Orientation and Scale

Orientation steerability

1st derivative of a Gaussian at angle θ

$$g_x(R^T(x;y)) = \cos\theta g_x + \sin\theta g_y = (\cos\theta \ \ \sin\theta)\nabla g.$$

To compute the derivative of an image in direction θ, all we need to do is the following:

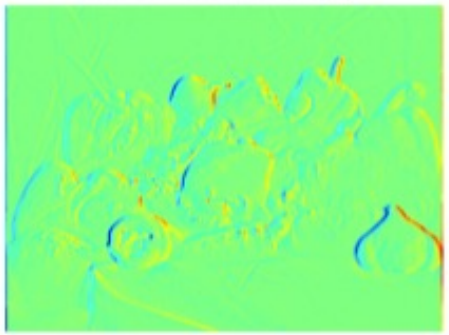
$$\cos\theta(g_x * I) + \sin\theta(g_y * I)$$

Steerability of the first Gaussian derivative Recall from the end of last lecture that the first Gaussian derivative is steerable. Indeed, if $g'(x, y, \theta)$ is the derivative of the Gaussian along direction θ, we have:

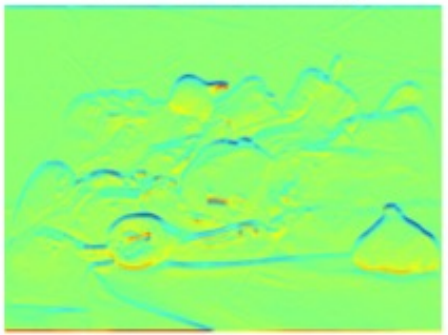
$$\begin{aligned} g'(x, y, \theta) &= \cos\theta g_x(x, y) + sin\theta g_y(x, y) \\ &= \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \underbrace{\begin{bmatrix} g_x \\ g_y \end{bmatrix}}_{\nabla G} \end{aligned}$$

This comes from the fact that derivation itself is steerable: the derivative of an image along a direction θ is simply the gradient of the image multiplied by a unit vector in direction θ: $I'(x, y, \theta) = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \nabla I$.

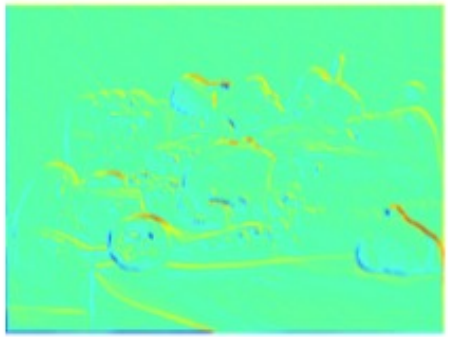
Gaussian derivative along x



Gaussian derivative along y



Derivative along theta=45deg



Derivative along theta=−30deg

Steerability of 2nd derivative

Steerability of Gaussian derivatives of higher order Remember that the m-th derivative of a Gaussian has the following Fourier:

$$\frac{\partial^m g(x,y)}{\partial x^m} \multimap\bullet (j\omega_x)^m e^{-\omega_x^2\sigma^2/2} e^{-\omega_y^2\sigma^2/2}$$

What happens when we rotate the Gaussian?

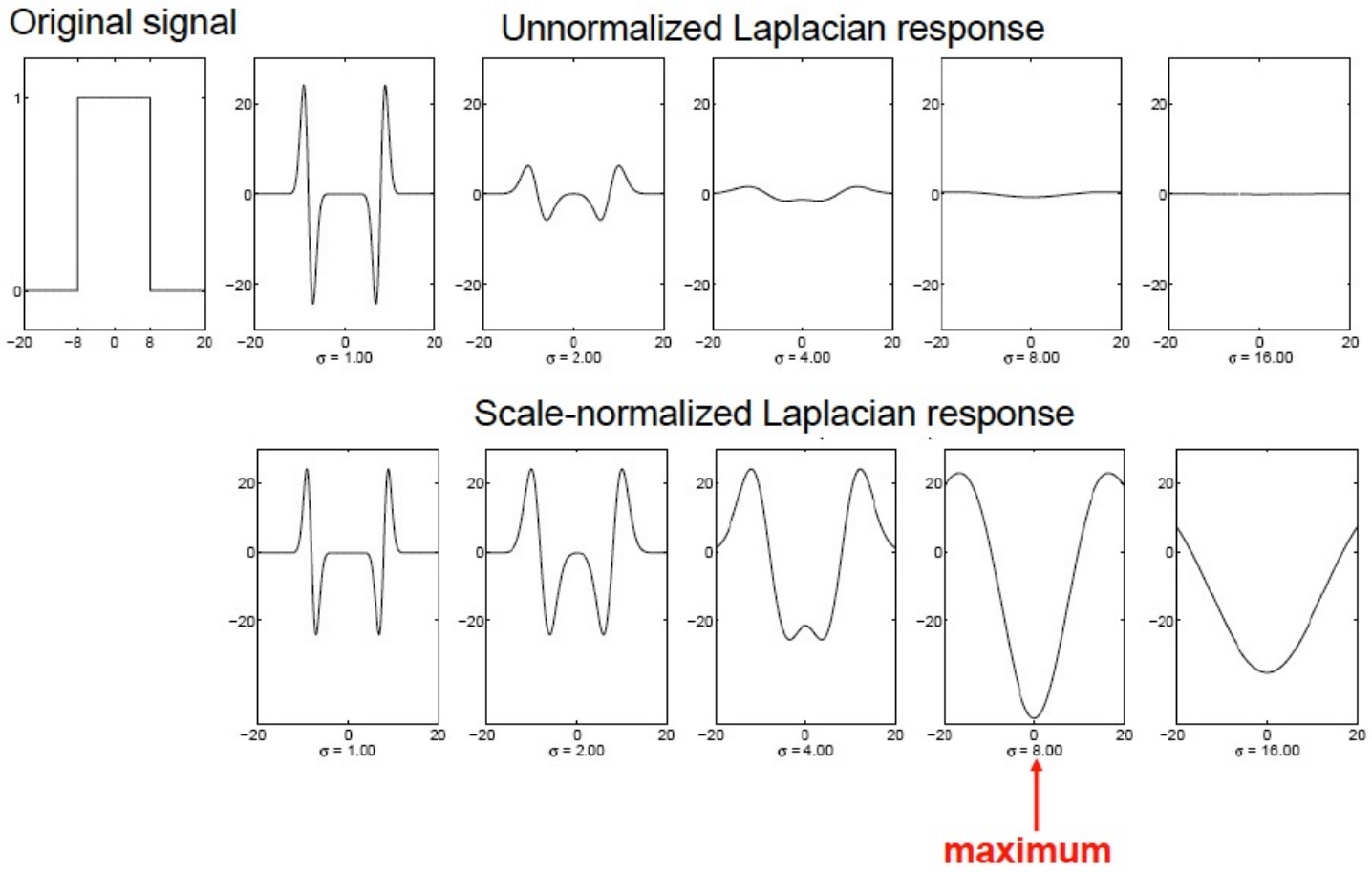
$$\frac{\partial^m g(x\cos\theta + y\sin\theta)}{\partial x^m} \multimap\bullet j^m(\omega_x\cos\theta + \omega_y\sin\theta)^m e^{-\omega_x^2\sigma^2/2} e^{-\omega_y^2\sigma^2/2}$$

For $m = 2$, we have the following steerability formula as in equation 1:

$$\underbrace{\cos^2\theta}_{A_1(\theta)}\omega_x^2 + \underbrace{\sin^2\theta}_{A_2(\theta)}\omega_y^2 + \underbrace{2\sin\theta\cos\theta}_{A_3(\theta)}\omega_x\omega_y$$

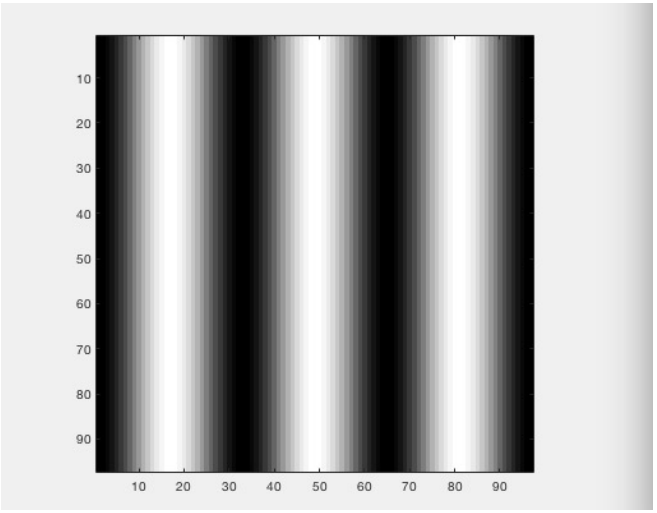
How do we deal with scale?

Effect of scale when filtering with 2nd derivative



Build a system that detects edges and keypoints independent from scale (size)

Let us look at an “unconventional edge”:



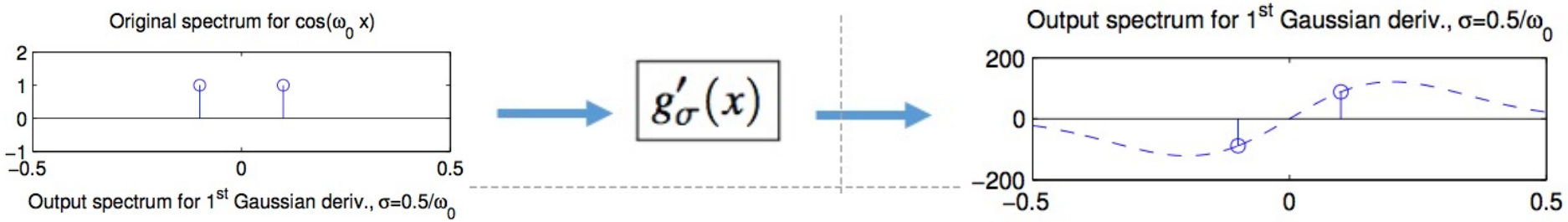
$\cos \omega_0 x$ → $g'_\sigma(x)$ →

Let is filter a cos with first derivative of Gaussian

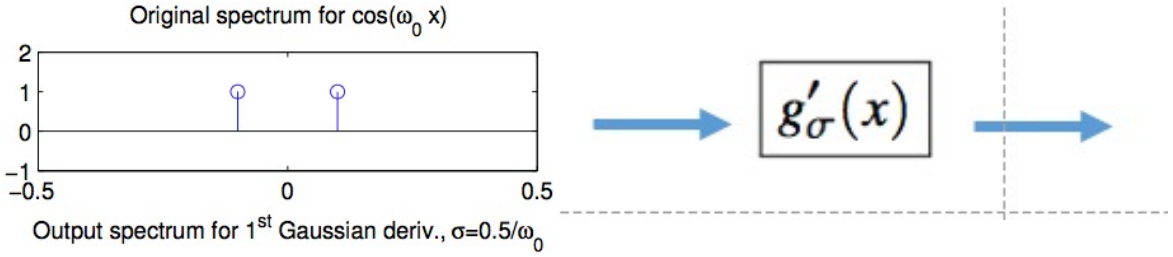
$\cos \omega_0 x$ → $\boxed{g'_\sigma(x)}$ → "smoothed derivative"

$$g'_\sigma = \frac{dg_\sigma}{dx} \text{ where } g_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

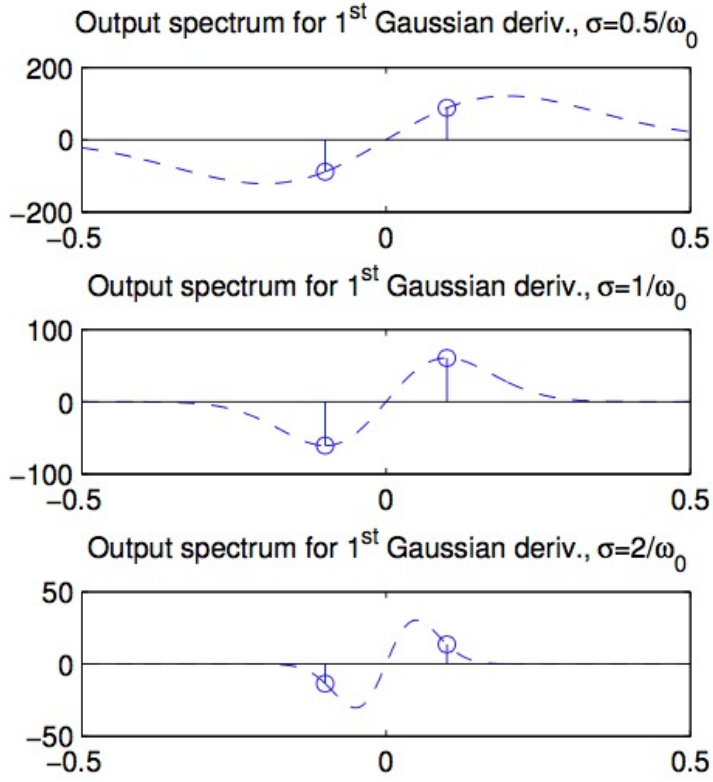
$$\tfrac{1}{2}(\delta(\omega-\omega_0)+\delta(\omega+\omega_0))\, j\omega e^{-\omega^2\sigma^2/2}$$
$$= \tfrac{1}{2}(\delta(\omega-\omega_0)-\delta(\omega+\omega_0))\, j\omega_0 e^{-\omega_0^2\sigma^2/2}$$



Response is maximum when filter scale σ matches incoming scale (ω_o).

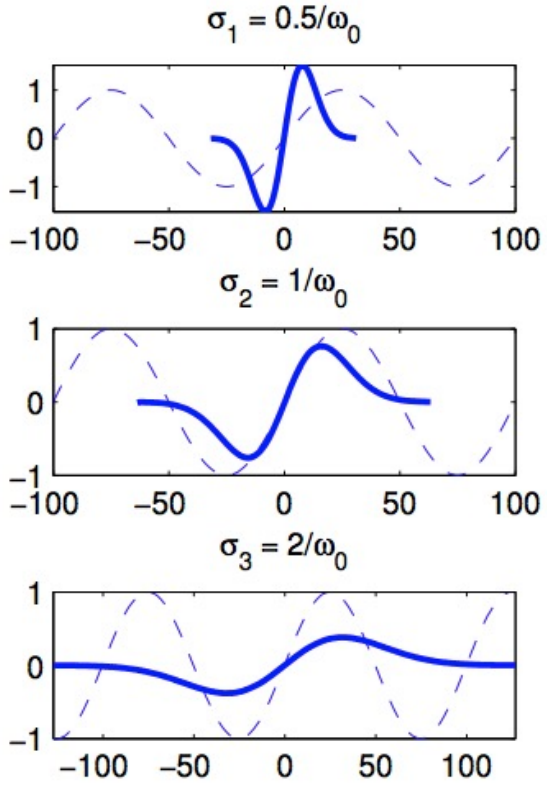


$$j\omega_0 e^{-\omega_0^2\sigma^2/2}$$



Look at edge detection as template matching

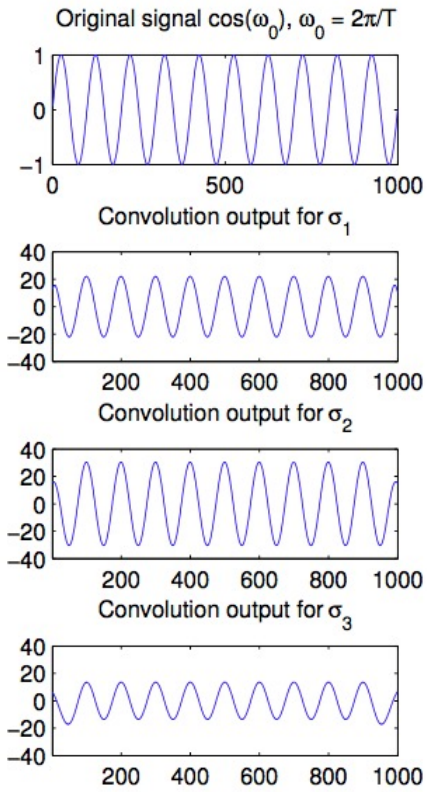
Which size of “edge template” matches the original edge?



Edge detection output for different σ's:

As σ increases the peaks at the edges weaken!!

$$\omega_0 e^{-\omega_0^2 \sigma^2 / 2}$$

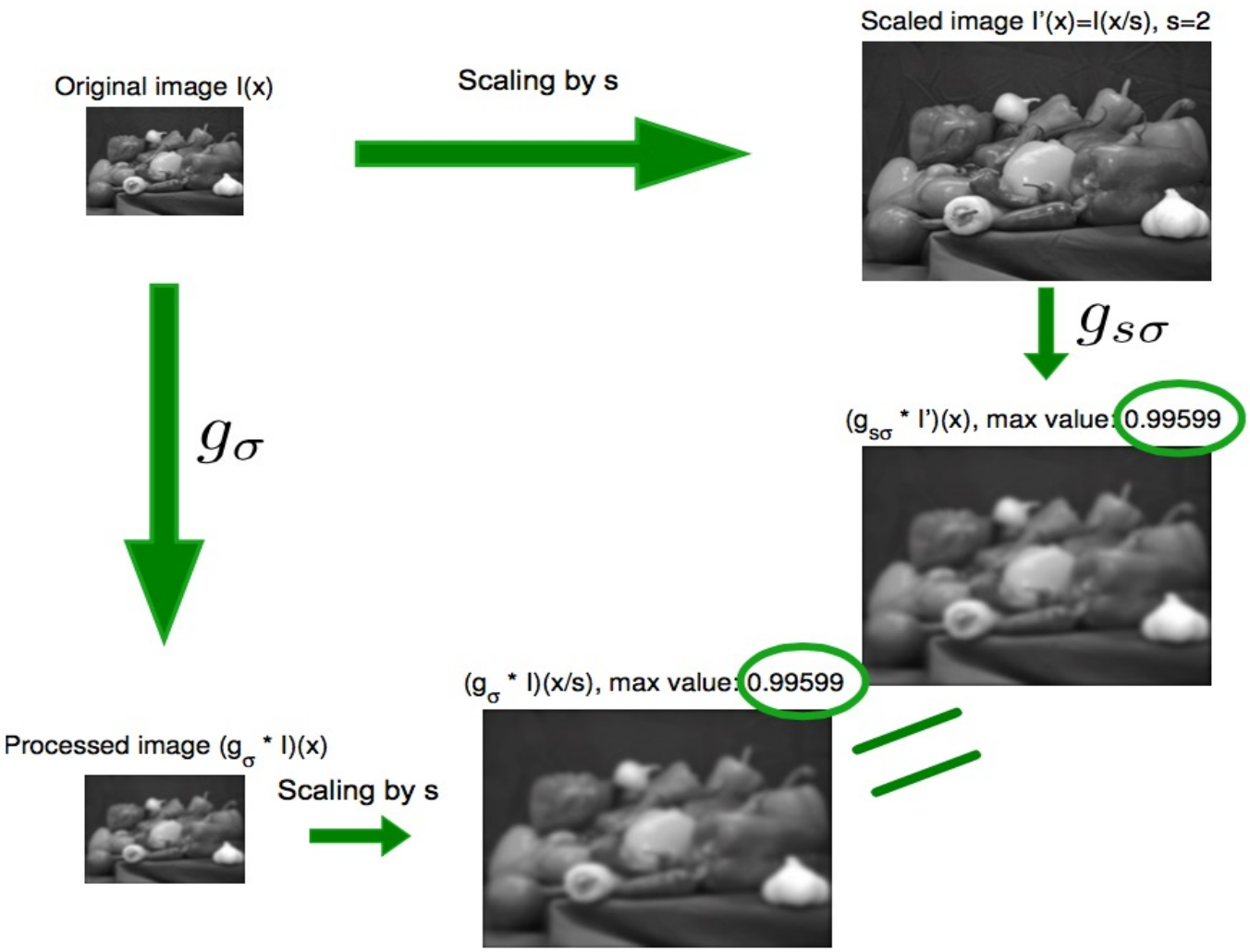


Why we need a scale normalization of the filter!

Assume a scaled version of the original signal $I'(x) = I\left(\frac{x}{s}\right)$ (twice as wide for s=2),
then calculate the convolution with Gaussian:

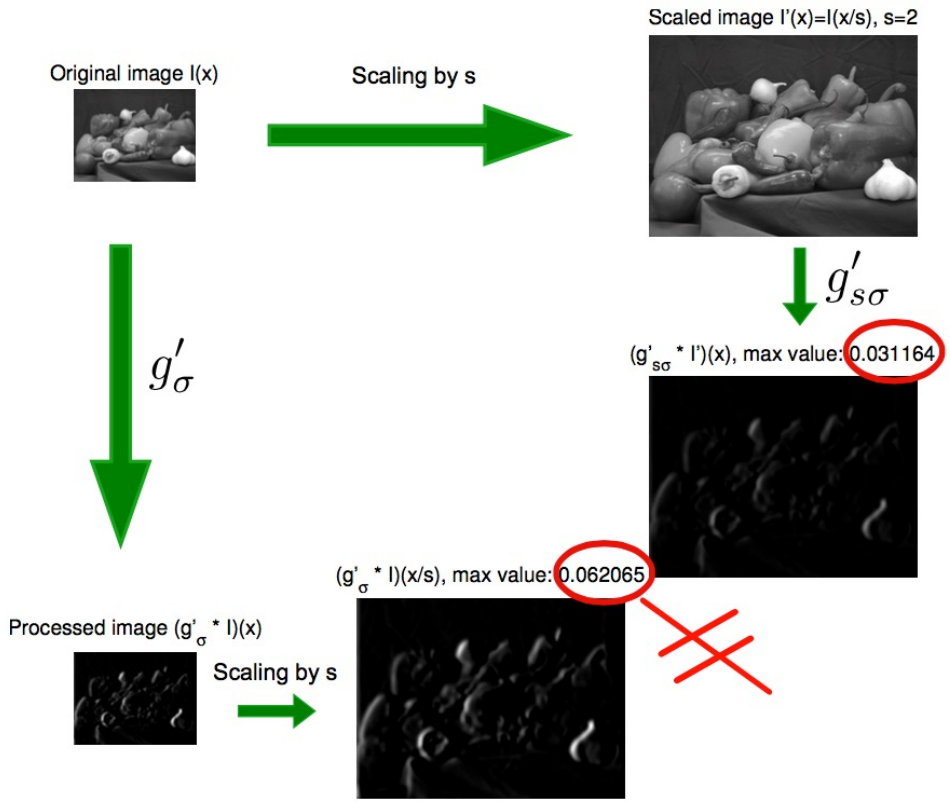
$$(I' \star g_{s\sigma})(x) = (I \star g_{\sigma})\left(\frac{x}{s}\right)$$

Scaling by s and convolving with scaled Gaussian is the same as applying a Gaussian and then scaling by s!!!

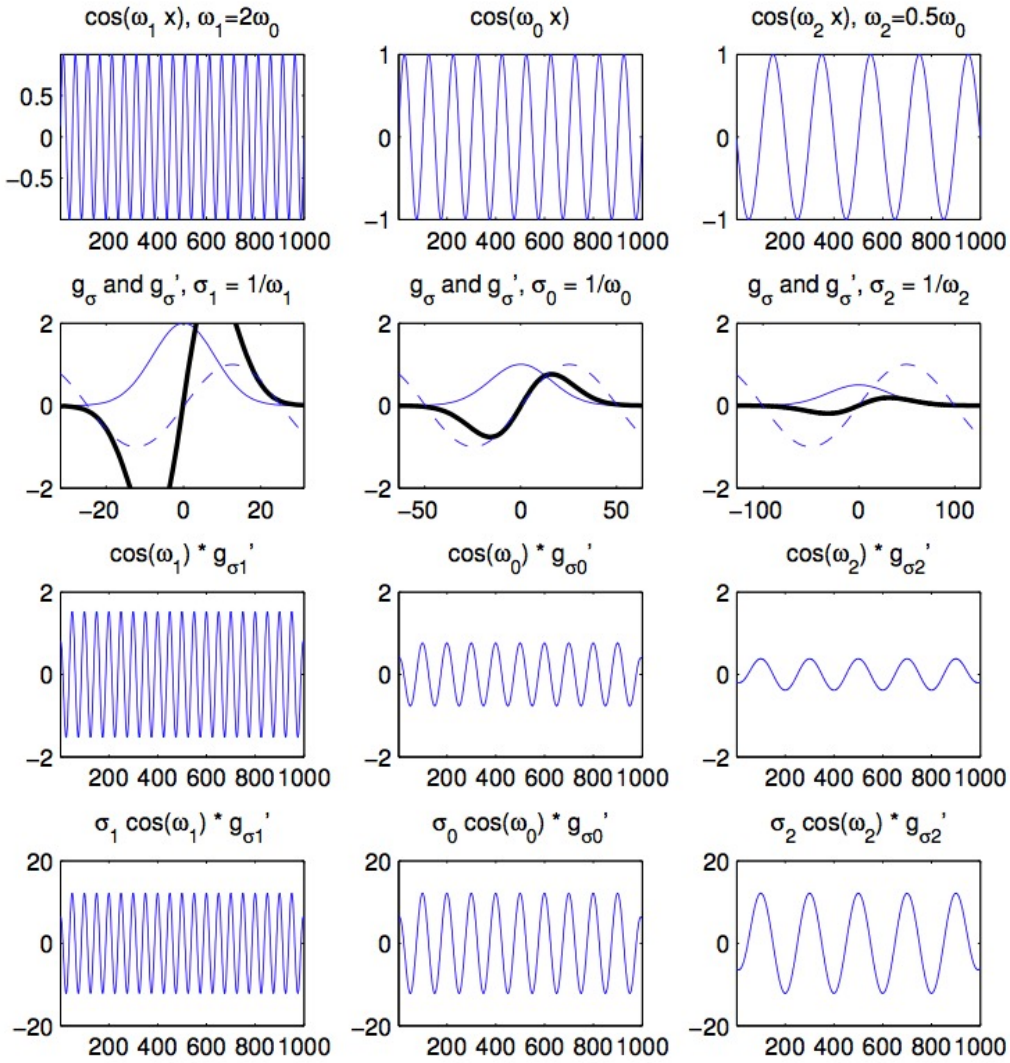

Original image I(x)
Scaling by s
Scaled image I'(x)=I(x/s), s=2
g_σ
$g_{s\sigma}$
$(g_{s\sigma} * I')(x)$, max value: 0.99599
$(g_\sigma * I)(x/s)$, max value: 0.99599
Processed image $(g_\sigma * I)(x)$
Scaling by s

But this is not true for the 1st derivative!

$$(I' \star g'_{s\sigma})(x) \neq \frac{1}{s}(I \star g'_{\sigma})\left(\frac{x}{s}\right)$$

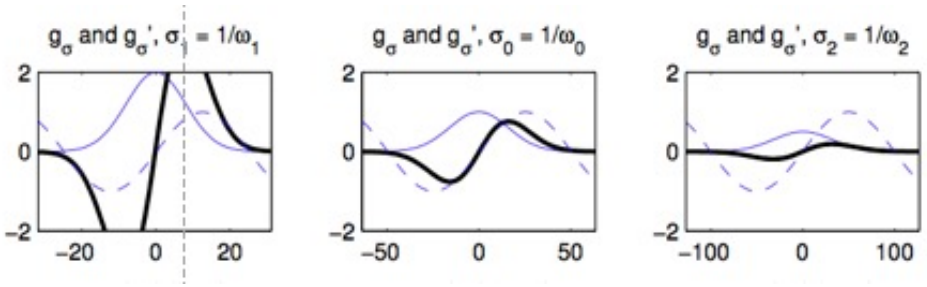


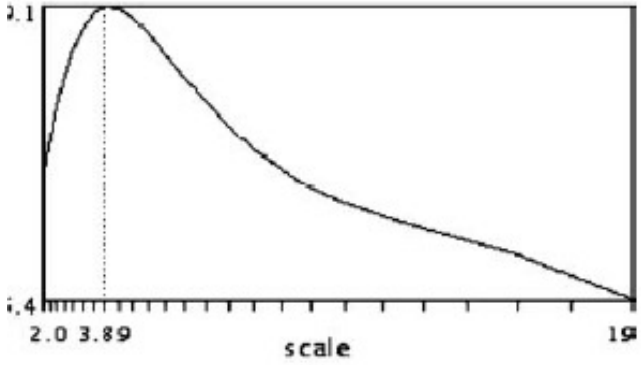
No matter what the scale of the feature, edge detector (1st Gaussian derivative) response should be the same when the filter matches the feature scale.



Can we find (select) the intrinsic image scale?

Yes, by taking the maximum over scale!





Normalize 2nd derivative and difference of Gaussians

$$\frac{\partial^2 g}{\partial \xi^2} = \frac{\partial^2 g}{\partial x^2}\frac{\partial^2 x}{\partial \xi^2} = \sigma^2 \frac{\partial^2 g}{\partial x^2}$$

$$\xi = \frac{x}{t^{\gamma/2}} = \frac{x}{\sigma} \qquad t = \sigma^2, \quad \gamma = 1$$

$$\frac{\partial g}{\partial \sigma} = \sigma \frac{1}{\sigma^2}\frac{\partial^2 g}{\partial \xi^2} = \frac{1}{\sigma}\frac{\partial^2 g}{\partial \xi^2}$$

DoG

APPROXIMATING $\frac{\partial g}{\partial \sigma}$

$$\frac{\partial g}{\partial \sigma} \approx \frac{g(x, \sigma + \Delta\sigma) - g(x, \sigma)}{\Delta\sigma}$$

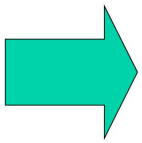
We note $\sigma + \Delta\sigma = \kappa\sigma$ ($\kappa = 1.01$ would be a good approximation), and we have:

$$\frac{\partial g}{\partial \sigma} \approx \frac{g(x, \kappa\sigma) - g(x, \sigma)}{\sigma(\kappa - 1)} = \textbf{DoG}$$

How do we build the scale space efficiently?

Binomial distribution and Pascal's Triangle

[1 1] * [1 1] --> [1 2 1]
[1 1] * [1 2 1] --> [1 3 3 1]
[1 1] * [1 3 3 1] --> [1 4 6 4 1]
..and so on…



1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
and so on…

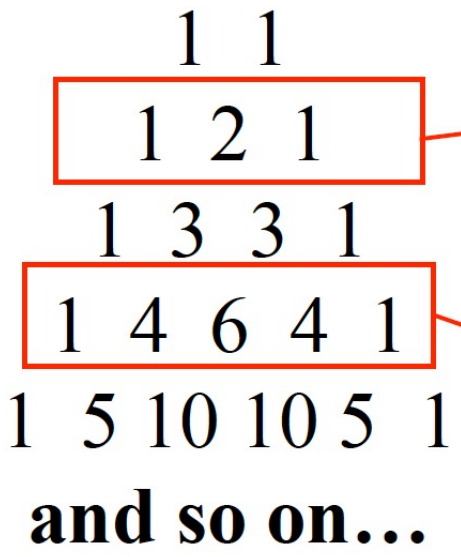
Pascal's Triangle

$$a_{nr} \equiv \frac{n!}{r!(n-r)!} \equiv \binom{n}{r}$$

n = number of elements in the 1D filter minus 1
r = position of element in the filter kernel (0, 1, 2…)

Image from Robert Collins

Look at odd-length rows of Pascal's triangle:



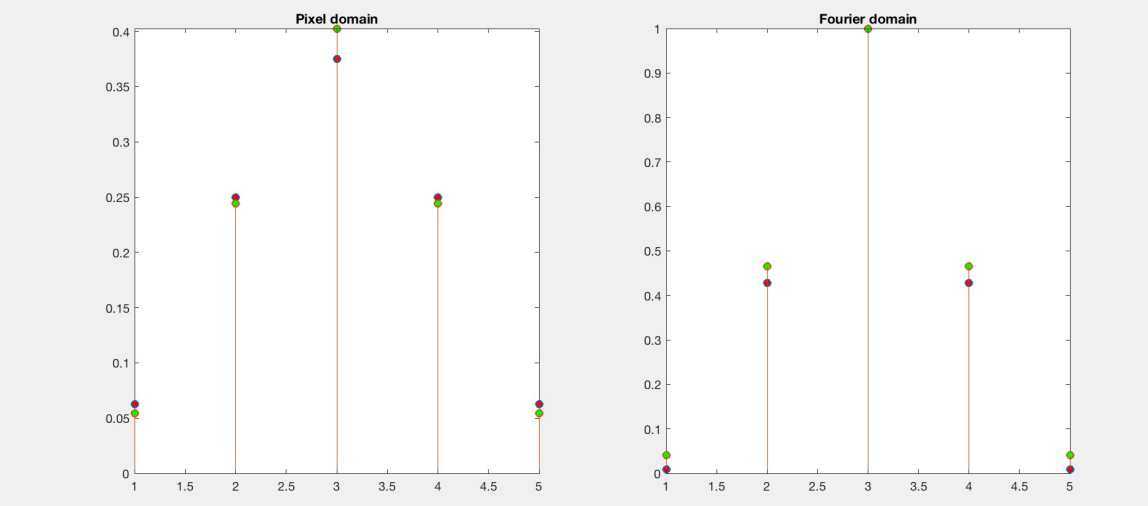
[1 2 1]/4 - approximates Gaussian with sigma=1/sqrt(2)

\begin{itemize}

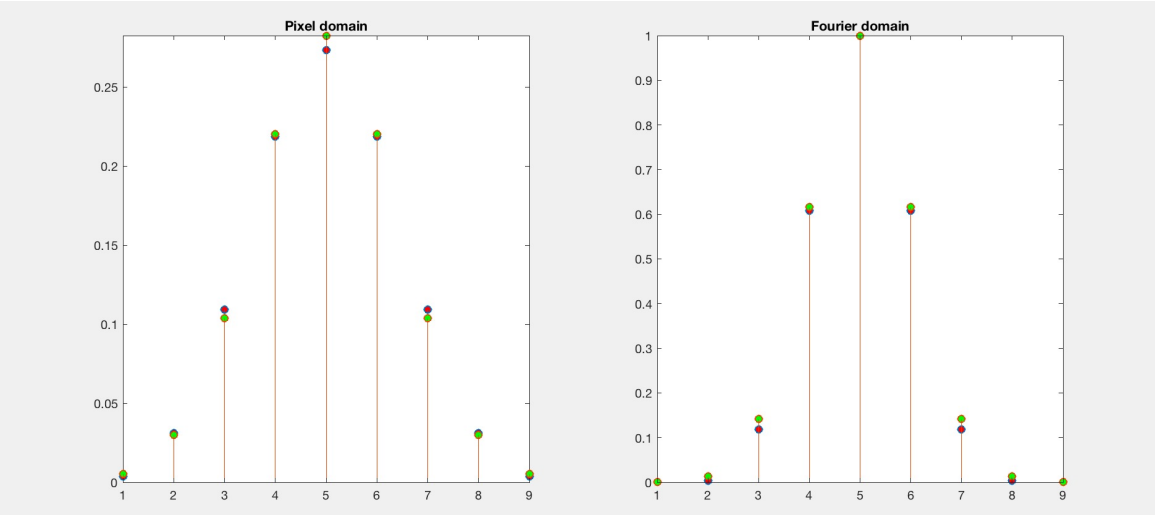
[1 4 6 4 1]/16 - approximates Gaussian with sigma=1

An easy way to generate integer-coefficient Gaussian approximations.

[1 4 6 4 1]/16



[1 4 6 4 1]/16*[1 4 6 4 1]/16*



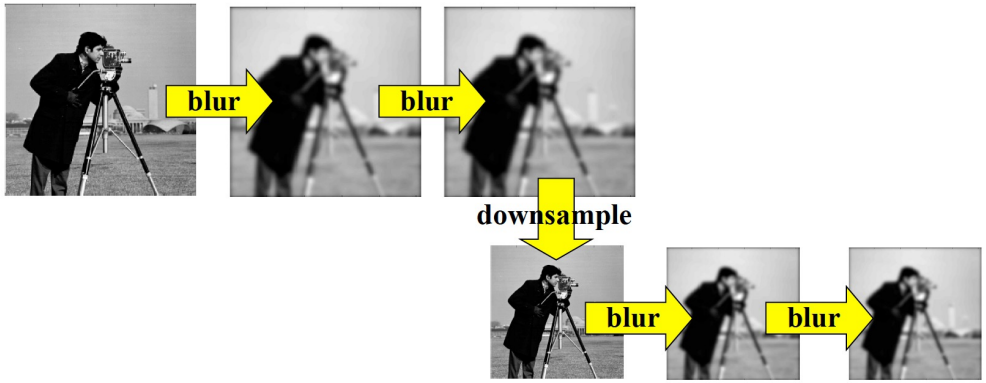

200
400
600
800
1000
1200
1400
1600
1800
2000
500
1000
1500
2000
2500
3000

How to produce two sigma's at every pyramid level

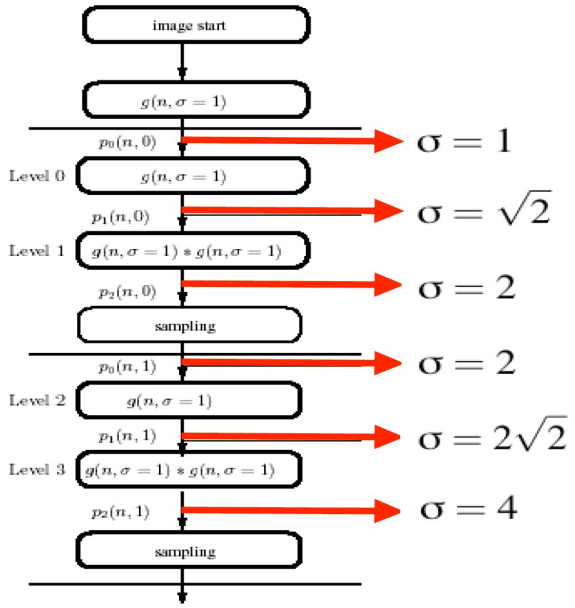
Fast Computation of Characteristic Scale Using a Half-Octave Pyramid

James L. Crowley, Olivier Riff, and Justus H. Piater
Projet PRIMA, Lab. GRAVIR-IMAG
INRIA Rhône-Alpes, France

General idea: cascaded filtering using [1 4 6 4 1] kernel to generate a pyramid with two images per octave (power of 2 change in resolution). When we reach a full octave, downsample the image.



Effective σ



Remember that LoG = DoG !

$$\nabla^2 G(r, \sigma_{\text{lap}}) = \frac{r^2 - \sigma_{\text{lap}}^2}{\sigma_{\text{lap}}^5 \sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\frac{r^2}{\sigma_{\text{lap}}^2}}$$

The difference of Gaussians is:

$$\text{DOG}(r, \sigma_{\text{dog}}) = \frac{1}{\sigma_1 \sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\frac{r^2}{\sigma_1^2}} - \frac{1}{\sigma_{\text{dog}} \sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\frac{r^2}{\sigma_{\text{dog}}^2}}$$

$$\sigma_1 = \sqrt{2}\sigma_{\text{dog}}$$

DoG vs LoG (Crowley, 2002)

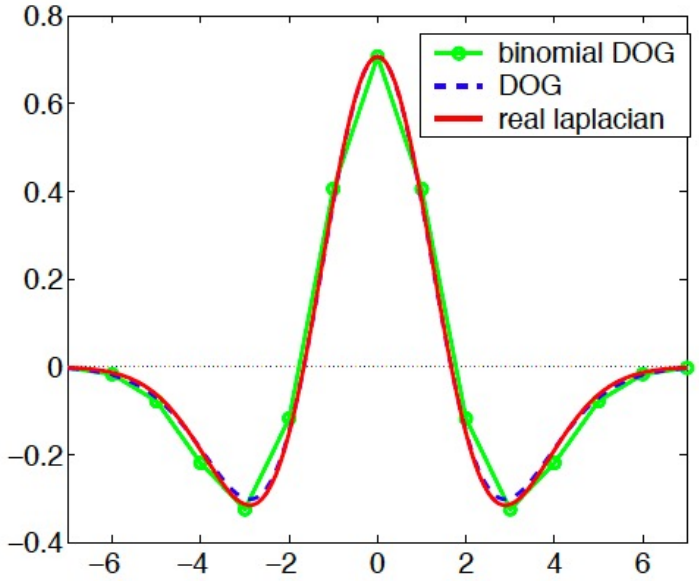
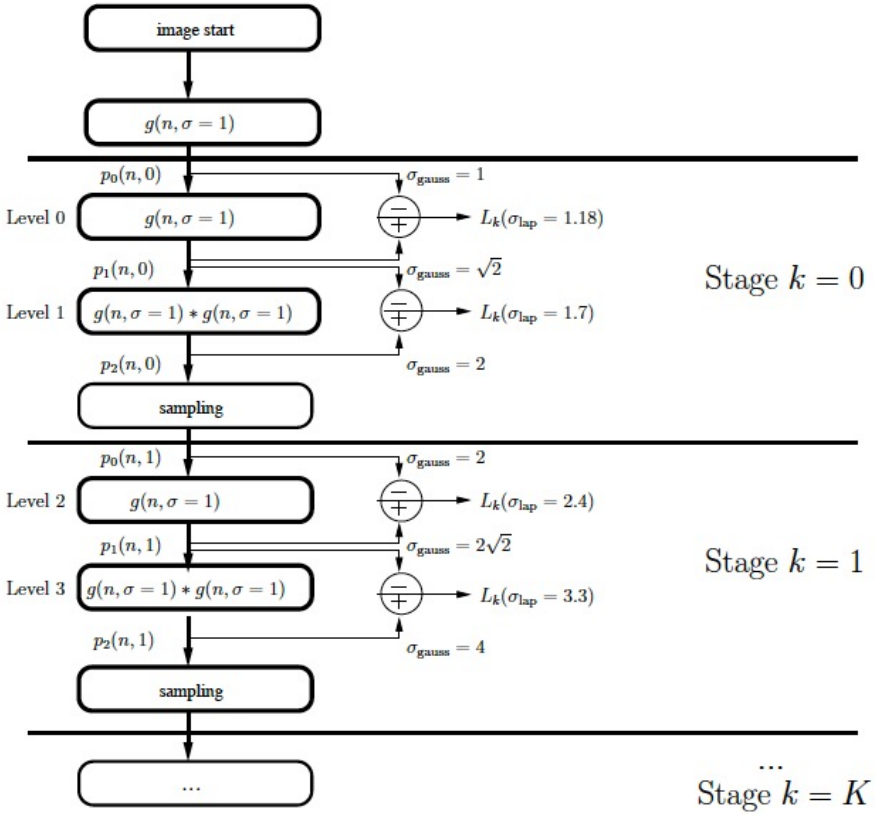


Figure 2: Comparisons of real Laplacian versus real DoG and binomial DoG for $\sigma_{\text{dog}} = \sqrt{2}$ and $\sigma_1 = \sqrt{2}\sigma_{\text{dog}} = 2$ and $\sigma_{\text{lap}} = 1.7$

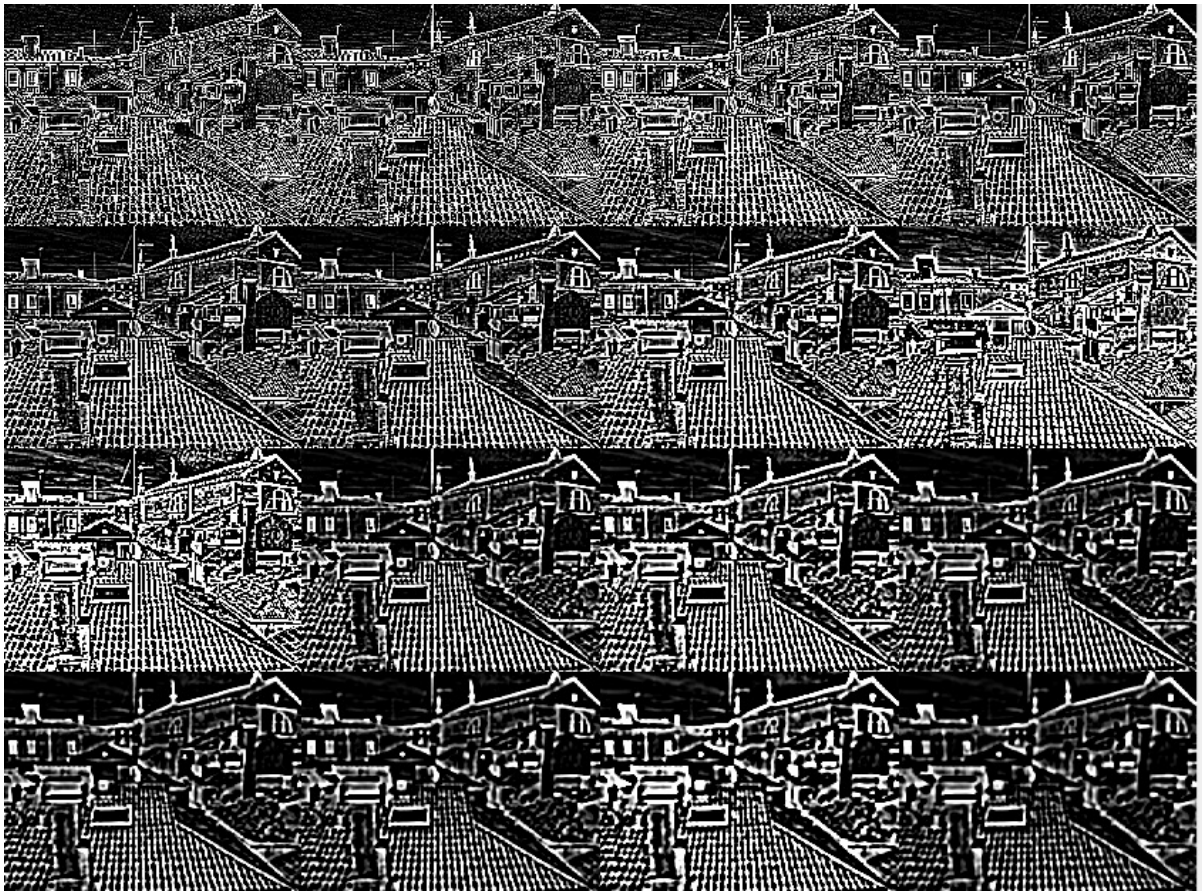
Effective σ of the Laplacian as Diff of Gaussians



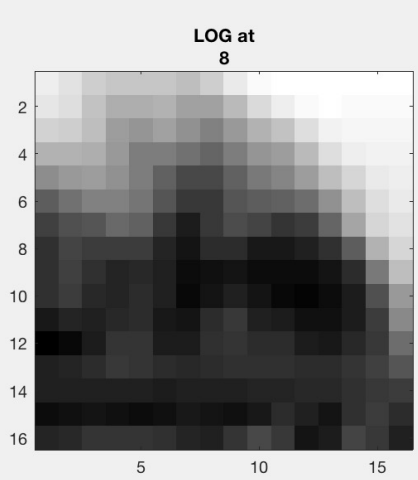
Reconstruction of the image from laplacian pyramid

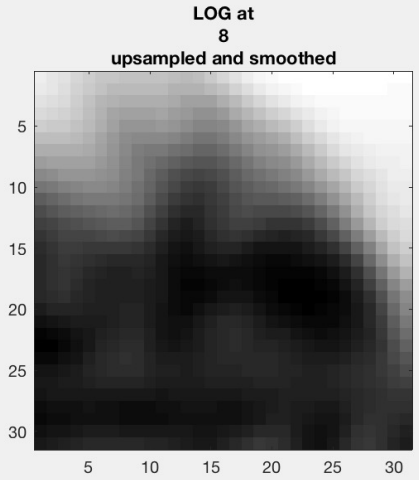
```
function [ img ] = pyrReconstruct( pyr )
%PYRRECONSTRUCT Uses a Laplacian pyramid to reconstruct a image
%   IMG = PYRRECONSTRUCT(PYR) PYR should be a 1*level cell array containing
%   the pyramid, SIZE(PYR{i}) = SIZE(PYR{i-1})*2-1
%       Yan Ke @ THUEE, xjed09@gmail.com

for p = length(pyr)-1:-1:1
    pyr{p} = pyr{p}+pyr_expand(pyr{p+1});  %% upsampled and smoothed at level (p+1)
end
img = pyr{1};

end
```

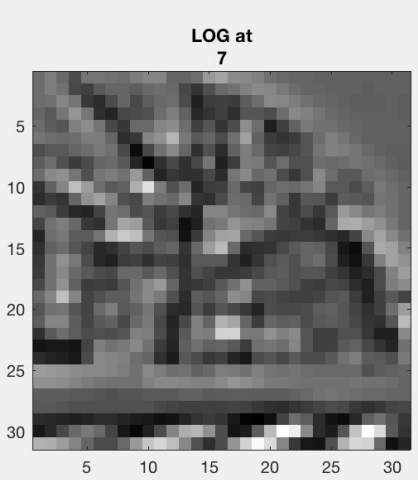
Laplacian Scale Space

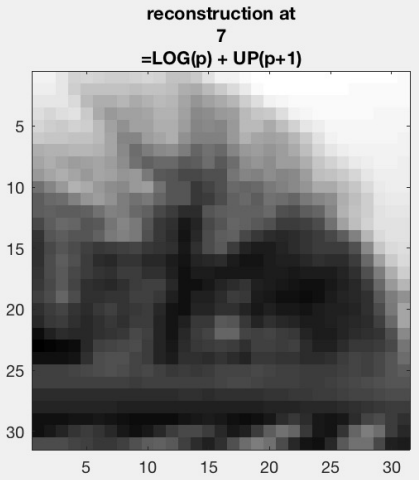


pyr{p} = pyr{p}+pyr_expand(pyr{p+1});

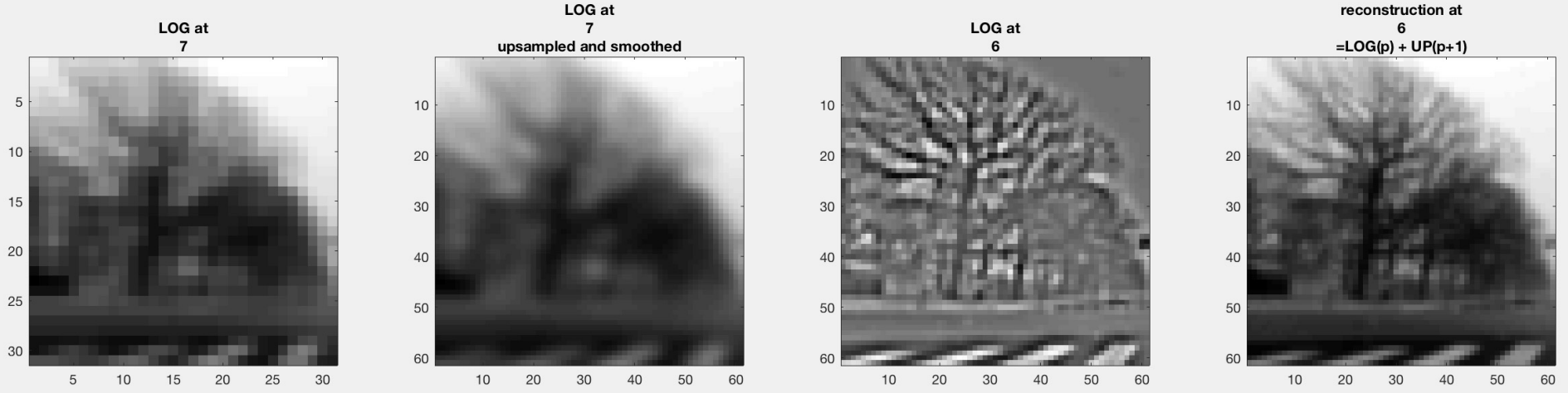




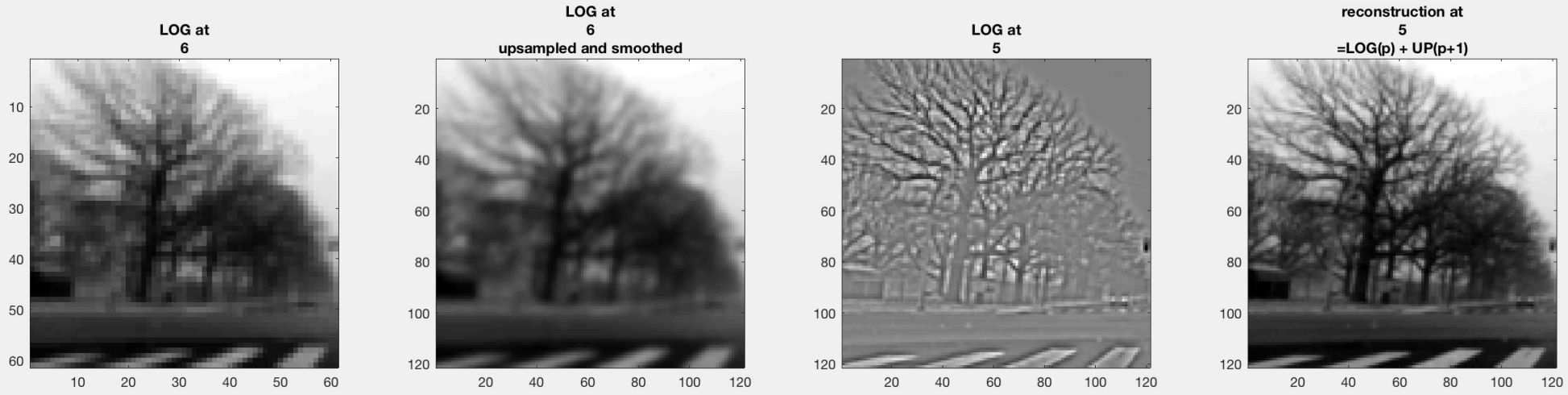




pyr{p} = pyr{p}+pyr_expand(pyr{p+1});



pyr{p} = pyr{p}+pyr_expand(pyr{p+1});



pyr{p} = pyr{p}+pyr_expand(pyr{p+1});