

Pyramids and Blending



Goal: Image Compositing



Compositing Procedure

1. Extract Sprites (e.g using *Intelligent Scissors* in Photoshop)

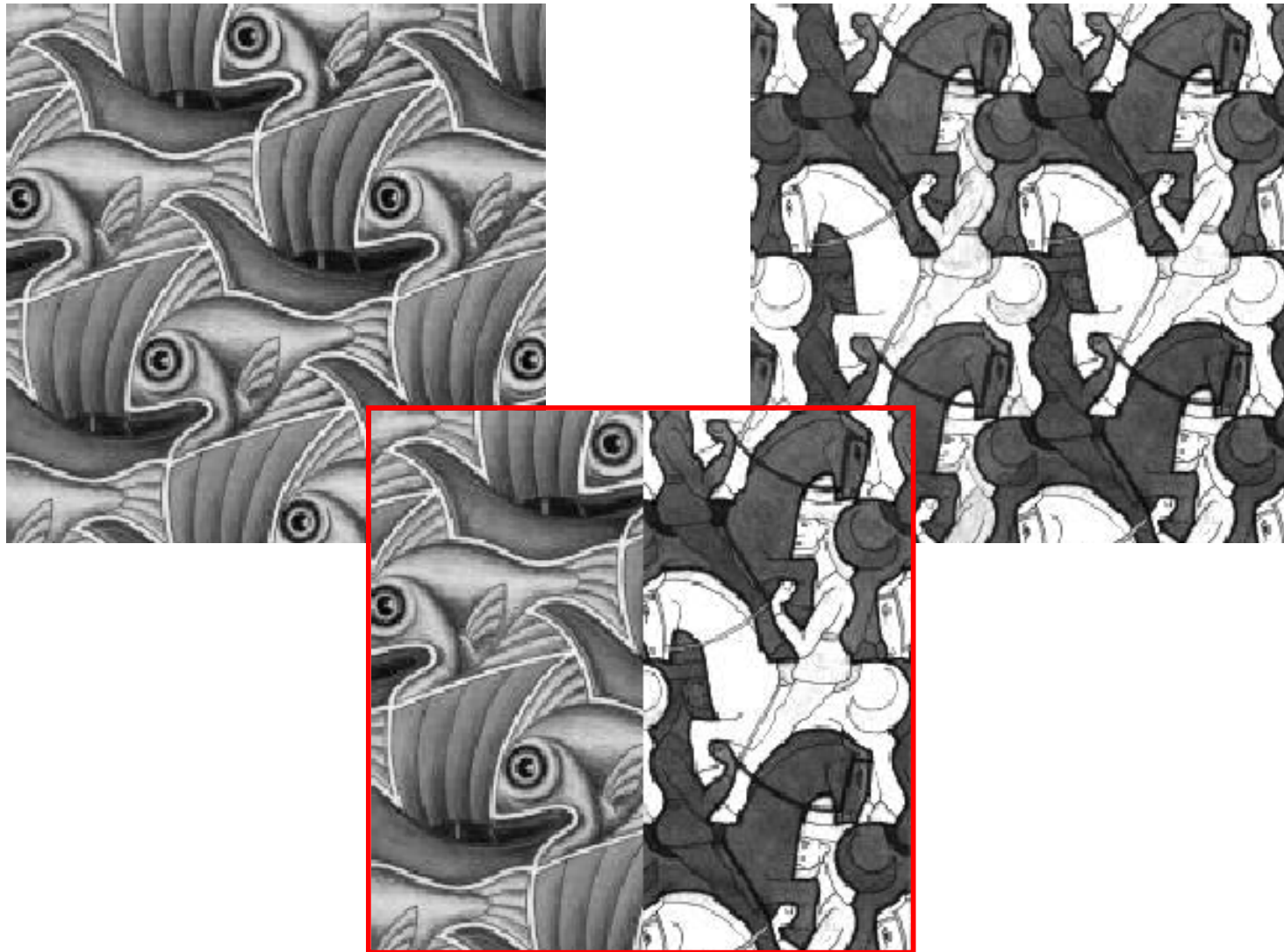


2. Blend them into the composite (in the right order)

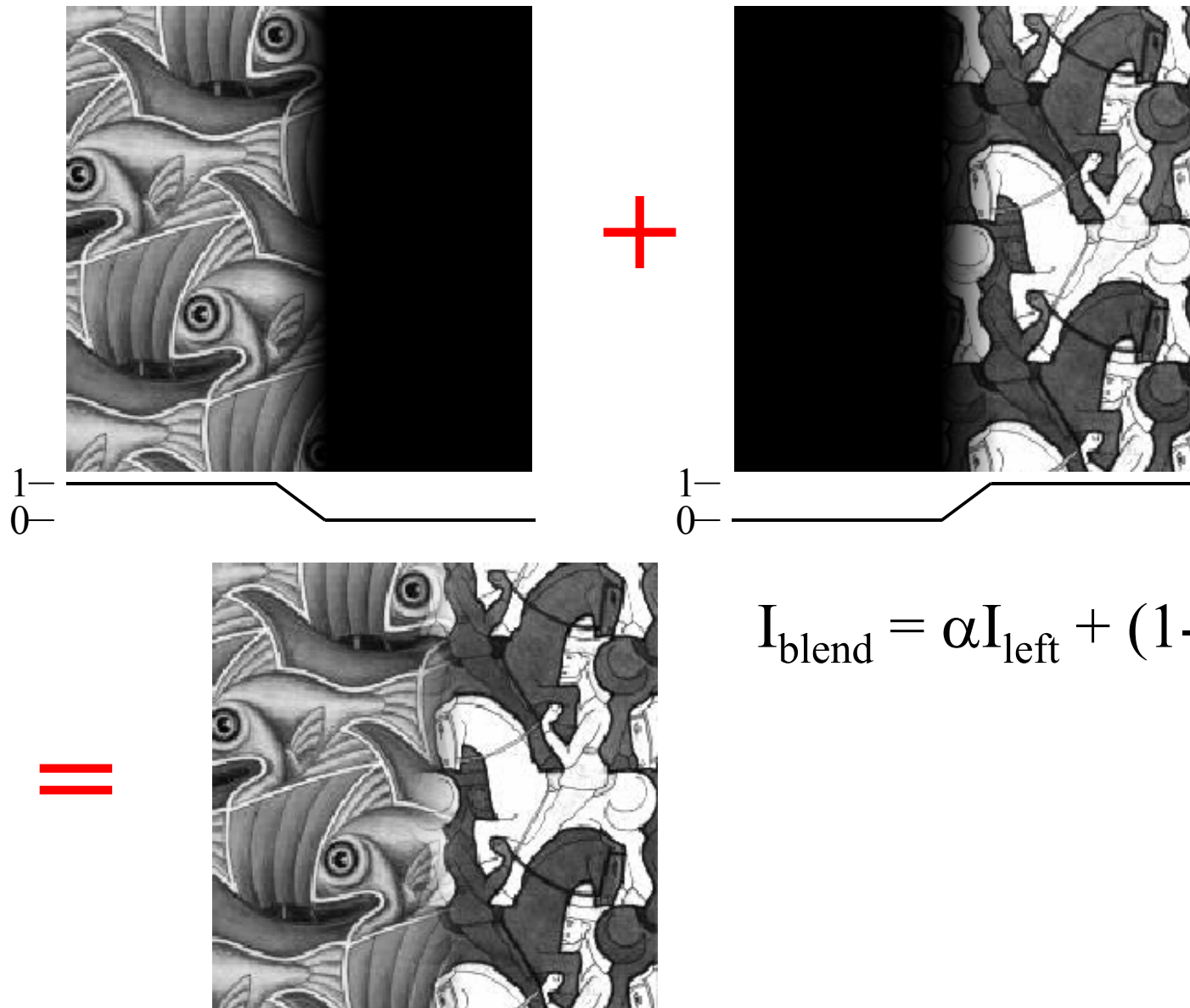


Composite by
David Dewey

Need blending to avoid seams

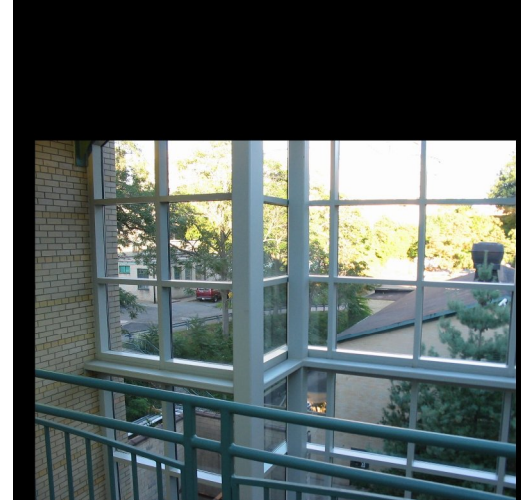


Alpha Blending / Feathering

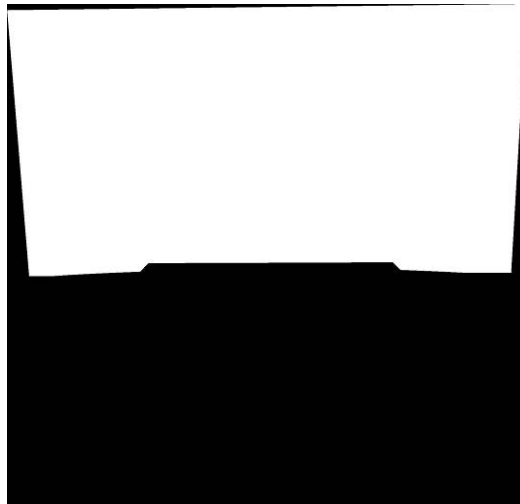


$$I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha) I_{\text{right}}$$

Setting alpha: center seam

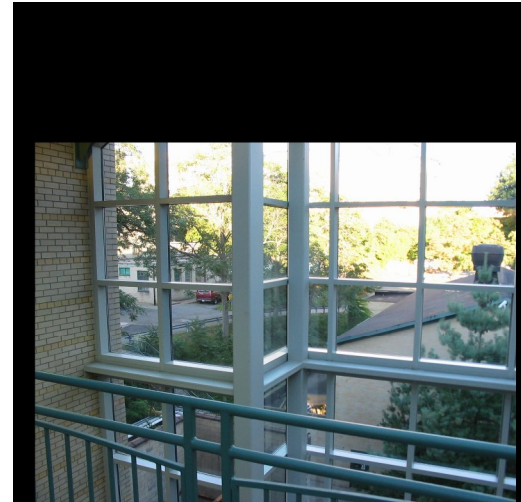


Distance
Transform
bwdist



$$\text{Alpha} = \text{logical}(\text{dtrans1} > \text{dtrans2})$$

Setting alpha: blurred seam

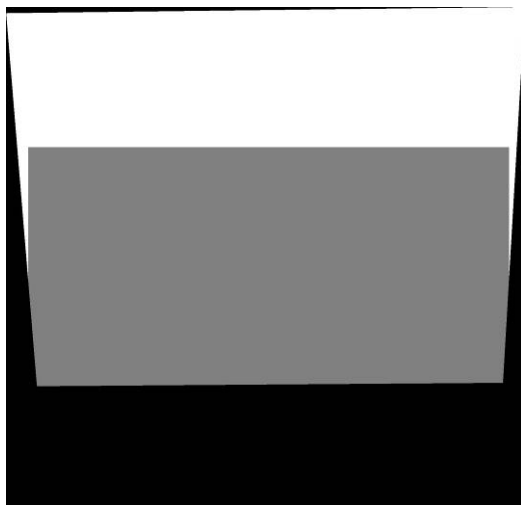
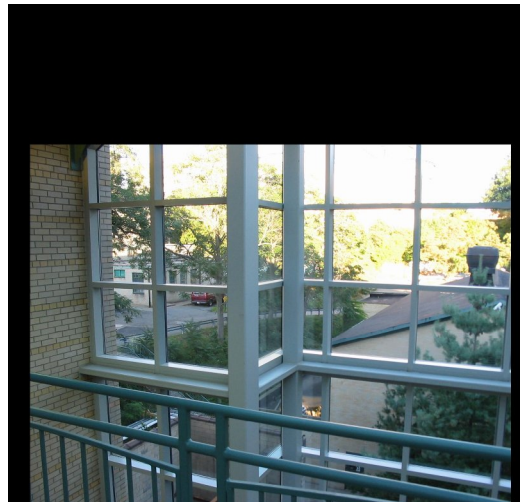


Distance
transform



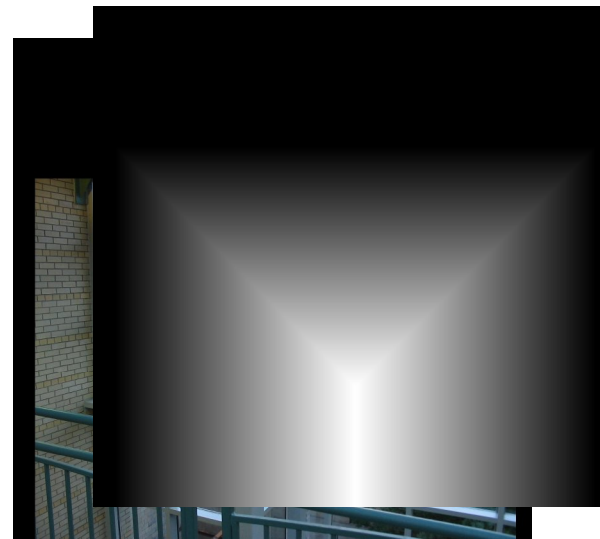
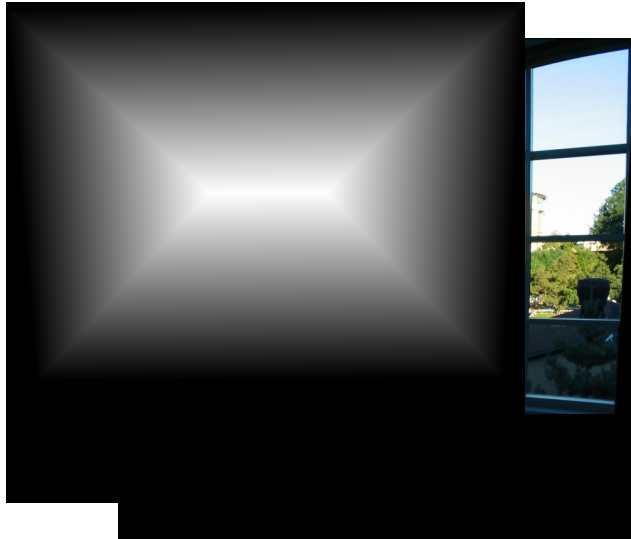
Alpha = blurred

Setting alpha: simple averaging

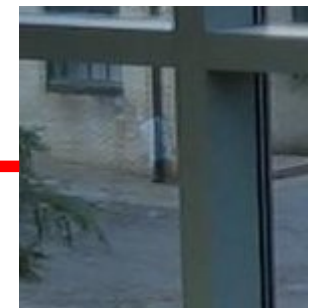
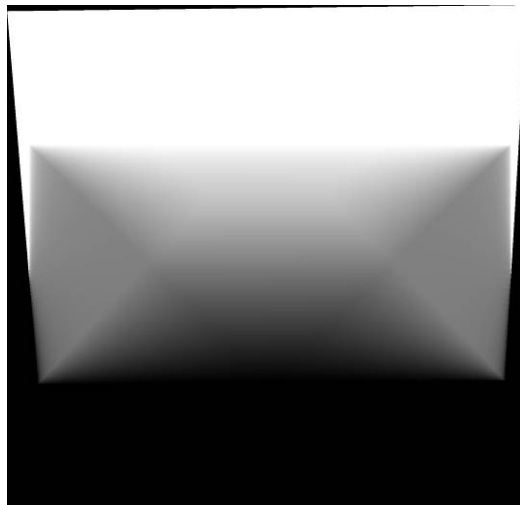


Alpha = .5 in overlap region

Setting alpha: center weighting



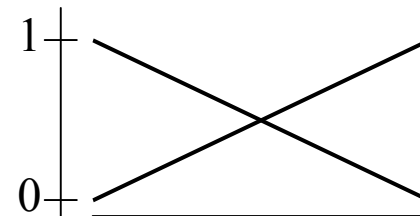
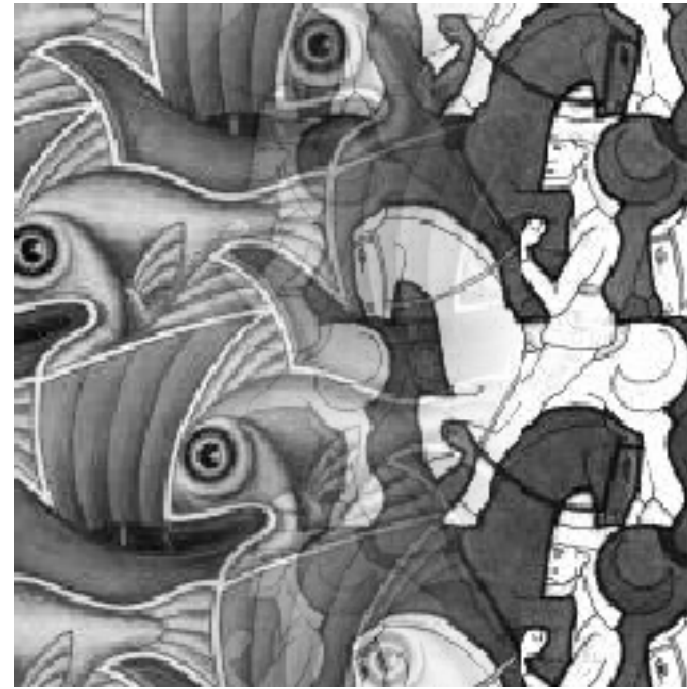
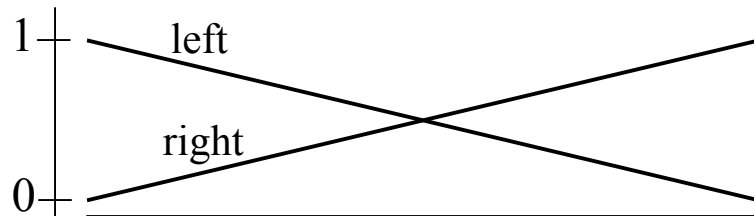
Distance
transform



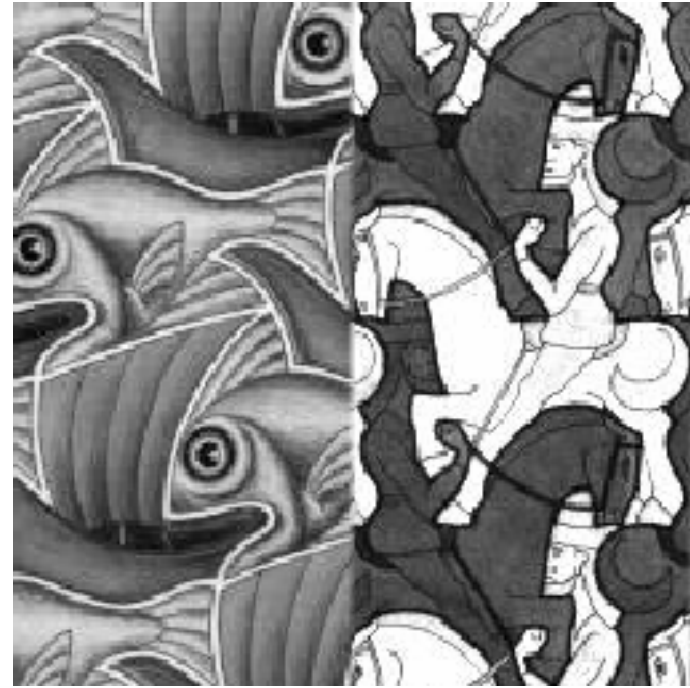
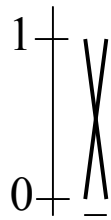
Ghost!

$$\text{Alpha} = \text{dtrans1} / (\text{dtrans1} + \text{dtrans2})$$

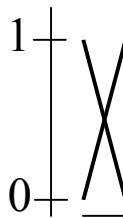
Affect of Window Size



Affect of Window Size



Good Window Size



“Optimal” Window: smooth but not ghosted

- Key Insight:

Coarse structure should blend very slowly between images (lots of feathering), while fine details should transition more quickly.

Natural to cast this in the Frequency Domain.

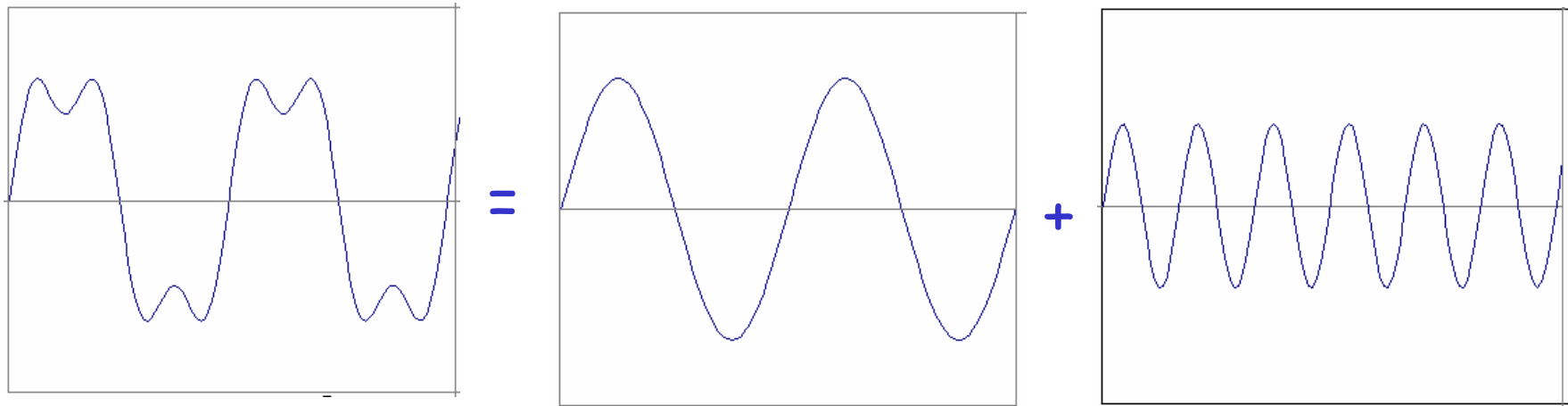
Fourier Domain: A sum of sines

Any periodic function can be rewritten as a weighted sum of sines.

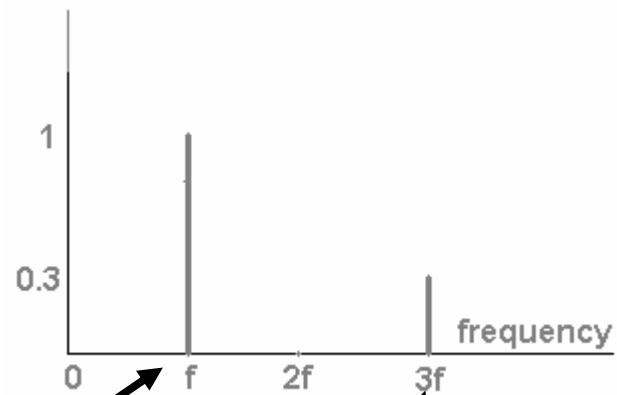
Basic building block: $A \sin(\omega x + \phi)$

Add as many as you want to get any function you want!

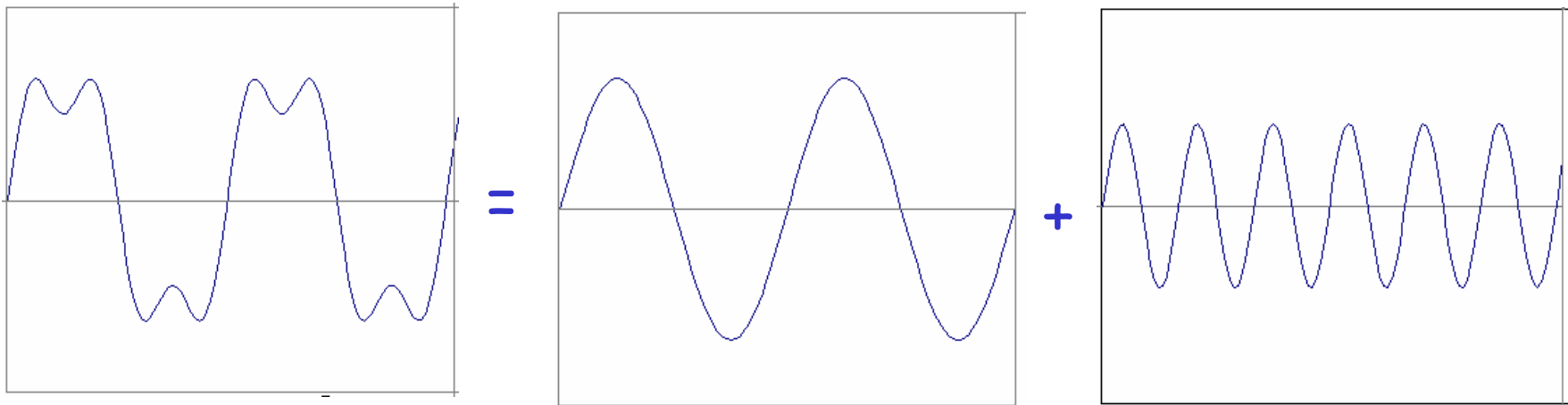
example : $g(t) = \sin(2pf t) + (1/3)\sin(2p(3f) t)$



Fourier Spectra

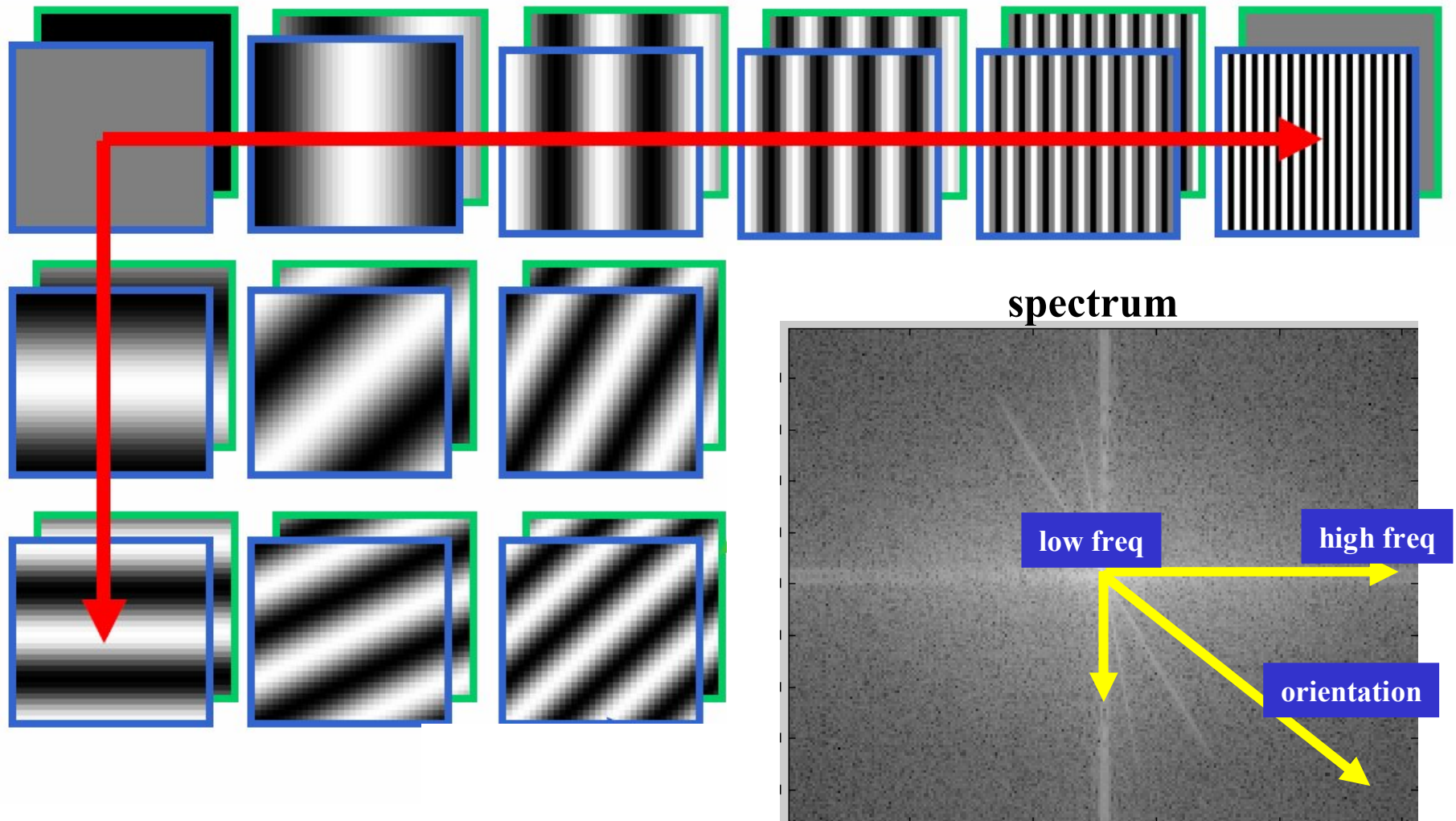


example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t)$

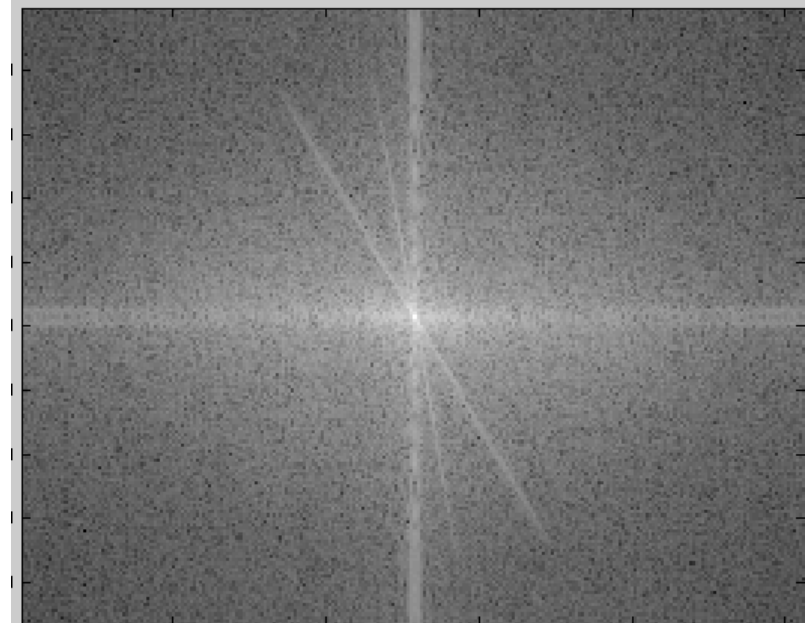


Extension to 2D

provides a basis for decomposing any 2D image into a linear combination of oriented frequencies.



Example: Man-made Scene



amplitude of fourier spectra

in Matlab, check out: `imagesc(log(abs(fftshift(fft2(im)))));`

Manipulating Frequencies

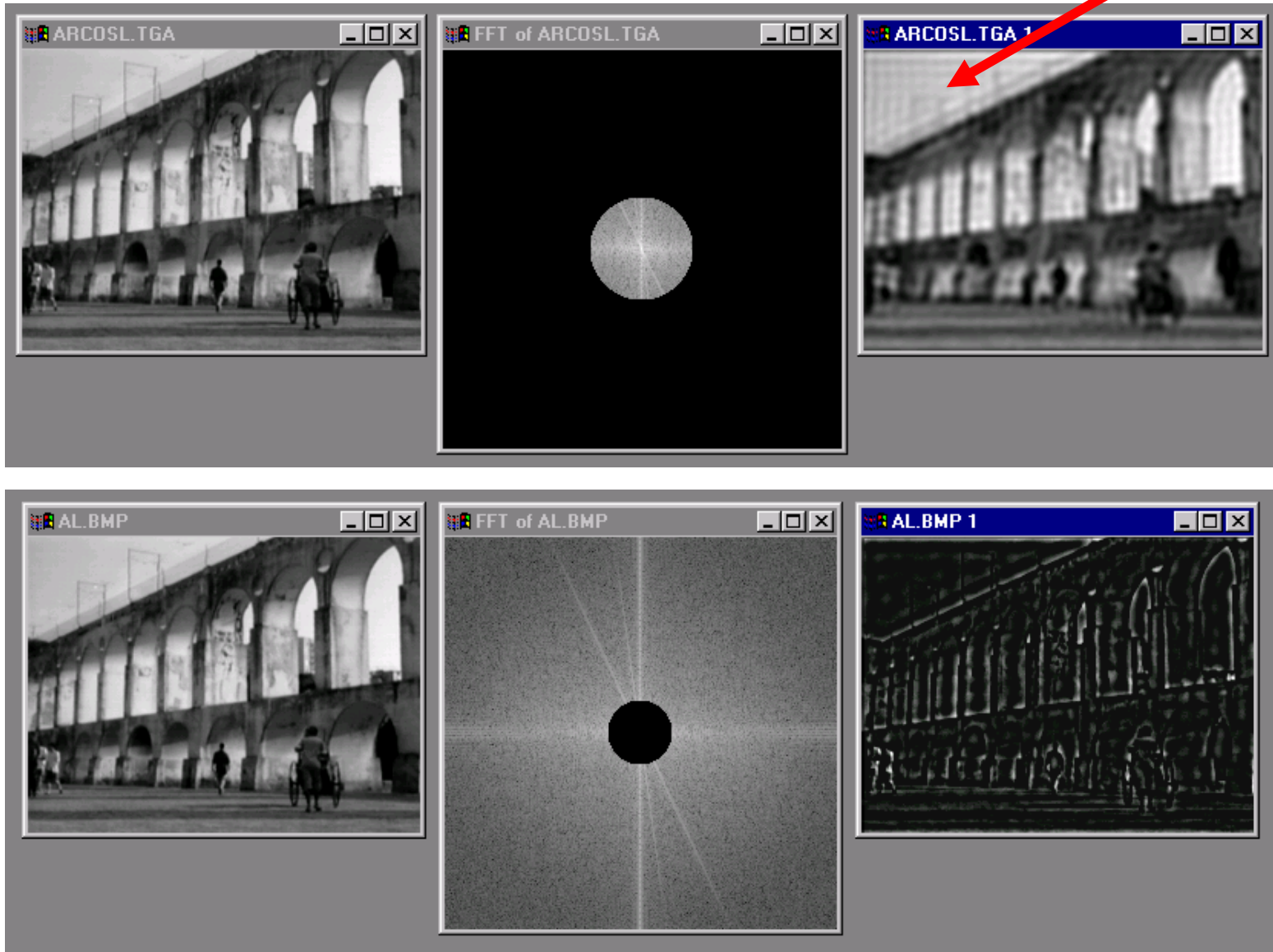
fourier transform and inverse fourier transform allow us to move back and forth from spatial to frequency domain.

=> can make a change in frequency domain and see what happens to image!



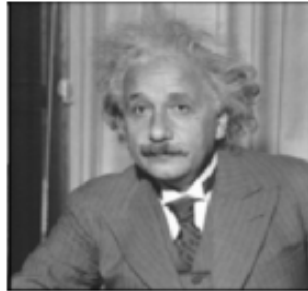
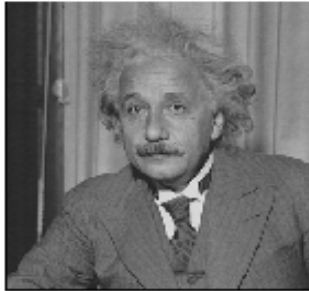
Low and High Pass filtering

note artifacts!

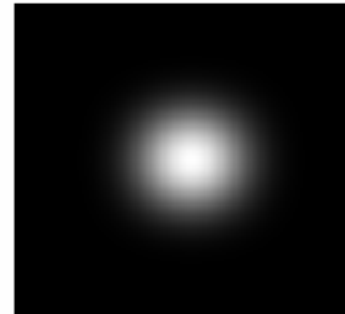


Smoother low/band-pass filtering

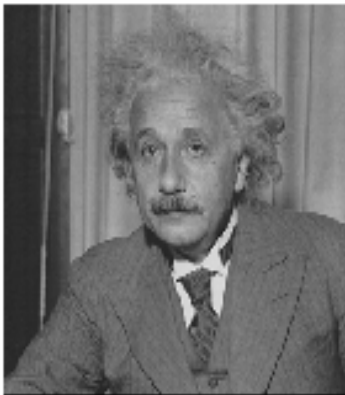
low-pass:



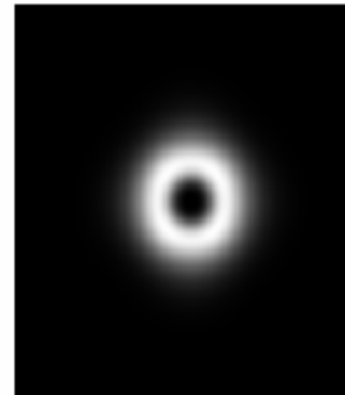
Gaussian



High-pass / band-pass:



LoG or DoG



2D convolution theorem example

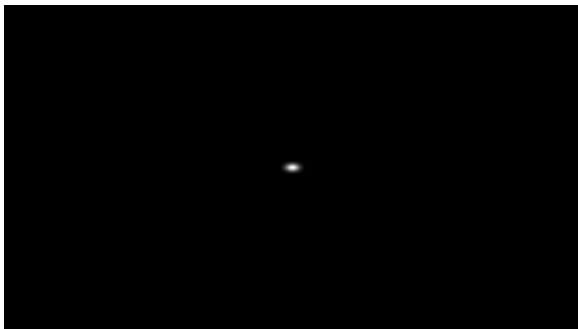
equivalence of operations in spatial and frequency domains

$f(x,y)$



$*$ convolution

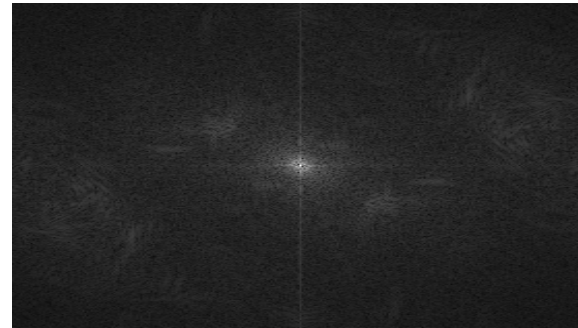
$h(x,y)$



$g(x,y)$

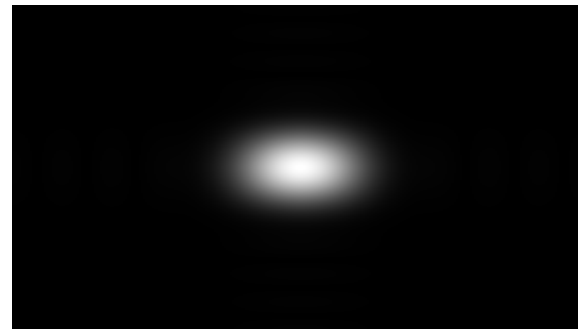


spatial domain

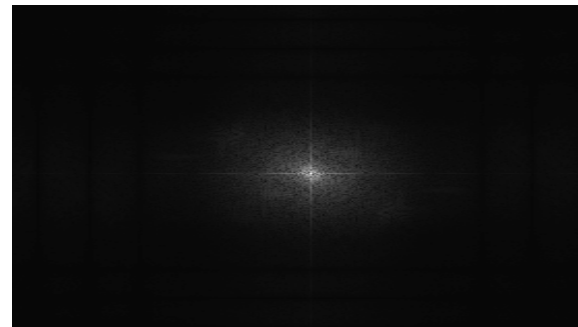


$|F(s_x, s_y)|$

$\times$ multiplication



$|H(s_x, s_y)|$



$|G(s_x, s_y)|$

frequency domain

Upshot of all this...

- Convolving with Gaussian in spatial domain is performing low-pass filtering in frequency domain
- Convolving with LoG/DoG filter in spatial domain is performing band-pass filtering in frequency domain

Recall: Gaussian Smoothing at Different Scales



original



$\sigma = 1$

Gaussian Smoothing at Different Scales



original



$\sigma = 3$

Gaussian Smoothing at Different Scales



original



$\sigma = 10$

Pyramid Representations

Because a large amount of smoothing limits the frequency of features in the image, we do not need to keep all the pixels around!

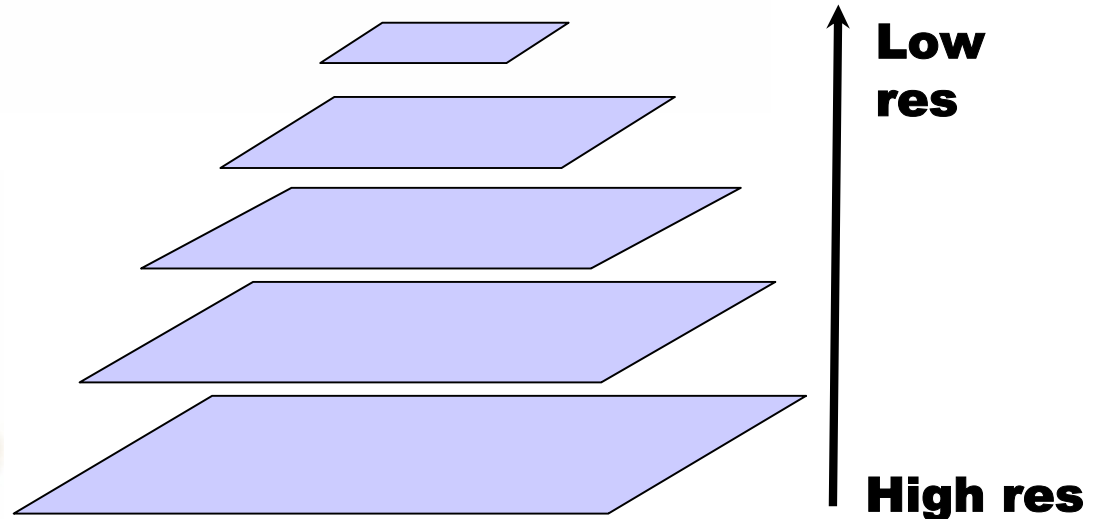
Example, if we smooth so much that we have uniform image, we only need to store 1 grey value!

Strategy: progressively reduce the number of pixels as we smooth more and more.

Leads to a “pyramid” representation if we subsample at each level.

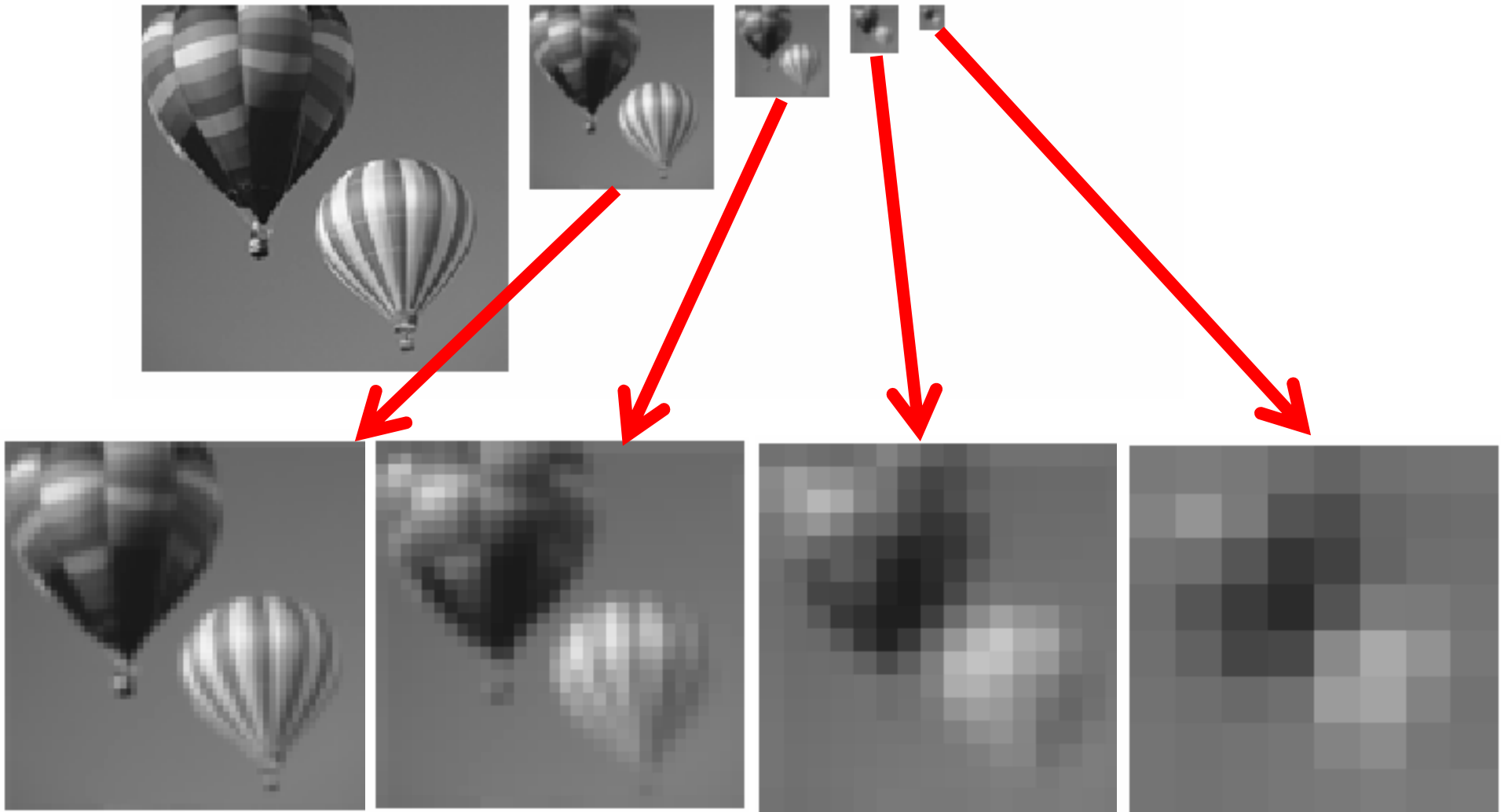
Gaussian Pyramid

High resolution  Low resolution



Upper Levels have Lower Resolution

High resolution  Low resolution



Aside: Binomial Approximation

The Binomial Approximation is useful for designing discrete Gaussian filters

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

n = number of elements in the 1D filter minus 1

r = position of element in the filter kernel (0, 1, 2...)

Note: generates the binomial coefficients
i.e. coefficients of $(x+y)^n$

Aside: Binomial Approximation

Easy to use if you
remember 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...)

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

Aside: Binomial Approximation

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

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

$[1 \ 2 \ 1]/4$ - approximates
Gaussian with $\sigma=1/\sqrt{2}$

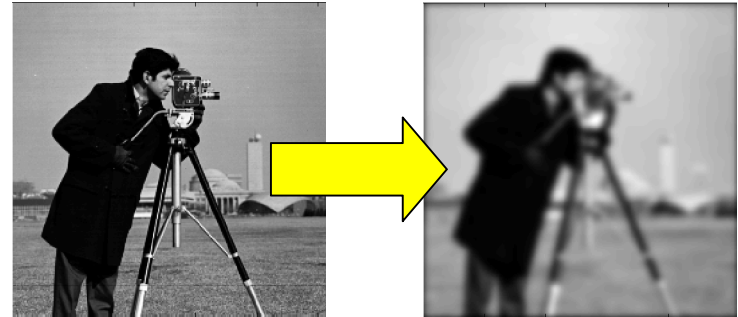
$[1 \ 4 \ 6 \ 4 \ 1]/16$ - approximates
Gaussian with $\sigma=1$

Cascaded filters: $[1 \ 2 \ 1]/4 * [1 \ 2 \ 1]/4 \longrightarrow [1 \ 4 \ 6 \ 4 \ 1]/16$

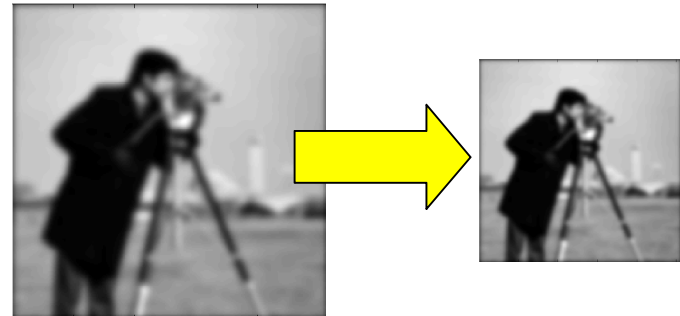
Generating a Gaussian Pyramid

Basic Functions:

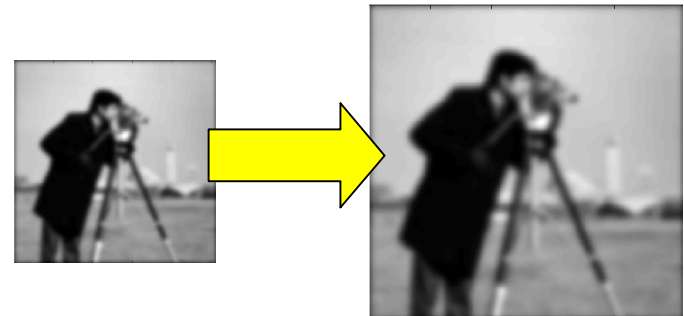
Blur (convolve with Gaussian to smooth image)



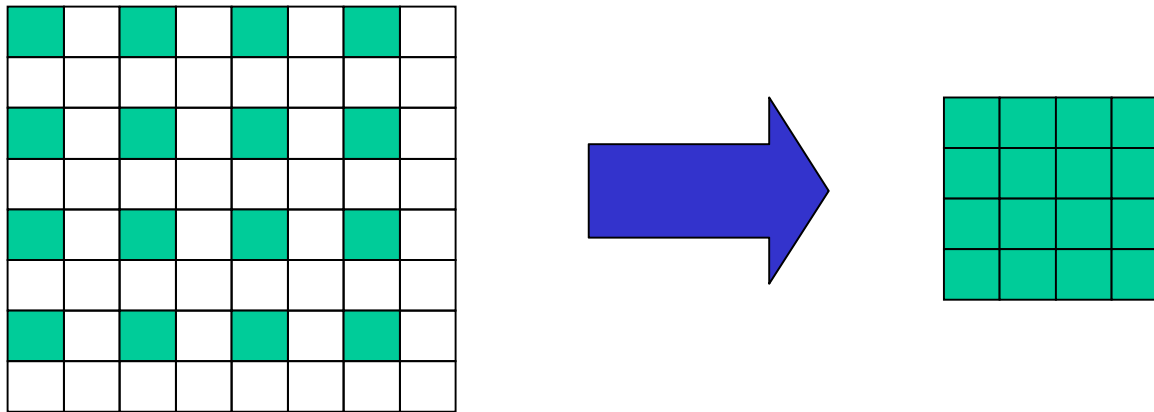
DownSample (reduce image size by half)



Upsample (double image size)

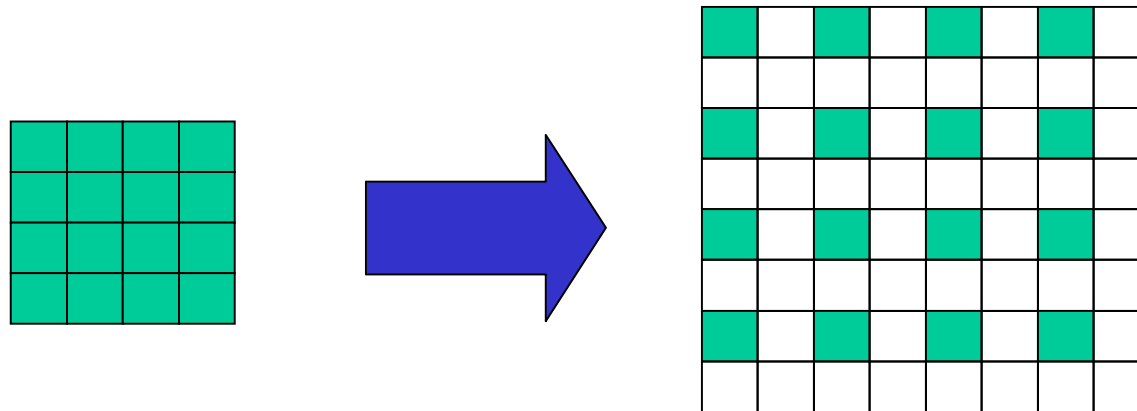


Downsample



Recall: this is a bad idea unless you have previously blurred (smoothed) the image!

Upsample

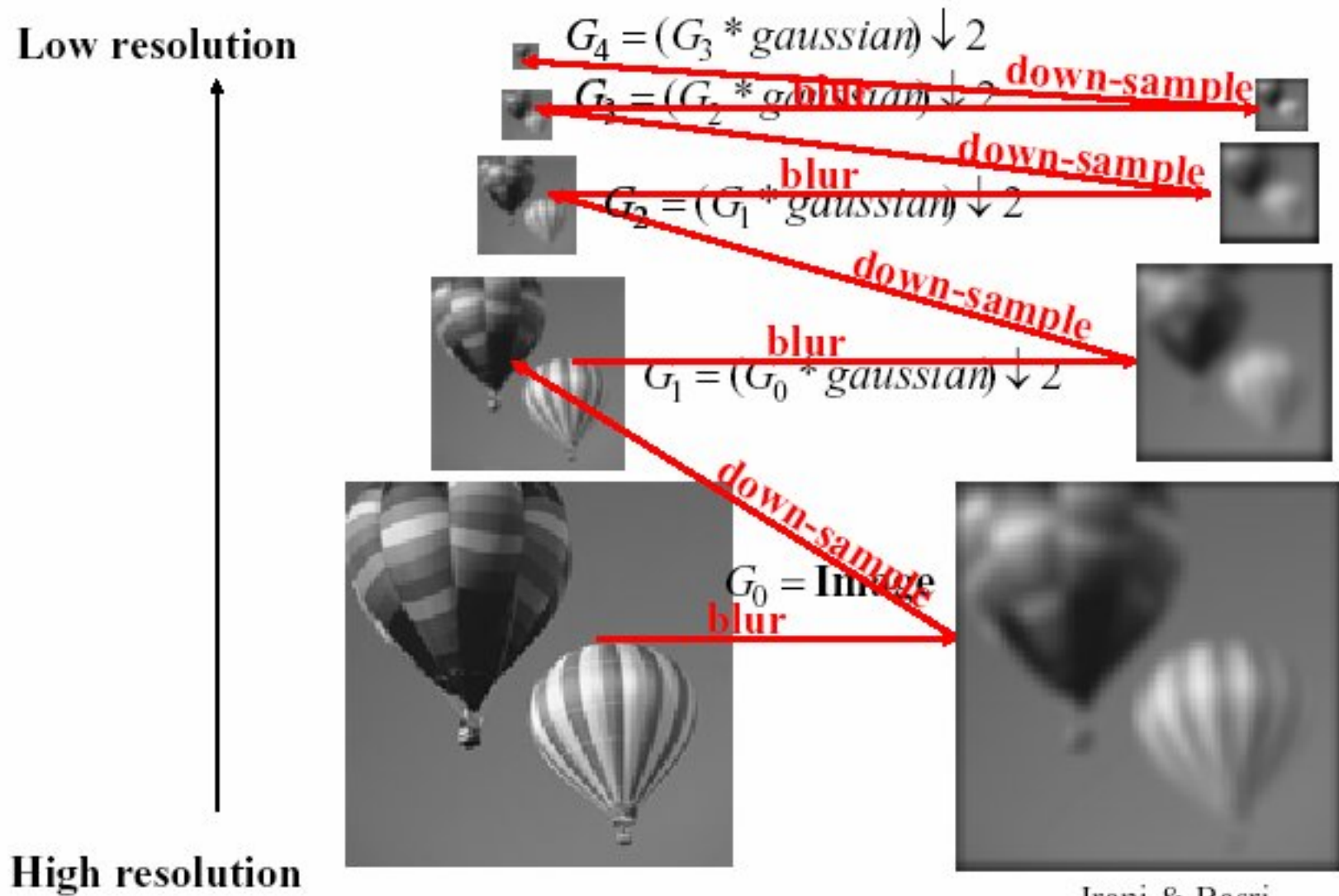


How to fill in the empty values?

Interpolation:

- initially set empty pixels to zero
- convolve upsampled image with Gaussian filter!
e.g. 5x5 kernel with $\sigma = 1$.
- **Must also multiply by 4. Explain why.**

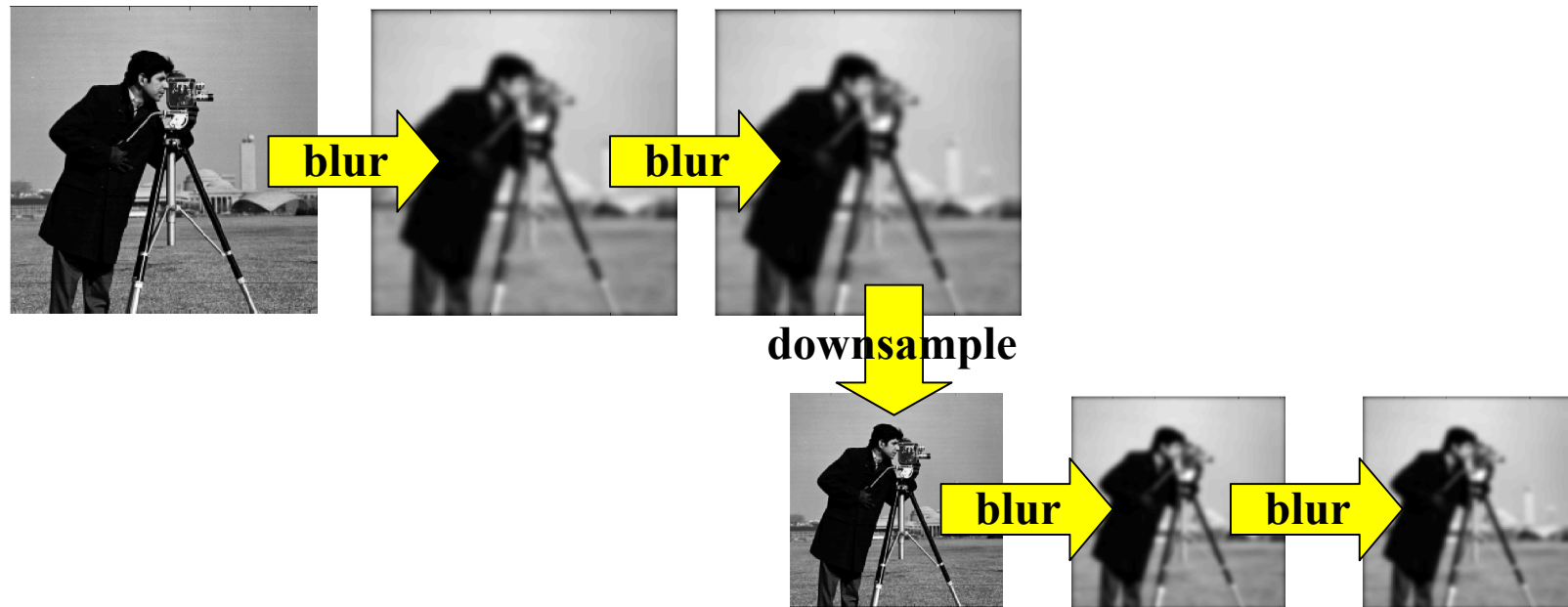
Forming the Gaussian Pyramid



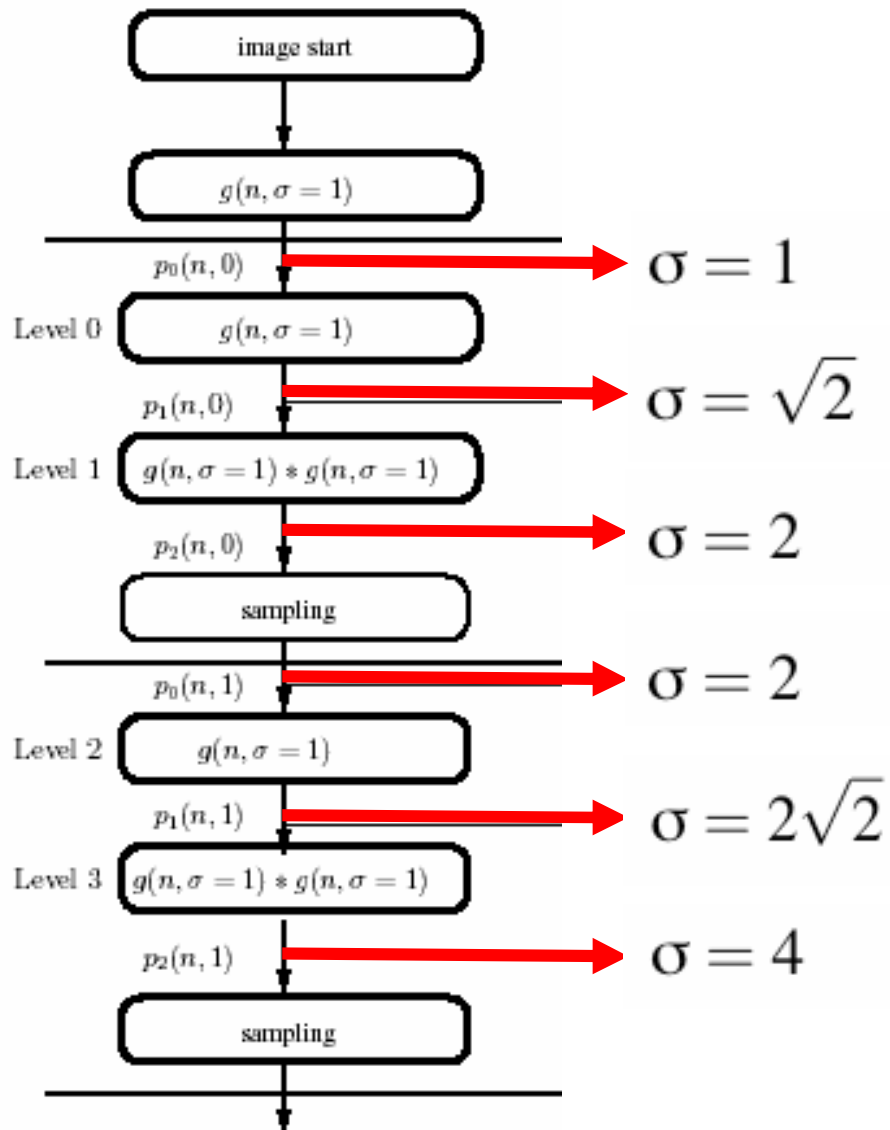
Specific Example

From Crowley et.al., “Fast Computation of Characteristic Scale using a Half-Octave Pyramid.”

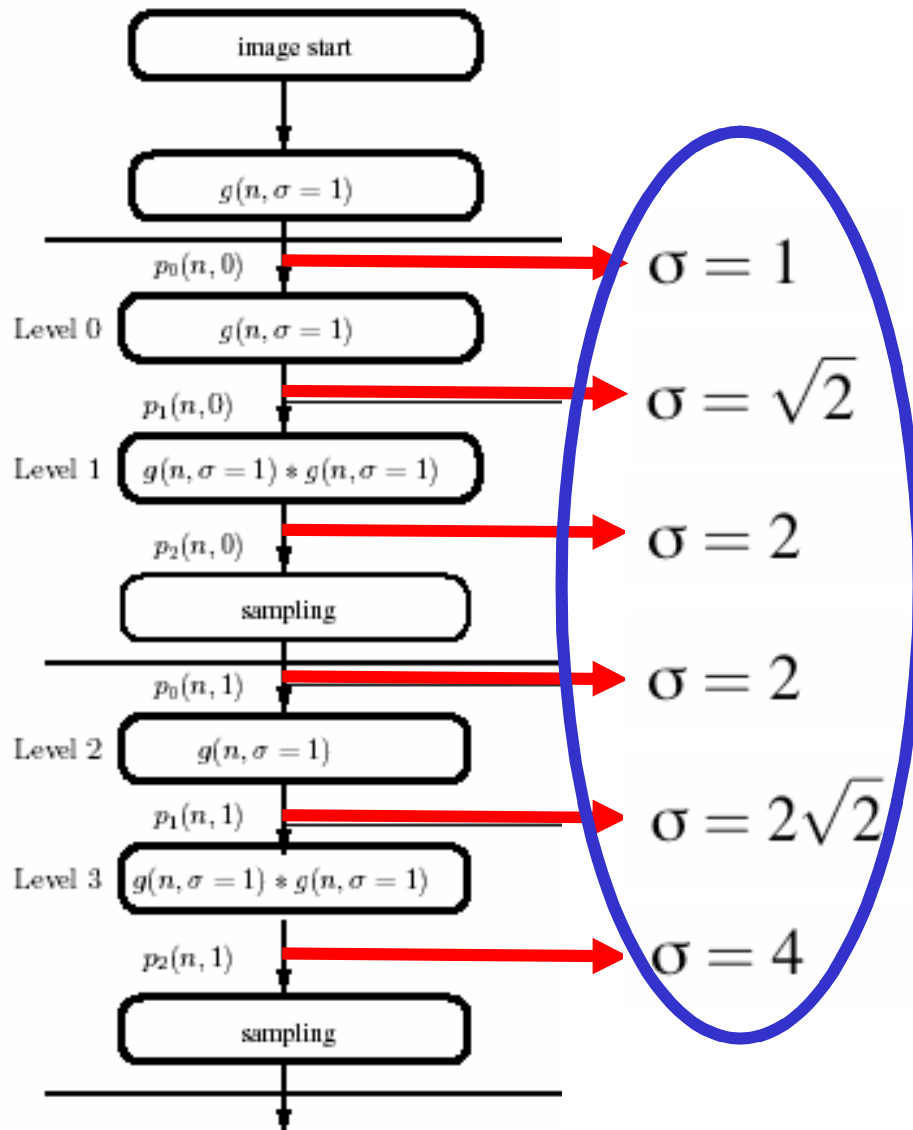
General idea: cascaded filtering using $[1 \ 4 \ 6 \ 4 \ 1]$ kernel to generate a pyramid with two images per octave (power of 2 change in resolution). When we reach a full octave, downsample the image.



Effective Sigma at Each Level

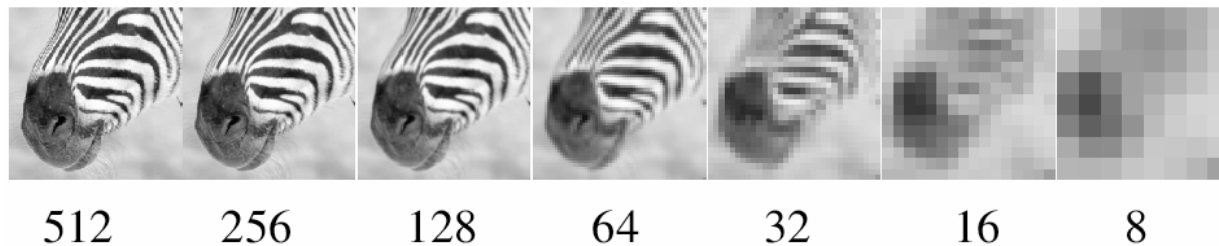


Effective Sigma at Each Level

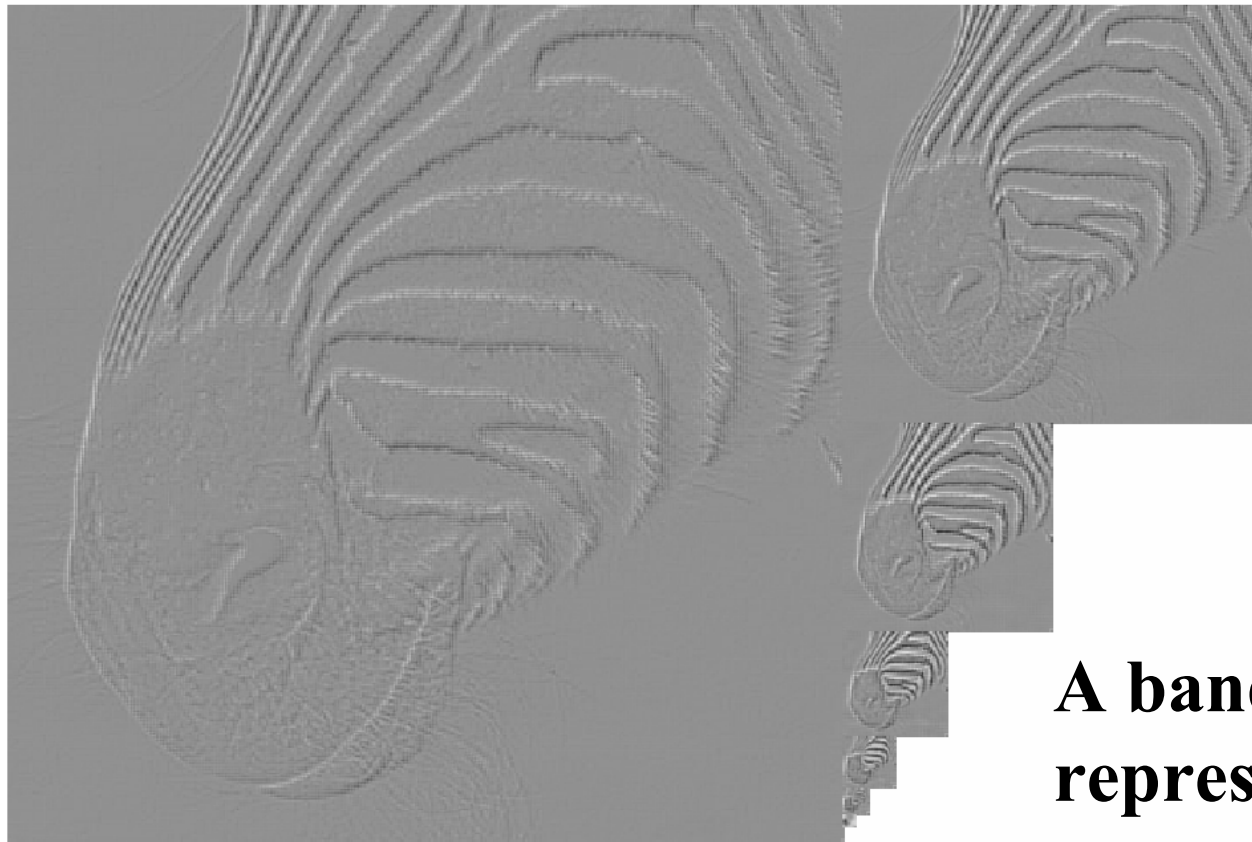
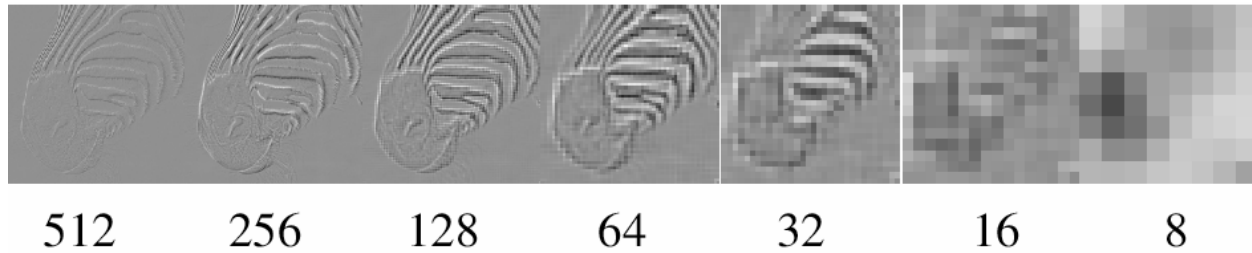


**CAN YOU
EXPLAIN HOW
THESE VALUES
ARISE?**

Gaussian pyramid

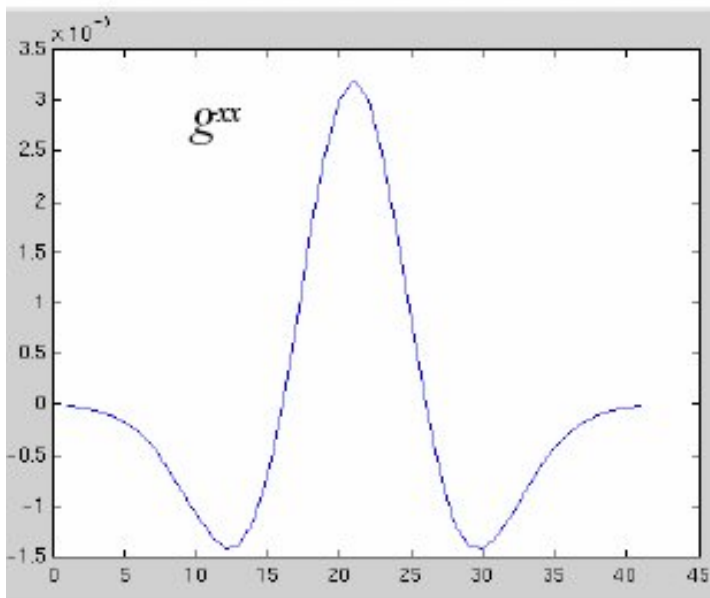
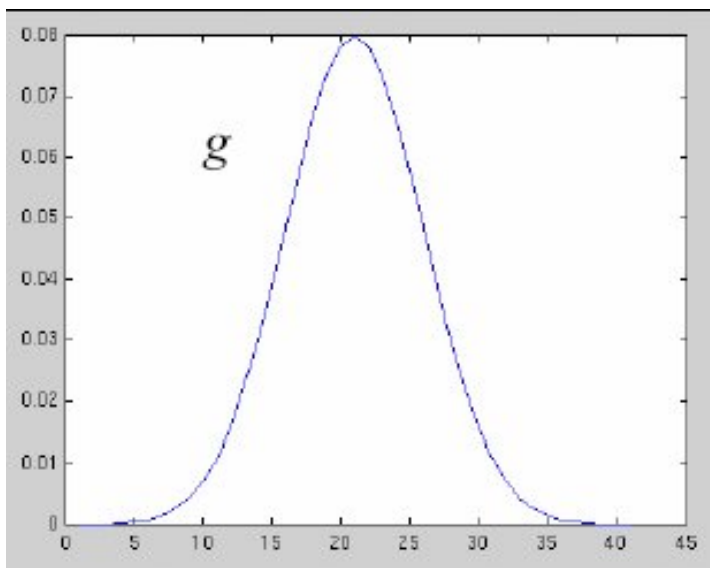


From Gaussian to Laplacian Pyramid

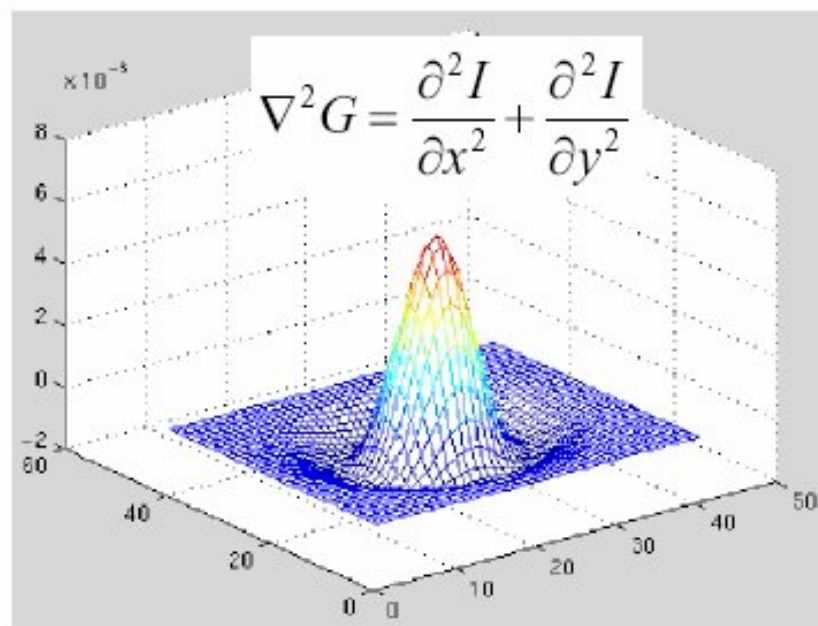


**A band-pass
representation**

LoG Operator



Second derivatives:
Laplacian

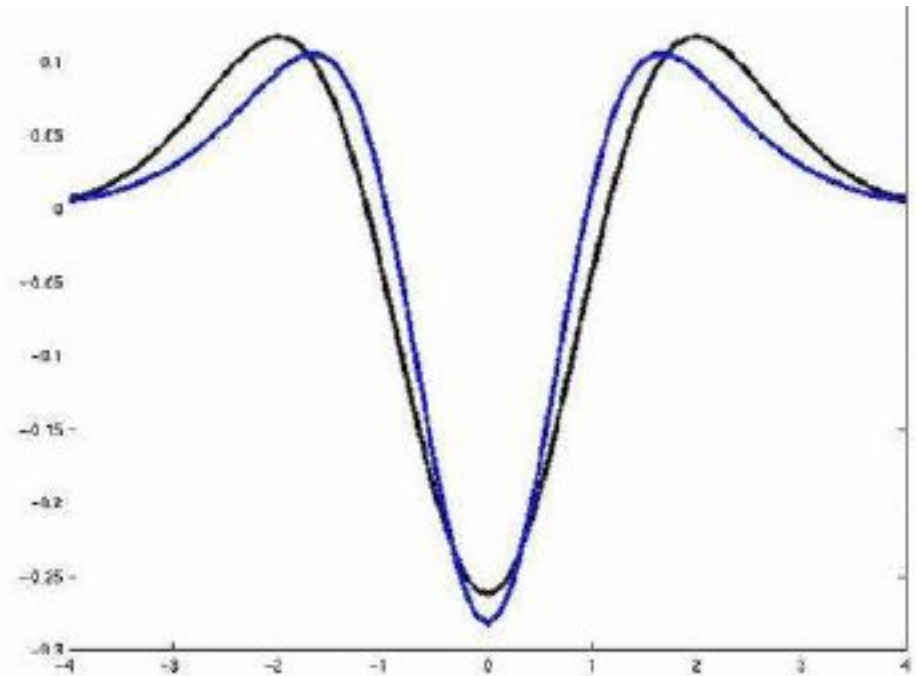
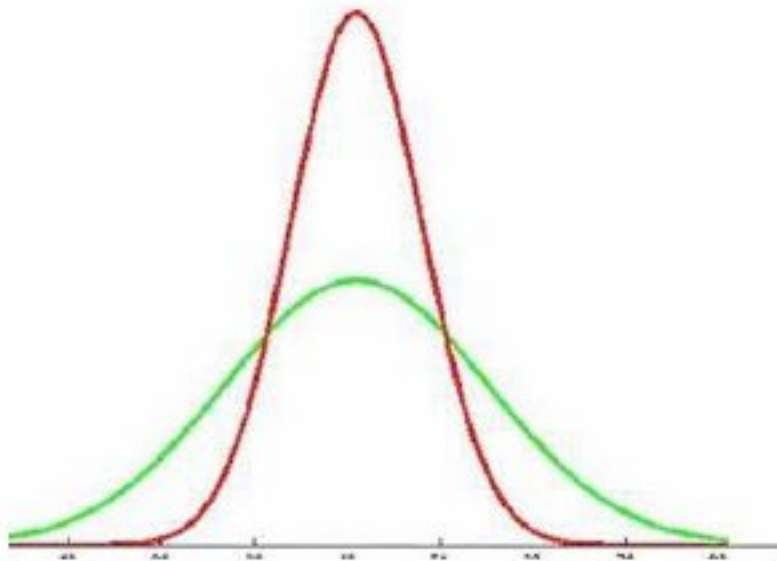


Approximating LoG with DoG

LoG can be approximate by a Difference of two Gaussians (DoG) at different scales

$$\nabla^2 G_\sigma \approx G_{\sigma_1} - G_{\sigma_2}$$

Best approximation when:
 $\sigma_1 = \frac{\sigma}{\sqrt{2}}, \sigma_2 = \sqrt{2}\sigma$



Laplacian Pyramids

- Given input I
- Construct Gaussian pyramid $I^G_1, \dots, I^G_n$
- Take the difference between consecutive levels:
- $I^L_k = I^L_k - I^L_{k-1}$
- Image I^L_k is an approximation of the Laplacian at scale number k

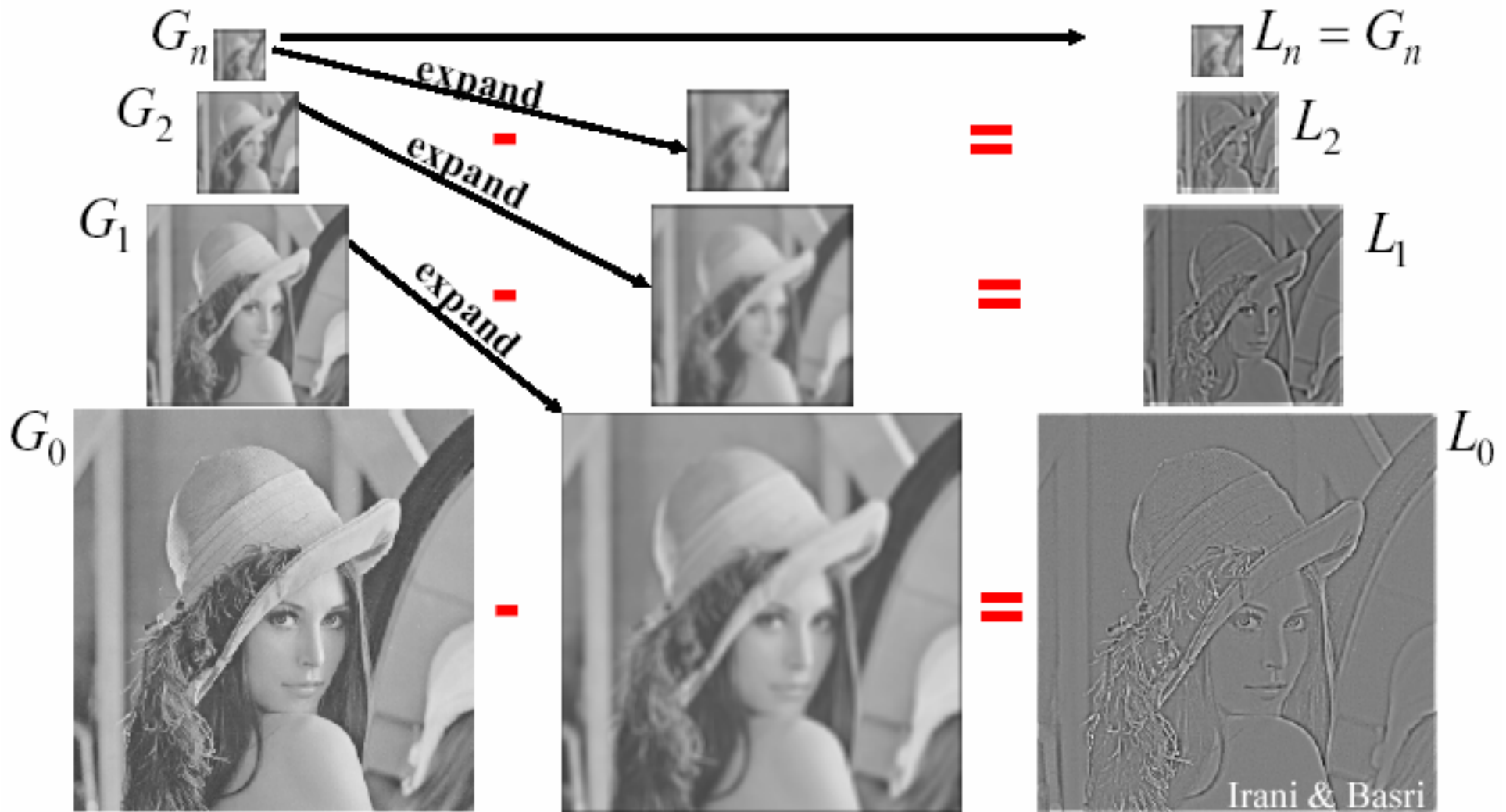
Generating the Laplacian Pyramid

$$L_i = G_i - \text{expand}(G_{i+1})$$

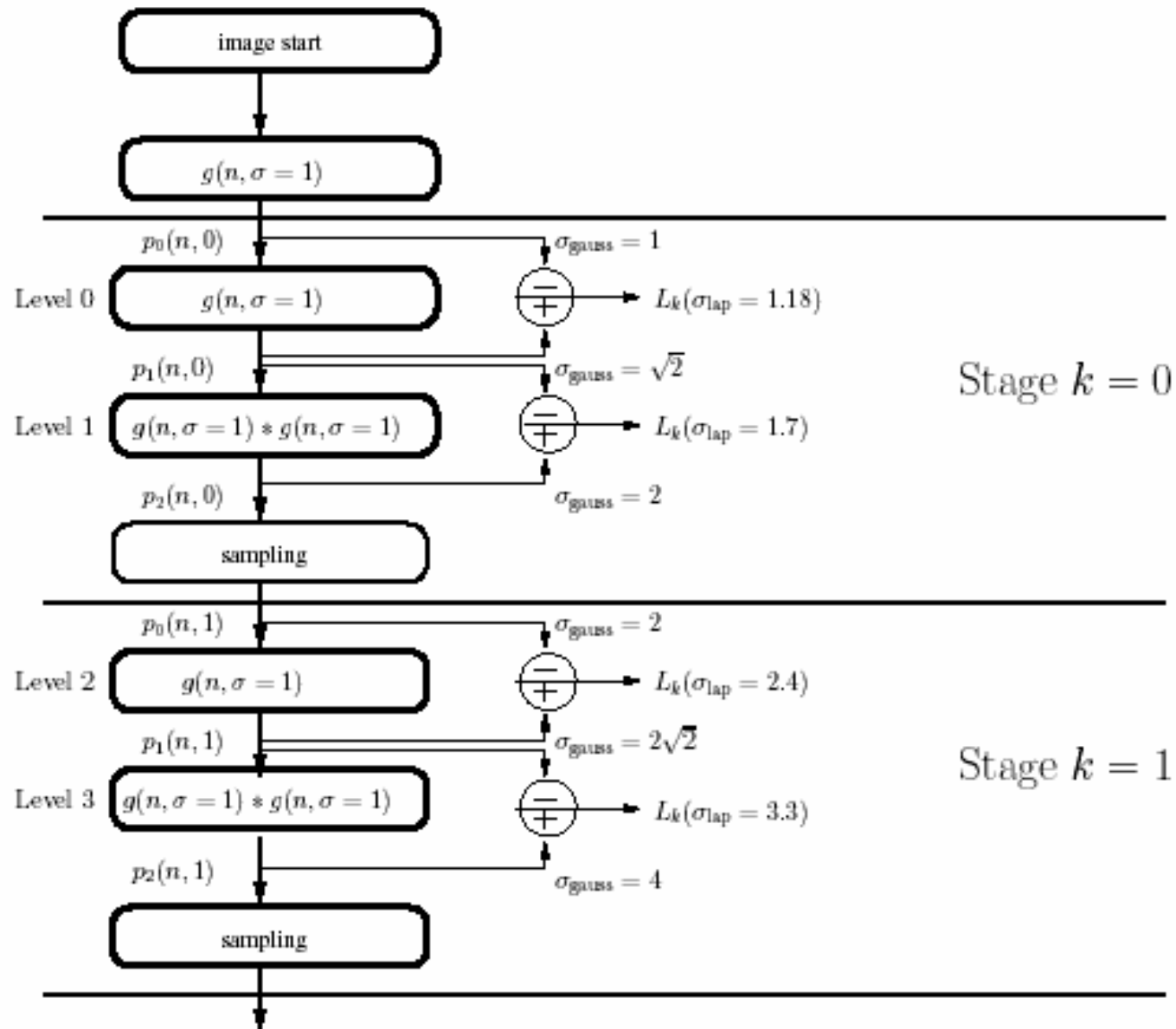
Gaussian Pyramid

$$G_i = L_i + \text{expand}(G_{i+1})$$

Laplacian Pyramid

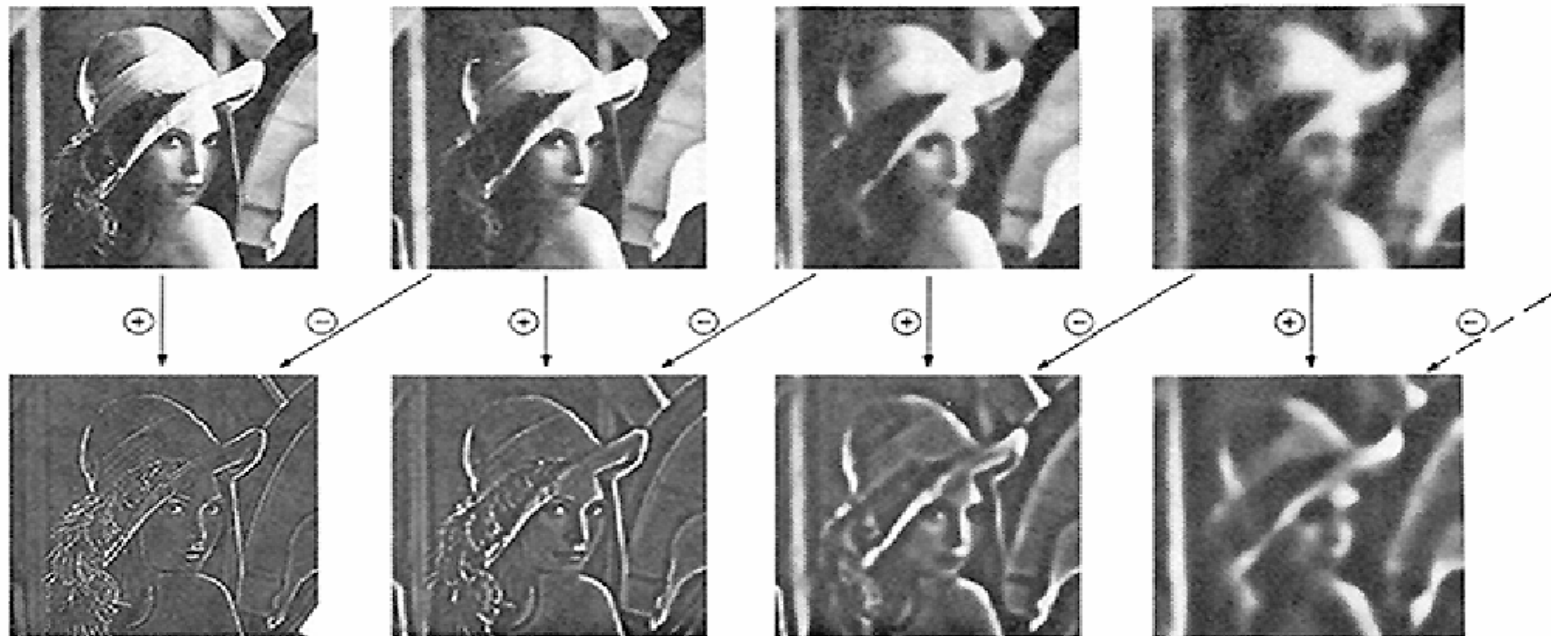


Or Using the Half-Octave Pyramid...

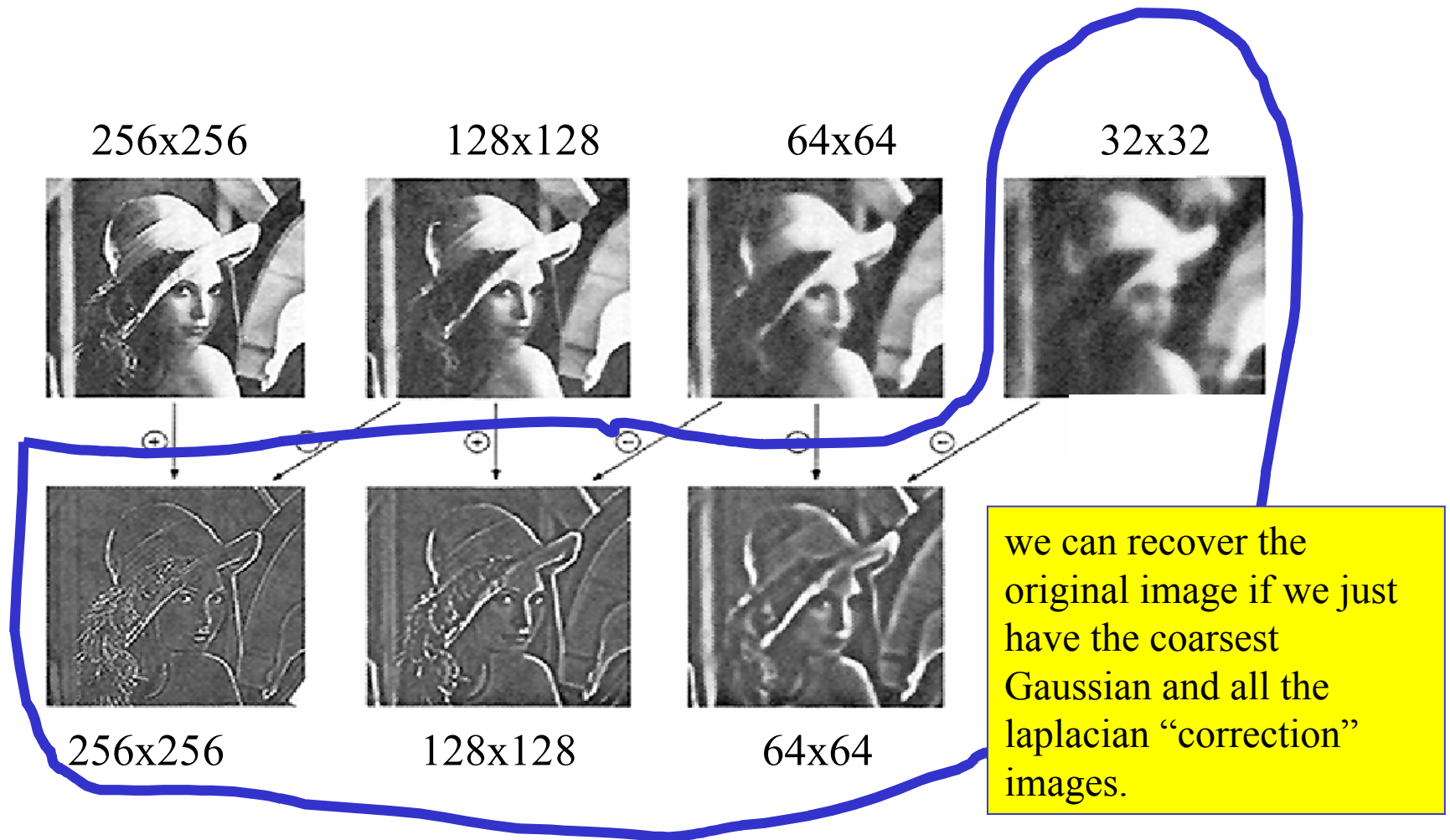


Laplacian Pyramid (cont)

- Coarse layer of the Gaussian pyramid predicts the appearance of the next finer layer.
- This prediction is not exact
- Laplacian pyramid stores the difference.



Lossless Image Decomposition

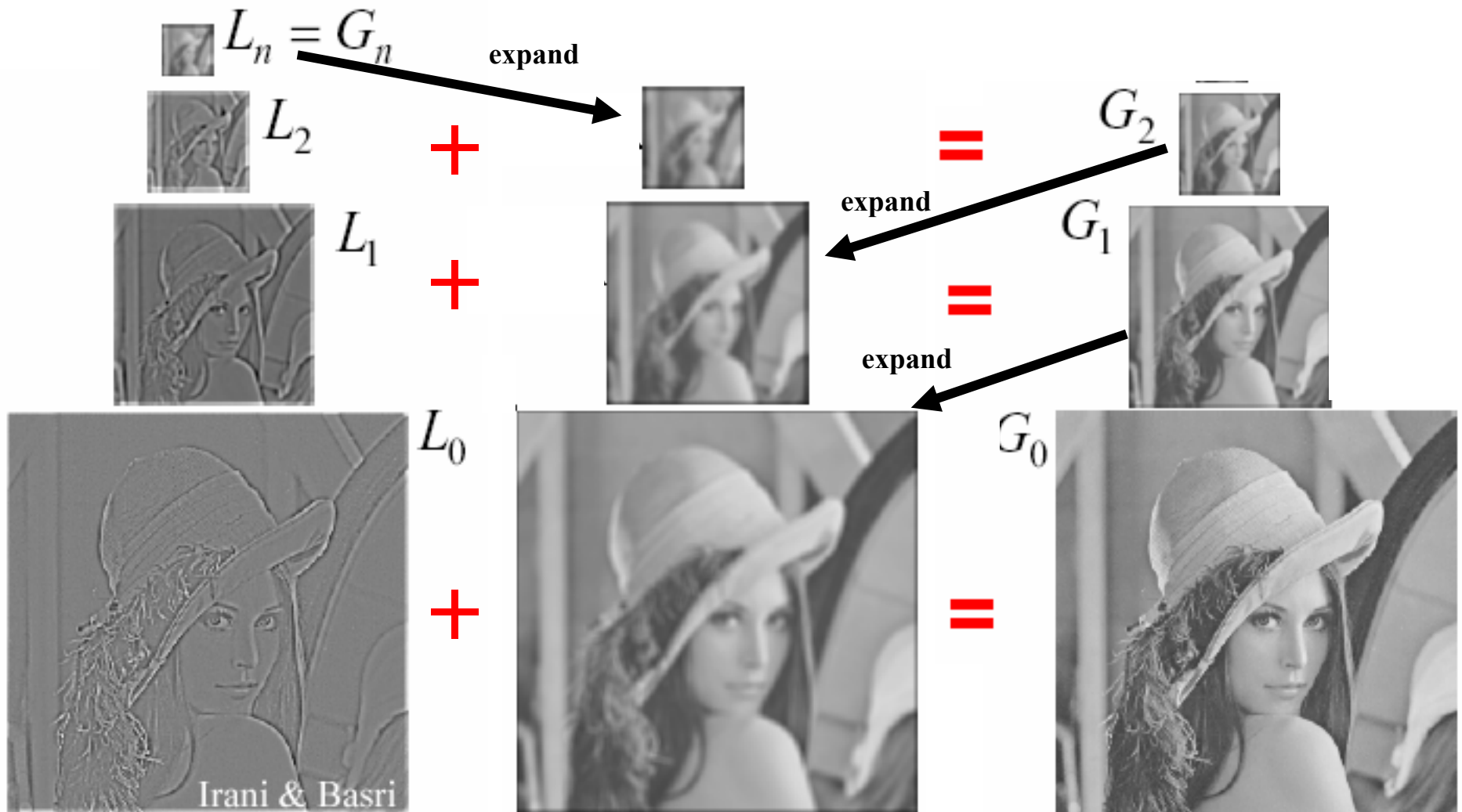


The Laplacian Pyramid as a Compact Image Code Burt, P., and Adelson, E. H.,
IEEE Transactions on Communication, COM-31:532-540 (1983).

Recovering the Original Image

$$L_i = G_i - \text{expand}(G_{i+1})$$

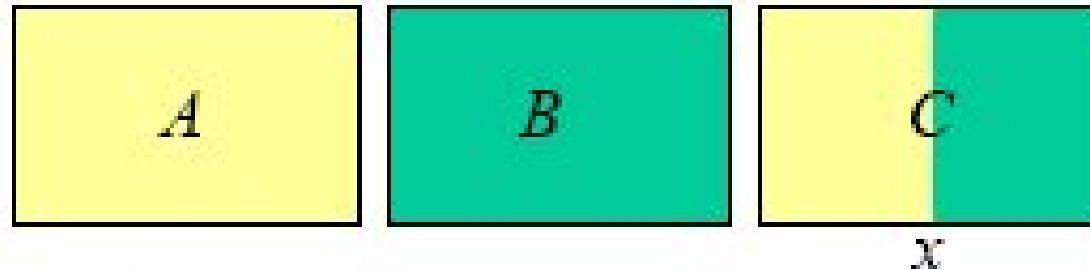
$$G_i = L_i + \text{expand}(G_{i+1})$$



Pyramid Blending

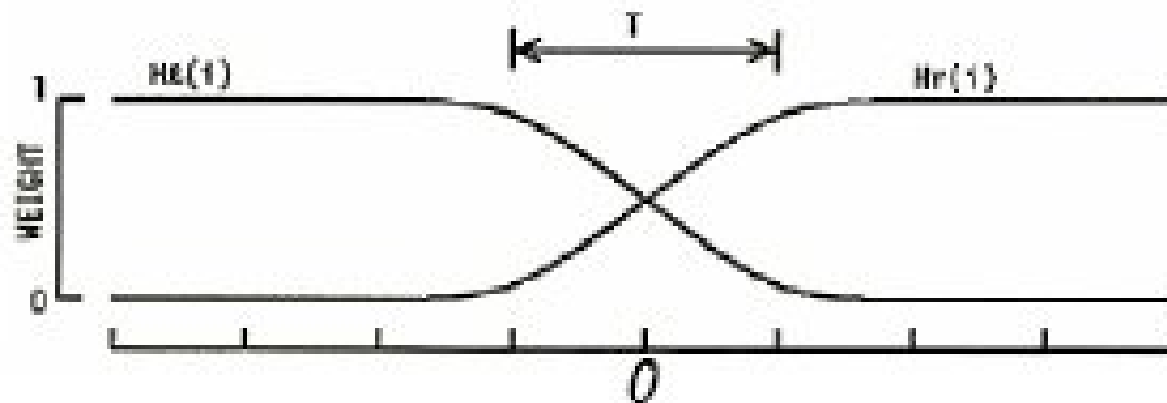
Burt and Adelson, “A multiresolution spline with application to image mosaics,” ACM Transactions on Graphics, 1983, Vol.2, pp.217-236.

Picture Merging with Spline

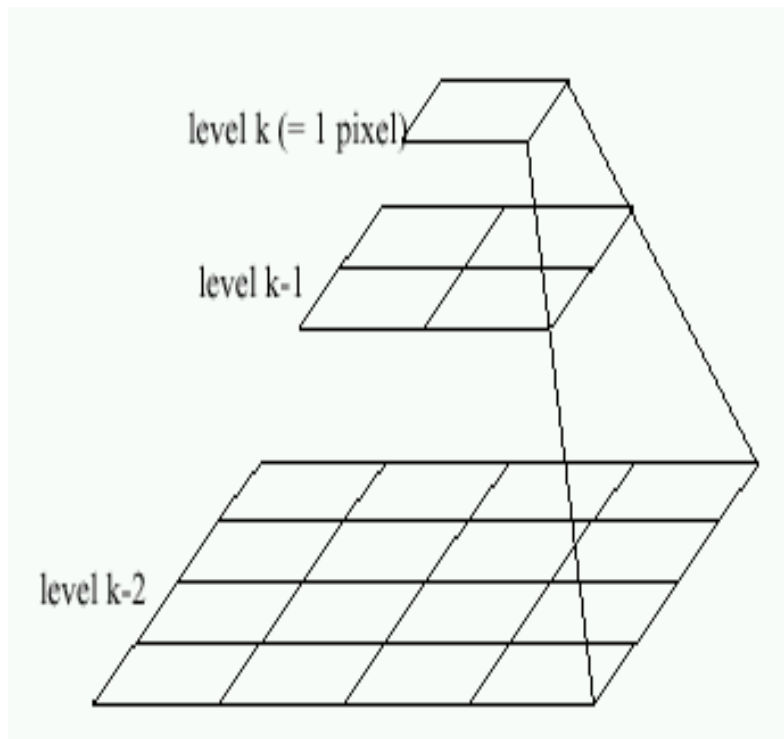


For every Row:

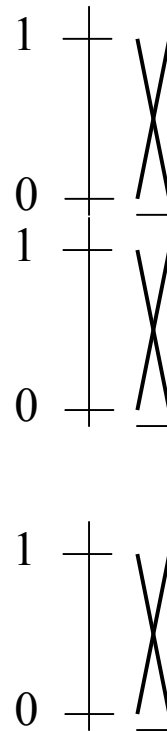
$$C(i) = H_l(i-x)A(i) + H_r(i-x)B(i)$$



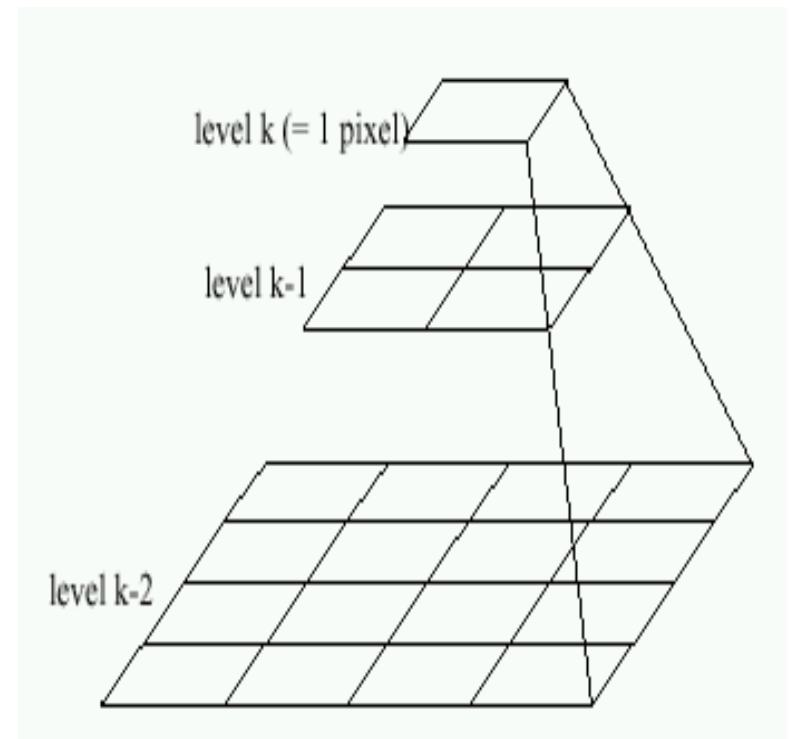
Pyramid Blending



Left pyramid

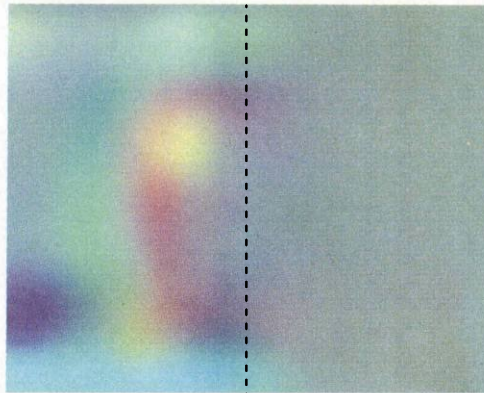


blend

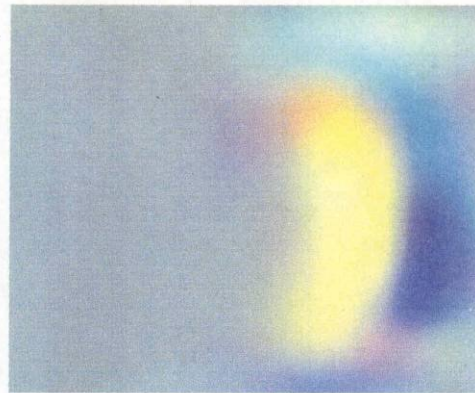


Right pyramid

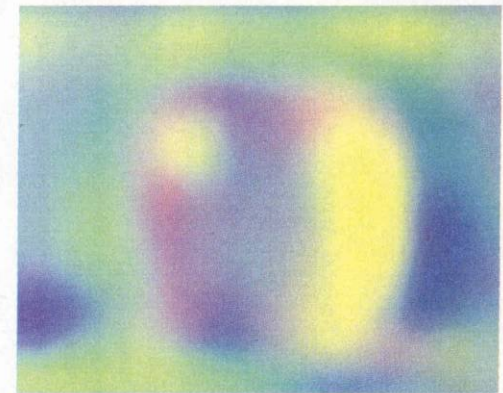
laplacian
level
4



(c)

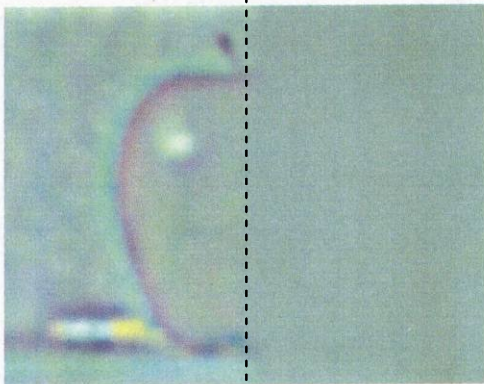


(g)

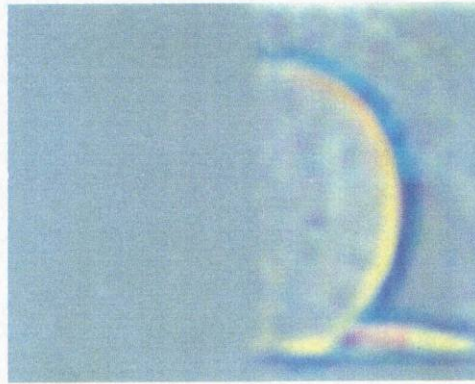


(k)

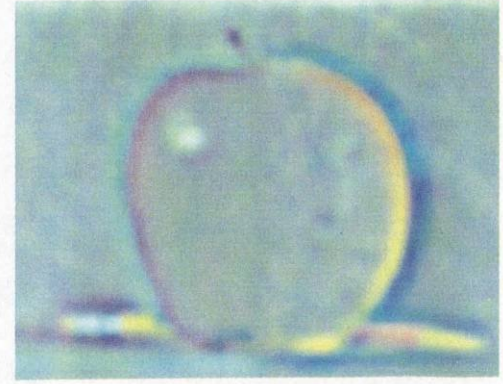
laplacian
level
2



(b)

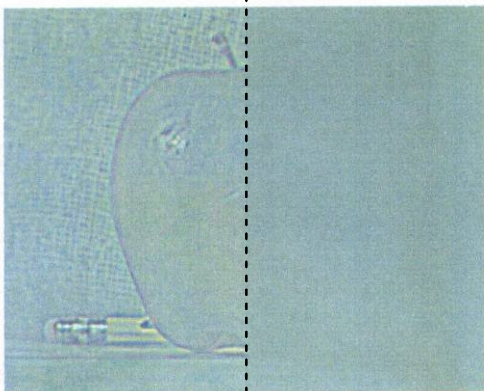


(f)

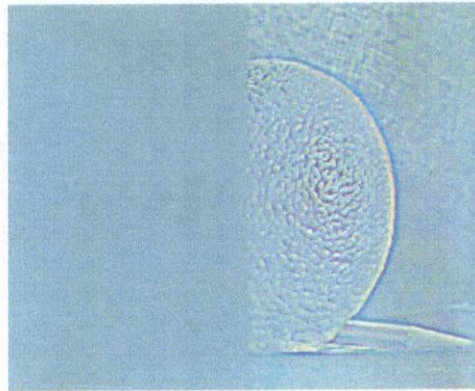


(j)

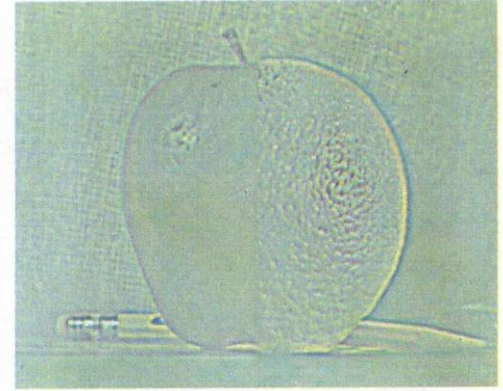
laplacian
level
0



(a)



(e)



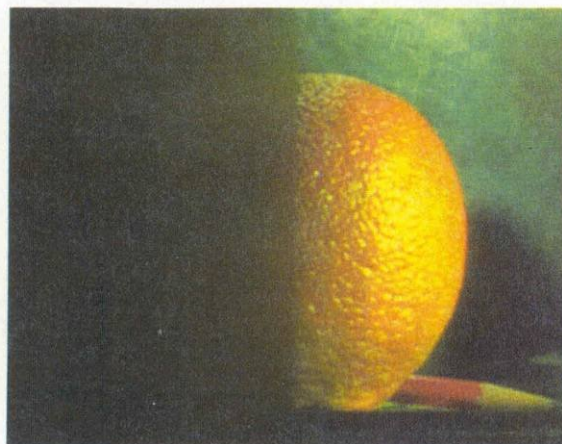
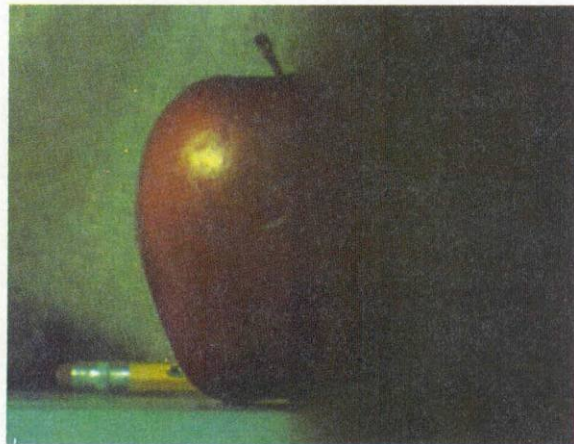
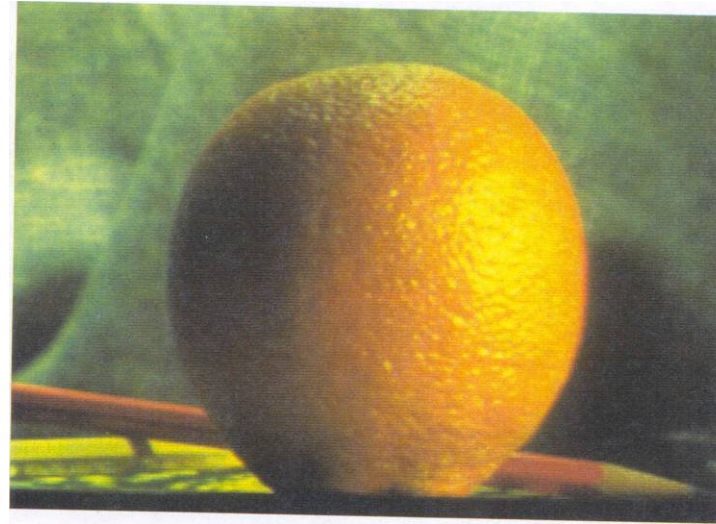
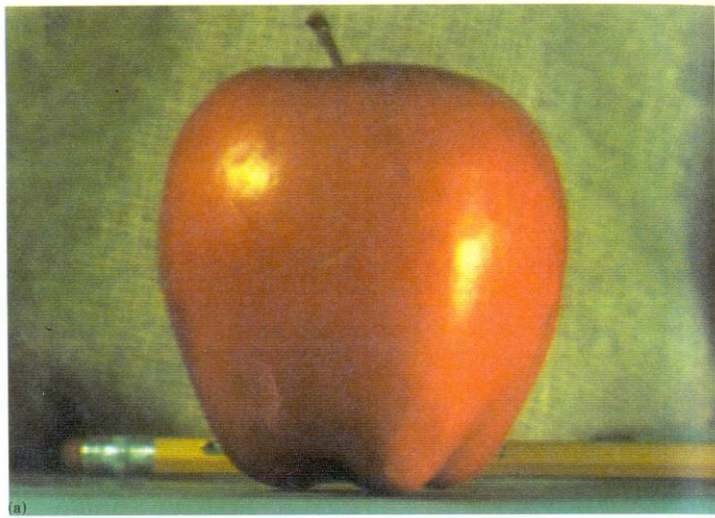
(i)

left pyramid

right pyramid

blended pyramid

Pyramid Blending



(d)

(h)

(l)

General Region Blending

- General Approach:
 1. Build Laplacian pyramids LA and LB from images A and B
 2. Build a Gaussian pyramid GR from selected region R (*mask that says which pixels come from left and which from right*)
 3. Form a combined pyramid LS from LA and LB using nodes of GR as weights:
 - $LS(i,j) = GR(I,j) * LA(I,j) + (1-GR(I,j)) * LB(I,j)$
 4. Collapse the LS pyramid to get the final blended image

Blending Regions

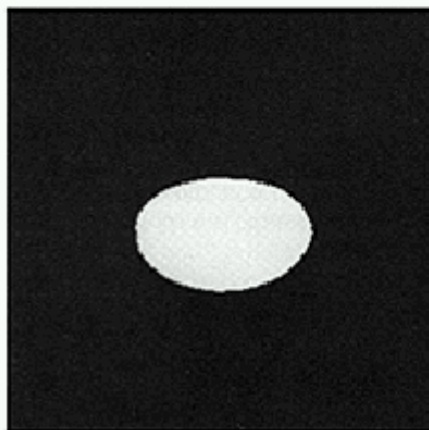
left



right



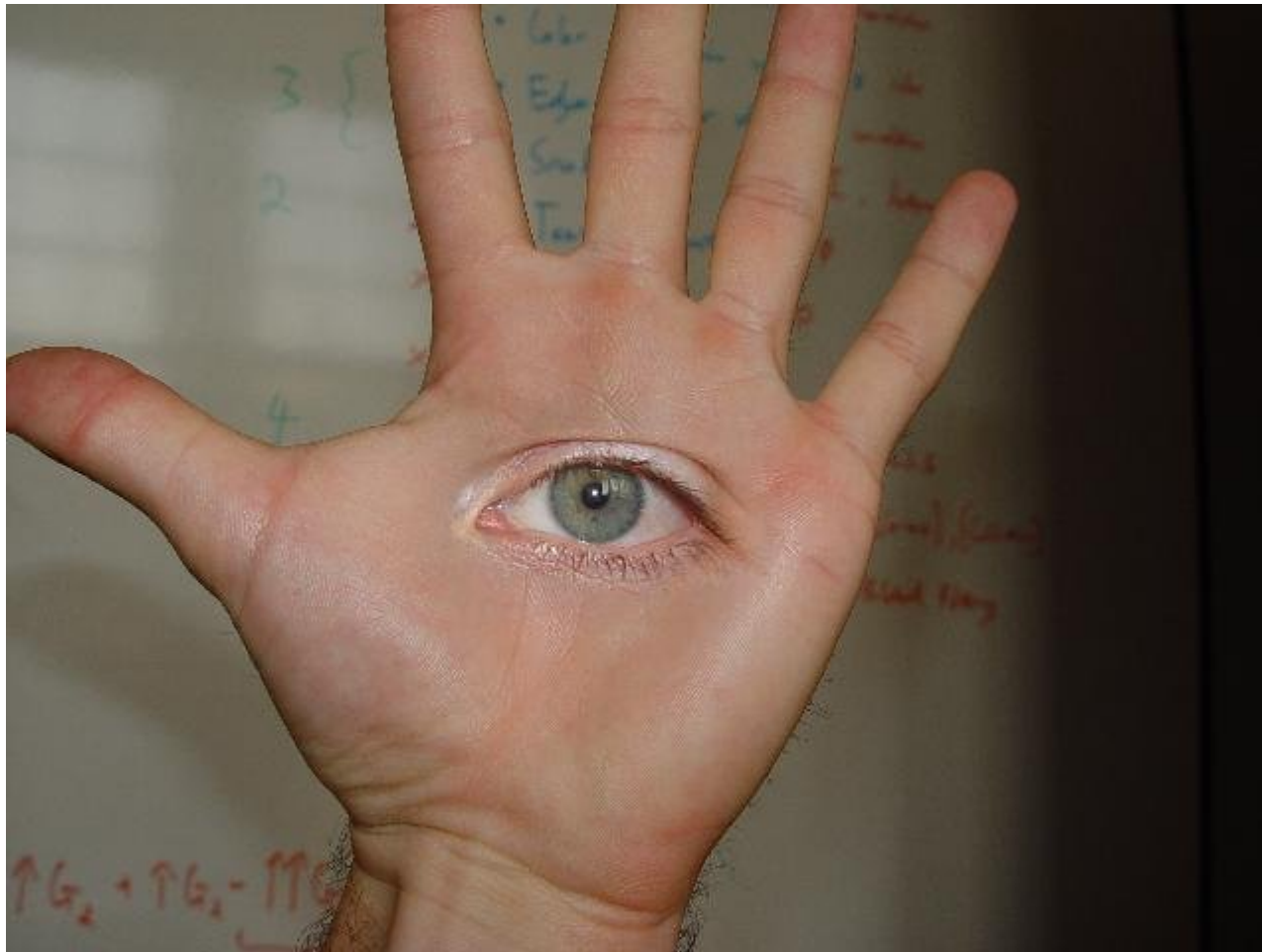
mask



blended



Horror Photo



© david dmartin (Boston College)

CMU Comp Photo Fall 2005

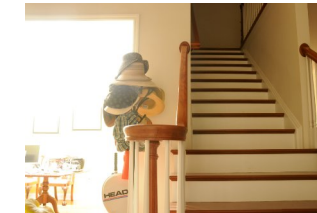
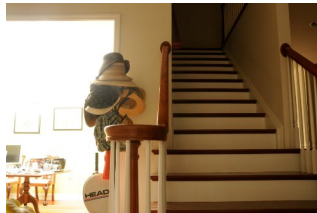
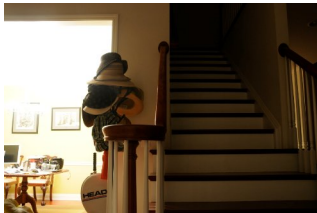
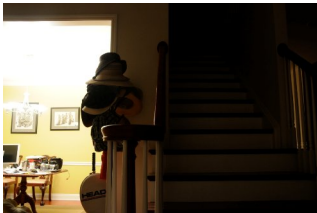
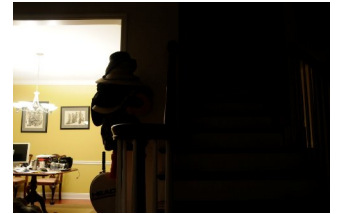
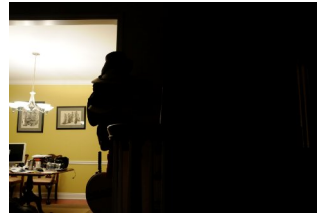
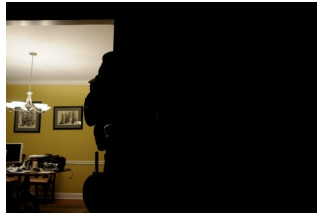
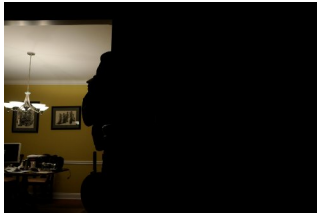


© Chris Cameron

<http://wiki.panotools.org/Enfuse>



Example



How it works

Image fusion maps each pixel $P(i, x, y)$ of every input image i to a single pixel $Q(x, y)$ in the output image: $P(i, x, y) \rightarrow Q(x, y)$, where x runs from 1 to the common width of the images, y from 1 to the common height, and i from 1 to the number of input images n .

Enfuse allows for weighting the contribution of each

$$w(P(1, x, y))P(1, x, y) + \dots + w(P(n, x, y))P(n, x, y) \rightarrow Q(x, y), (W)$$

where

- each w is non-negative to yield a physical intensity and
- the sum of all w is one to leave the total intensity unchanged.

The pixel weights w themselves are weighted sums with the same constraints

$$\begin{aligned} w(P) = & w_{\text{exp}} f_{\text{exp}}(P) + \\ & w_{\text{sat}} f_{\text{sat}}(P) + \\ & w_{\text{cont}} f_{\text{cont}}(P, r_{\text{cont}}) + \\ & w_{\text{ent}} f_{\text{ent}}(P, r_{\text{ent}}), \end{aligned}$$



The exposures + weight masks are then blended by pyramid blending (using another program called “Enblend”)

How it works

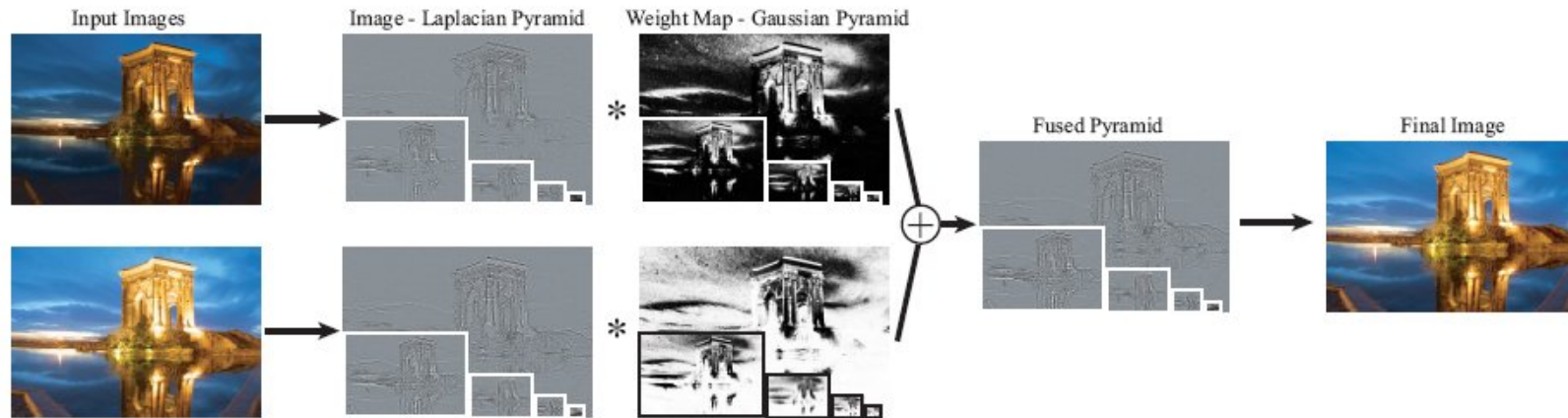
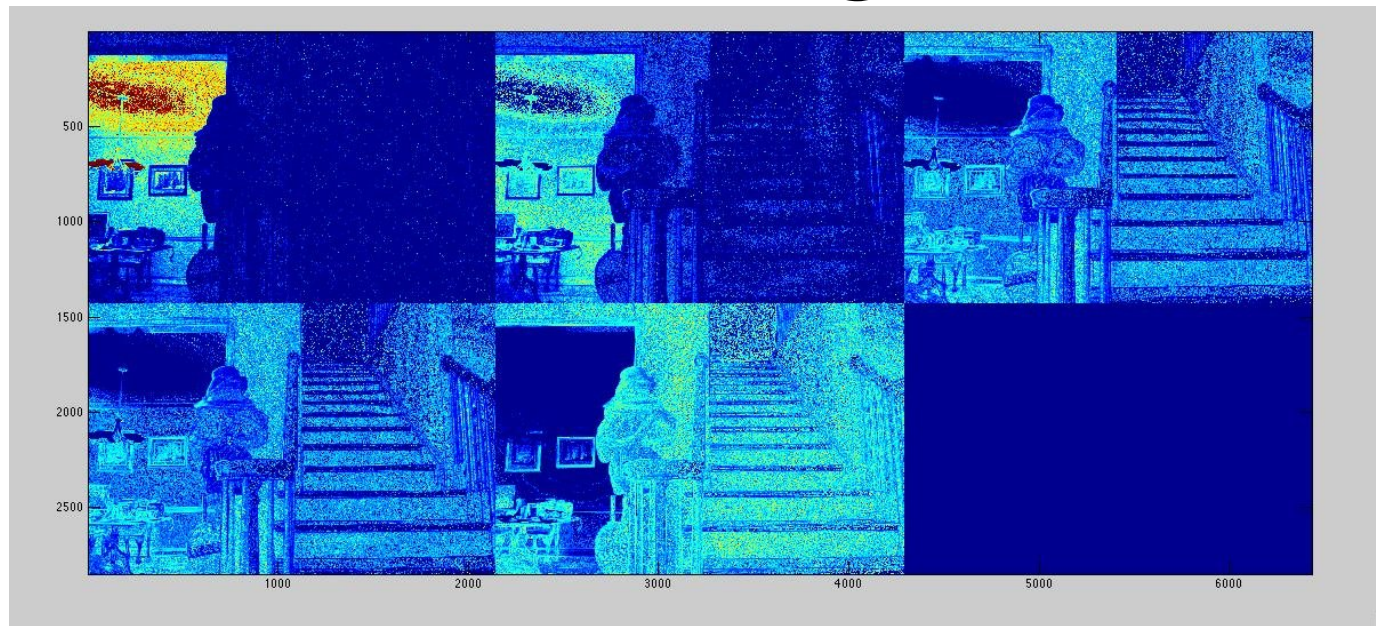


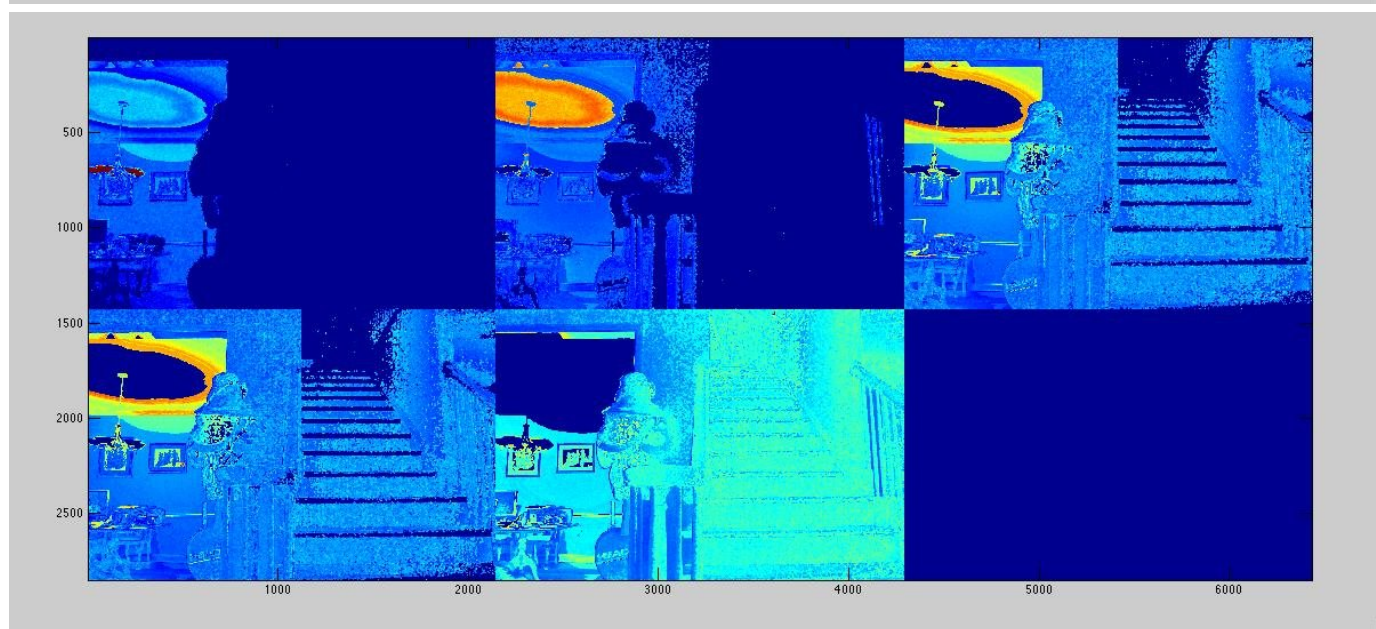
Figure 3. We fuse differently exposed images using a Laplacian decomposition of the images and a Gaussian pyramid of the weight maps, which represent measures such as contrast and saturation. Image courtesy of Jacques Joffre.

http://research.edm.uhasselt.be/~tmertens/papers/exposure_fusion_reduced.pdf

Weight Masks

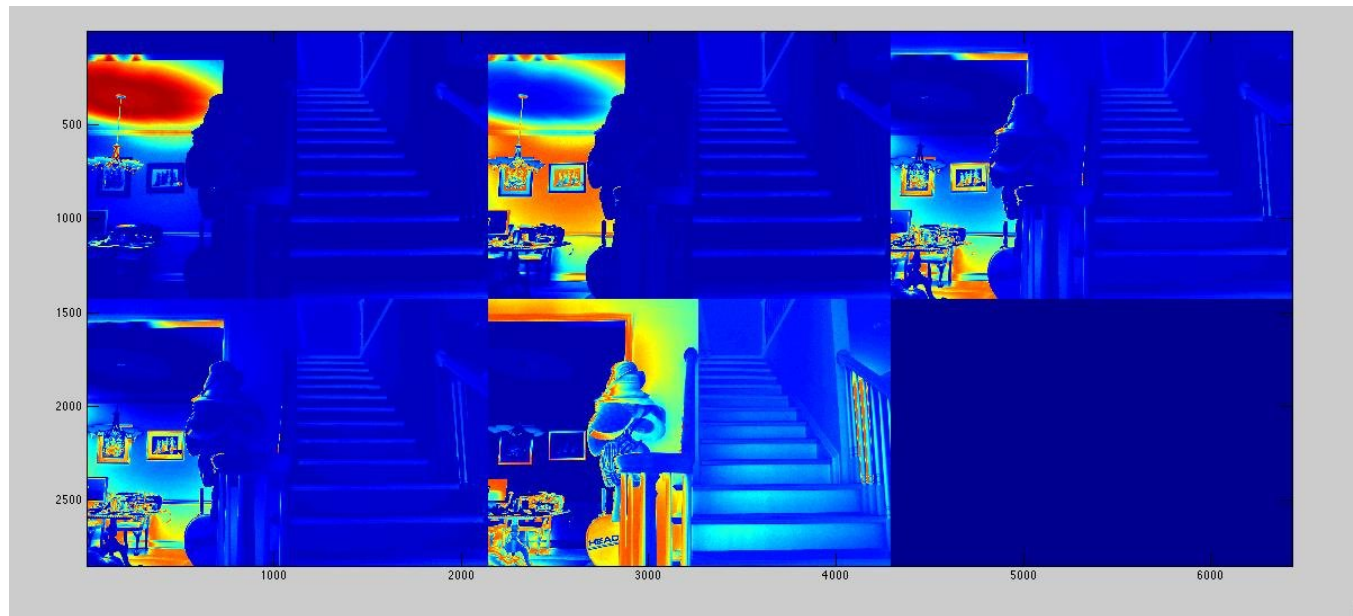


**considering
contrast
only**

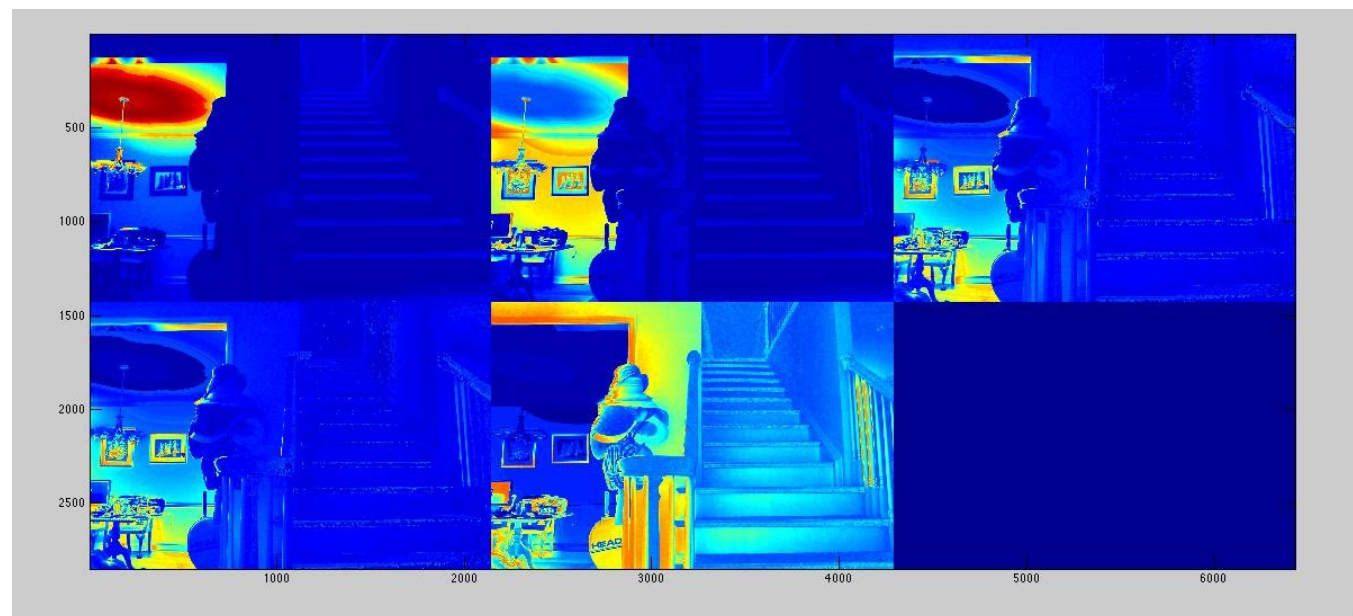


**considering
saturation
only**

Weight Masks



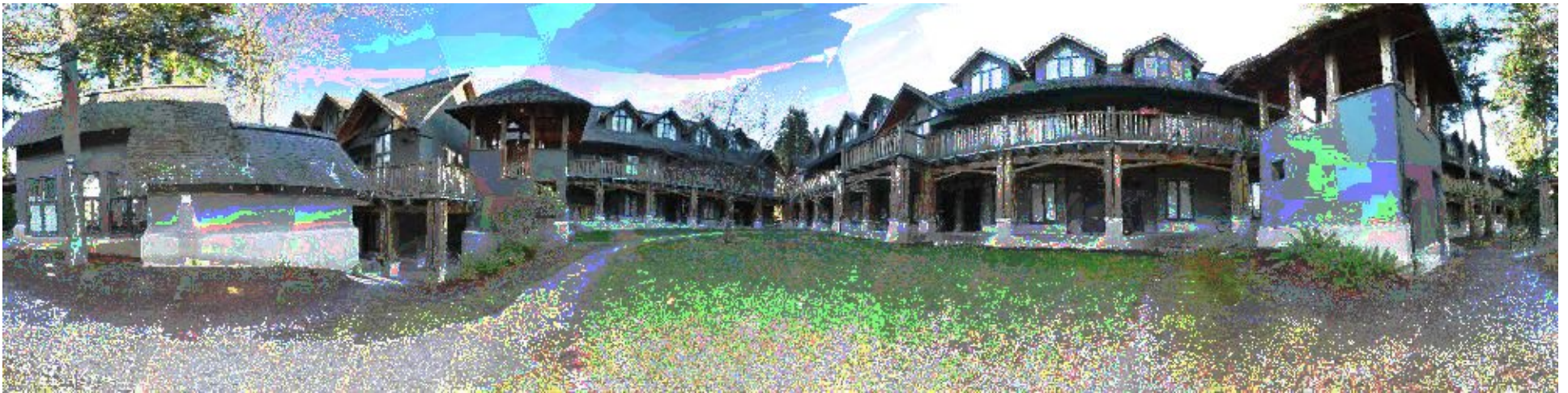
**considering
exposure
only**



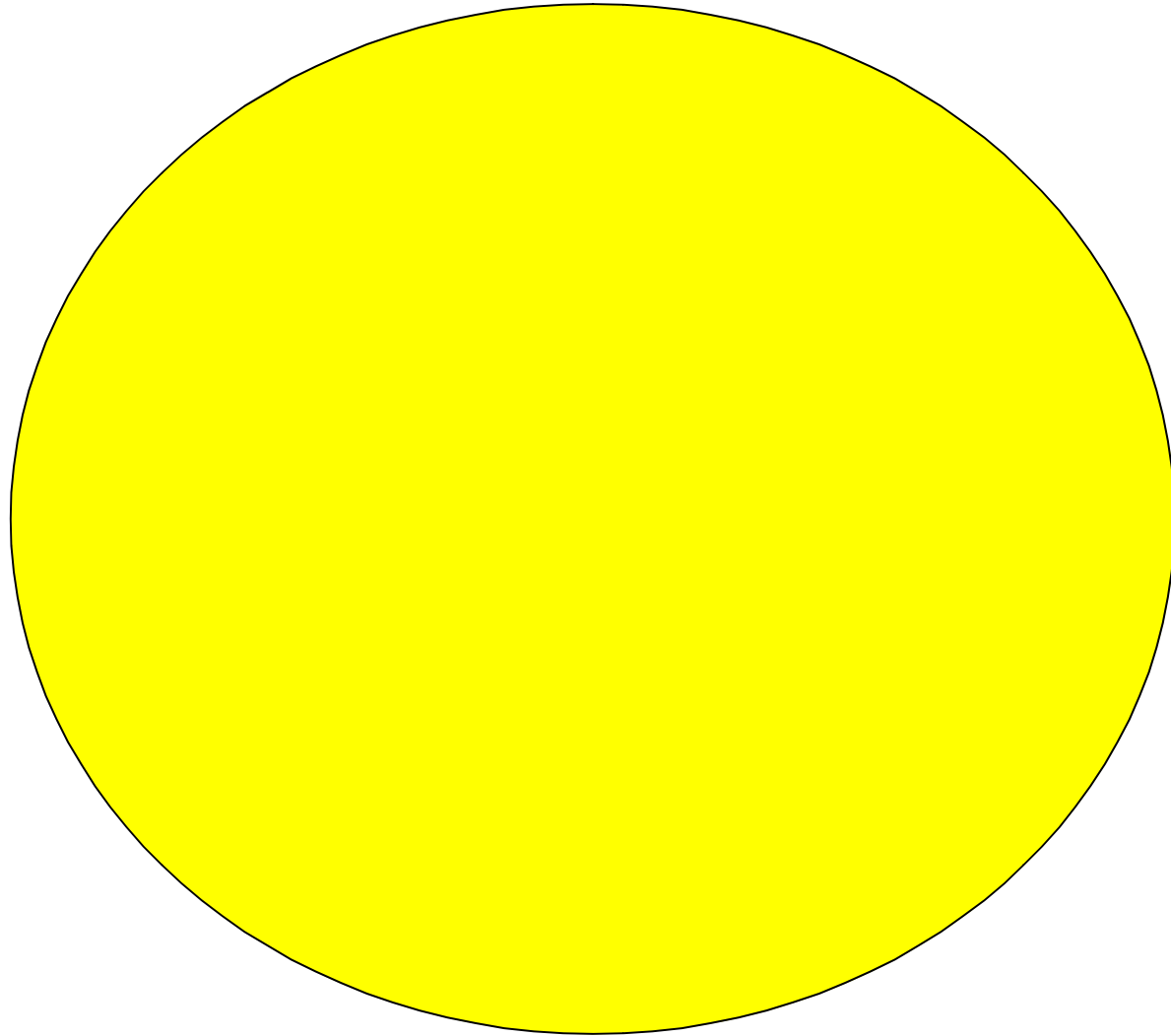
**default
.77*contrast+
.23*saturation**

Simplification: Two-band Blending

- Brown & Lowe, 2003
 - Only use two bands: high freq. and low freq.
 - Blends low freq. smoothly
 - Blend high freq. with no smoothing: use binary alpha



Stop Here



2-band Blending

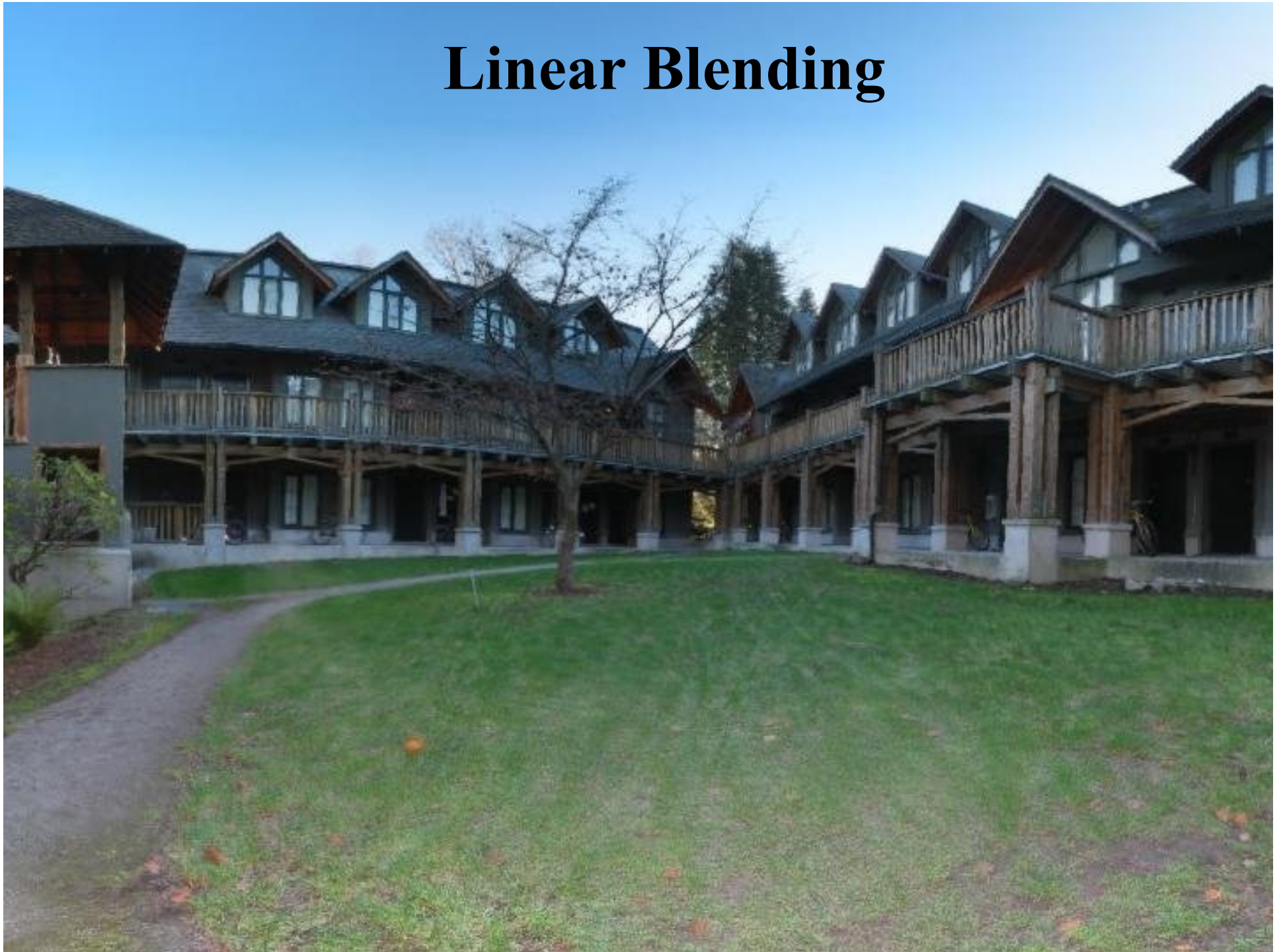


Low frequency ($\lambda > 2$ pixels)



High frequency ($\lambda < 2$ pixels)

Linear Blending



2-band Blending

