Formation et Analyse d'Images

James L. Crowley

ENSIMAG 3 Premier Sémestre 2008/2009

Séance 8 18 décembre 2008

Describing Contrast with Gaussian Derivatives

Lesson Outline:

1 Gaussian Derivative Operators as basis description

Appearance is what you see.

We seek a set K local basis functions, $d_k(x,y)$ to describe "appearance" around a point in an image.

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

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

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

Such a basis can be obtained by a taylor series representation of a function f(x) around a point, a, is

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

The basis set for a taylor series is the series of local derivatives.
These are used to define a basis jet for local appearance.
This is called the "local jet".

1.1 The Sampled Gaussian Functions

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

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

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

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

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

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

This factor is slightly smaller than 2πσ when the Gaussian is evaluated over a finite window *$w_R(i,j)$* of size *NxN* pixels. This factor can be determined from the filter, and can be ignored in discussing the properties of the Gaussian.

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

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

Derivative Filters:

$$G_x(i,j,\sigma) = -\frac{i}{\sigma^2} G(i,j,\sigma)$$
$$G_{xx}(i,j,\sigma) = \frac{i^2 - \sigma^2}{\sigma^4} G(i,j,\sigma)$$
$$G_{xy}(i,j,\sigma) = \frac{ij}{\sigma^4} G(i,j,\sigma)$$
$$G_{xxx}(n,\sigma) = -\frac{i^3 - i\sigma^2}{\sigma^6} G(i,j,\sigma)$$

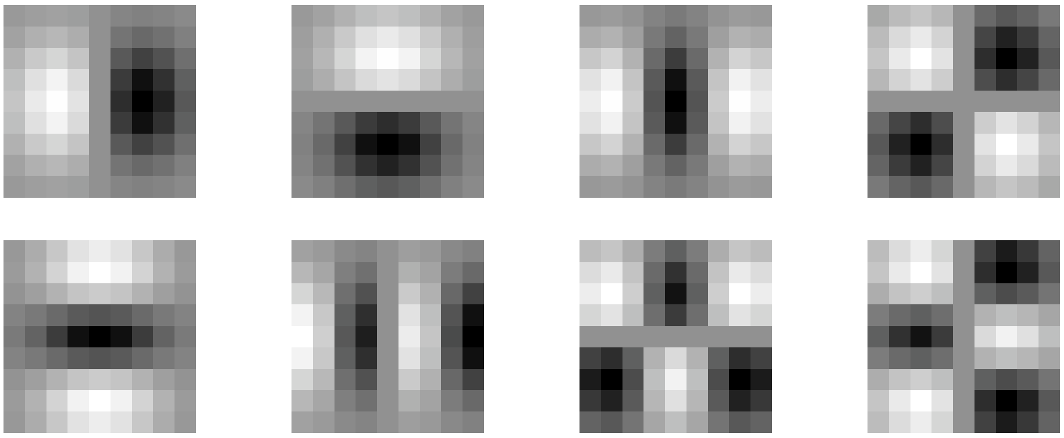
Laplacian:

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

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

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}, …

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

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

A subset of the derivatives in 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.

1.2 Using the Gaussian to compute image derivatives

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

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

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 $\qquad \vec{\nabla} G(i,j,\sigma) = \begin{pmatrix} G_x(i,j,\sigma) \\ G_y(i,j,\sigma) \end{pmatrix}$

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

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

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

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 $P_{xy}(i,j)$

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).

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

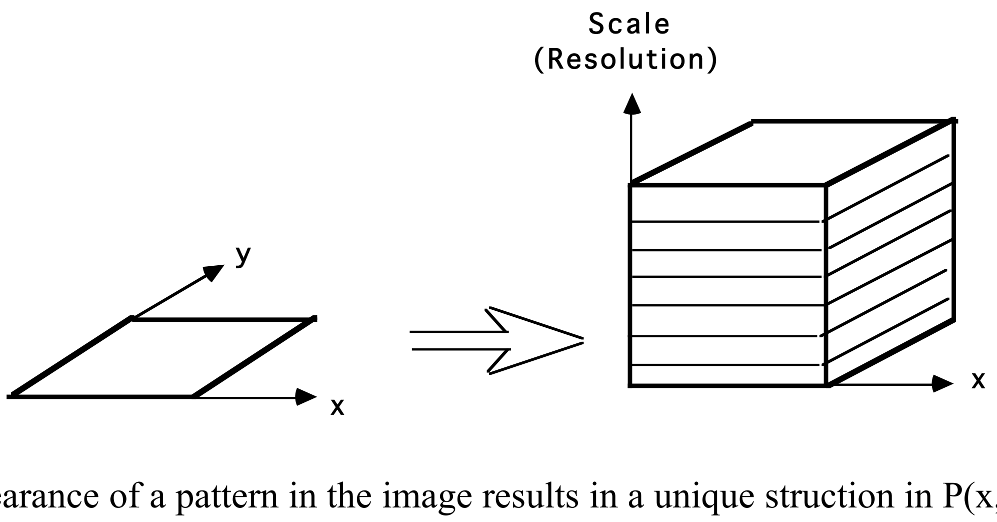
In this way, doubling the size of x(t) translates the signal by s+1.

2.1 Image Scale Space:

Continuous Case.

Let P(x,y) be the image.
Let G(x, y, 2^s) 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.

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

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:

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

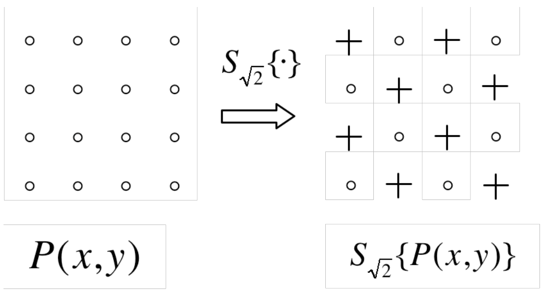
2.3 Resampled Pyramid

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, $\Delta x_k = \{\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 8\sqrt{2}, \ldots\}$

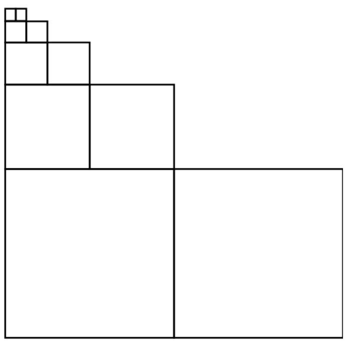
How ? with the diagonal sampling operator $S_{\sqrt{2}}\{\}$

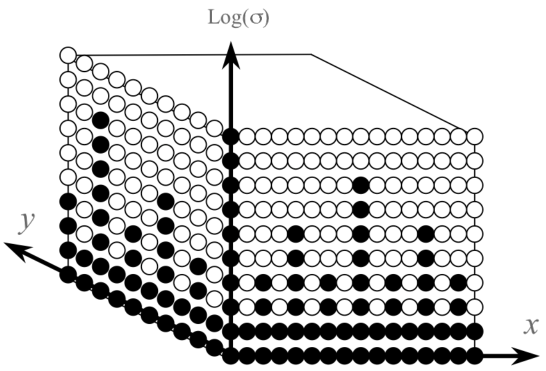


For k even, the √2 resampling operator, $S_{\sqrt{2}}{}^{k}\{\}$, selects even columns of even rows and odd columns of odd rows.
For k odd, diagonal sample operator eliminates every second column (starting with even columns on even rows and odd columns on odd rows). For k odd, resampling eliminates every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

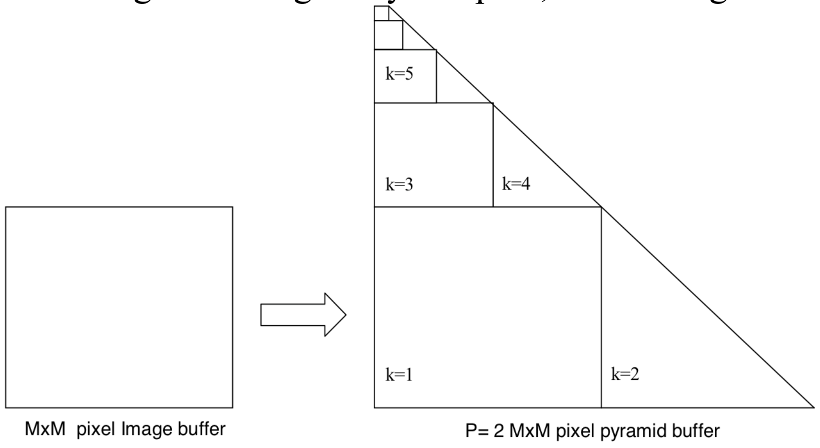
$$S_{\sqrt{2}^k}\{P\{x,y)\} = \begin{cases} P(x,y) & \text{if } (x+y)^2 \text{ Mod } 2^{k-1} = 0 \\ 0 & \text{otherwise} \end{cases}$$

Data Structure





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



For an image of size MxM, Number of pixels is

$$P = M\text{x}M \text{ x } (1 + ½ + ¼ + \ldots) = 2M^2$$

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

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

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:

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

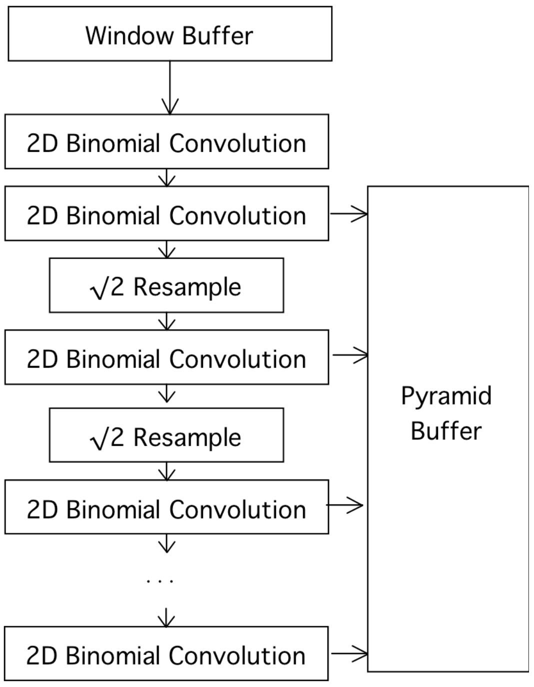
$$P_y(i,j,k) =< P(i,j), G_y(i,j,\sigma) \approx P(i,j+\Delta x_k,k) - P(i,j+\Delta x_k,k)$$

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

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

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

2.4 Cascade Convolution Pyramid Algorithm:



2.5 Scale Invariant Interest Points

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

Recall

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

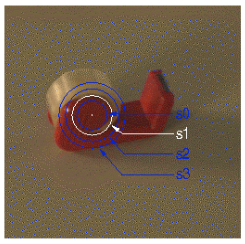
Thus for $\sigma_1/\sigma_2 = \sqrt{2}$

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

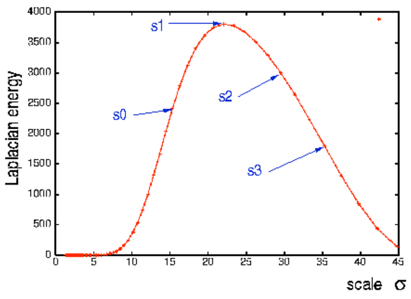
We can detect scale invariant keypoints as

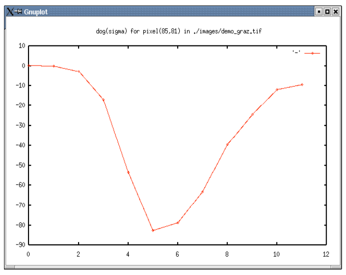
$$(i,j,k)_n = \underset{i,j,k}{\arg-\max}\{\nabla^2 P(i,j,k)\}$$

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.

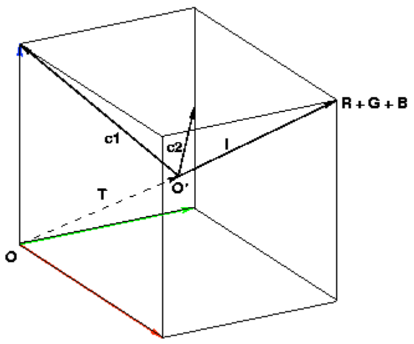
(this will be the subject of our next séances).

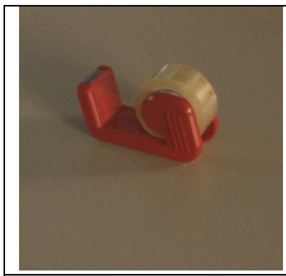
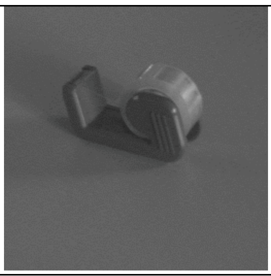
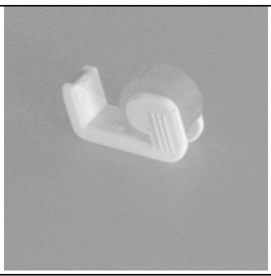
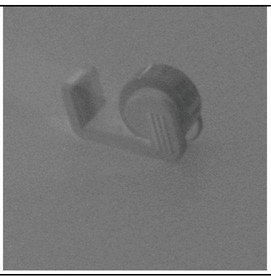
2.6 Color Opponent Scale Space

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

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

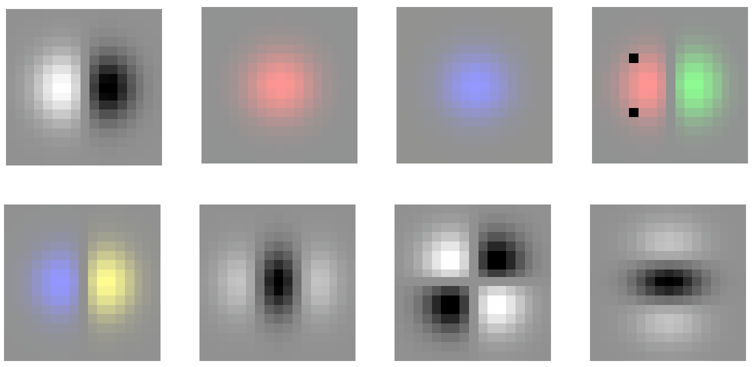
This representation separates luminance and chrominance.



				
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),