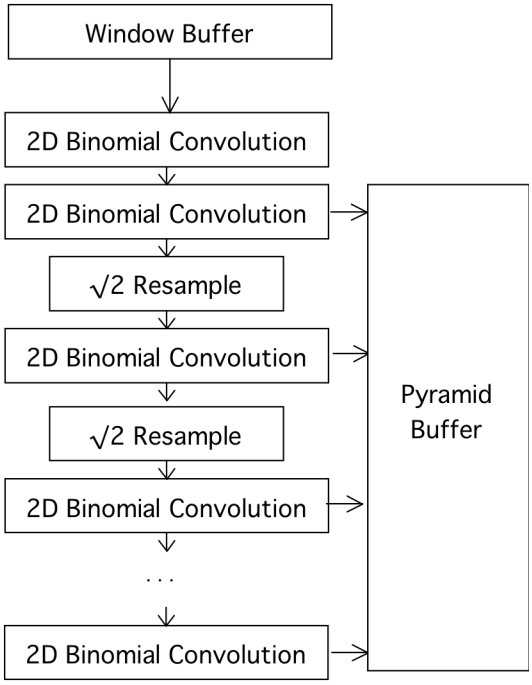
then

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

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

Practically, the computational cost is exorbitant.

We can use Cascade Convolution Methods to reduce cost to O(N)



Each binomial convolution provides convolution with G(x,y,1).

Cumulative Variance is 1,2,4,8,16,32,64,…

Cumulative standard deviation (σ) is $1, \sqrt{2}, 2, 2\sqrt{2}, 4, 4\sqrt{2}, 8, 8\sqrt{2}, 16, \ldots$

1.7 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 "keypoints"
(often called "interest points").

Recall the Laplacian of the image :

$$\nabla^2 P(x,y,s) = P * \nabla^2 G(x,y,\sigma) = P * G_{xx}(x,y,\sigma) + P * G_{yy}(x,y,\sigma) \approx P * \nabla^2 G(x,y,\sigma_1) - P * \nabla^2 G(x,y,\sigma_2)$$

Scale invariant keypoints are given by

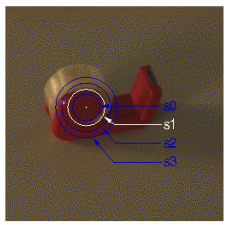
$$(x,y,s) = \arg\max_{x,y,s} \{\nabla^2 P(x,y,s)\}$$

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

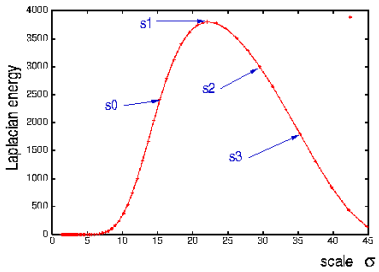
We can detect scale invariant keypoints as

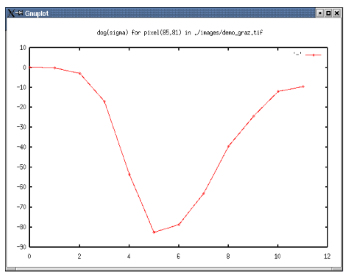
$$(i,j,k)_n = \arg\max_{i,j,k} \{\nabla^2 P(i,j,k)\}$$

Examples:



zero crossing of Laplacian at si





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.

In fact, the scale of the maximal Laplacian is an invariant at ALL image points.

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

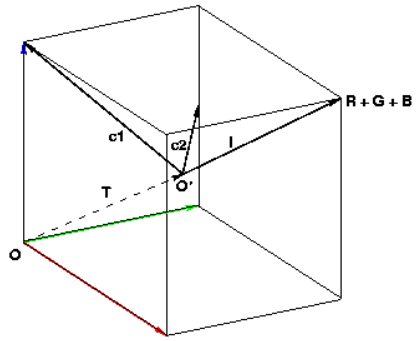
$$\sigma_i = \arg\max_{\sigma}\{P * \nabla^2 G(i,j,\sigma)\}$$
$$\sigma_i = \arg\max_{\sigma}\{\nabla^2_{\sigma=2^k} P(i,j)\}$$
$$\sigma_i = \arg\max_{k}\{P(i,j,k) - P(i,j,k-1)\}$$

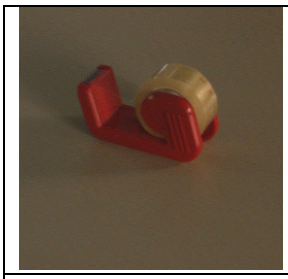
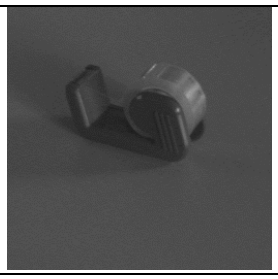
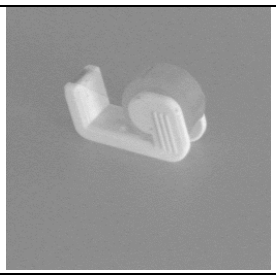
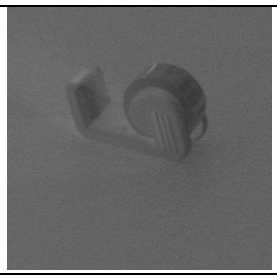
1.8 Color Opponent Scale Space

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

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

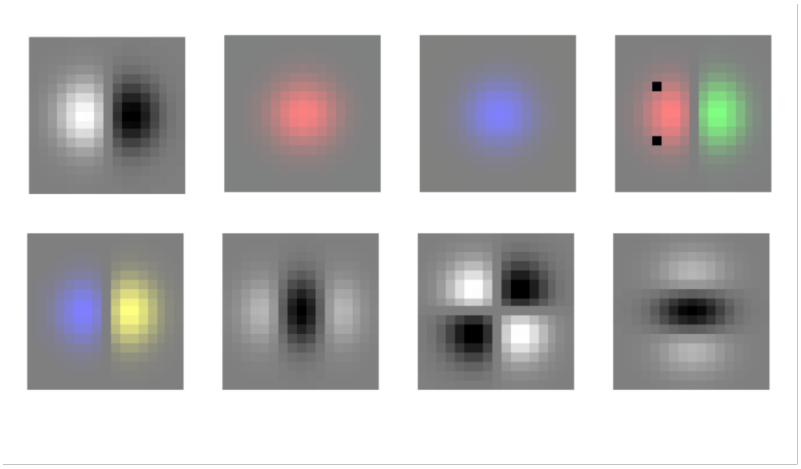
This representation separates luminance and chrominance.



				
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 C2(i,j,k),

This gives us a feature vector for appearance:

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