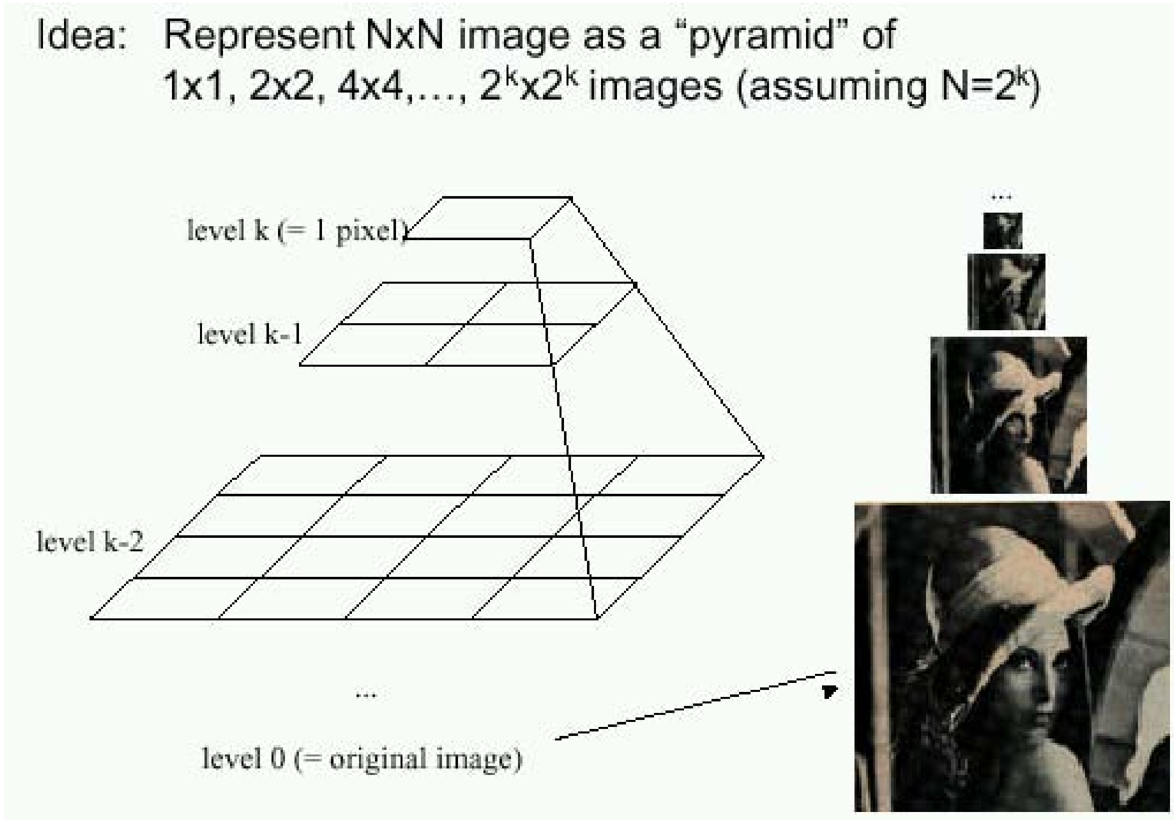
Image Pyramids and Blending

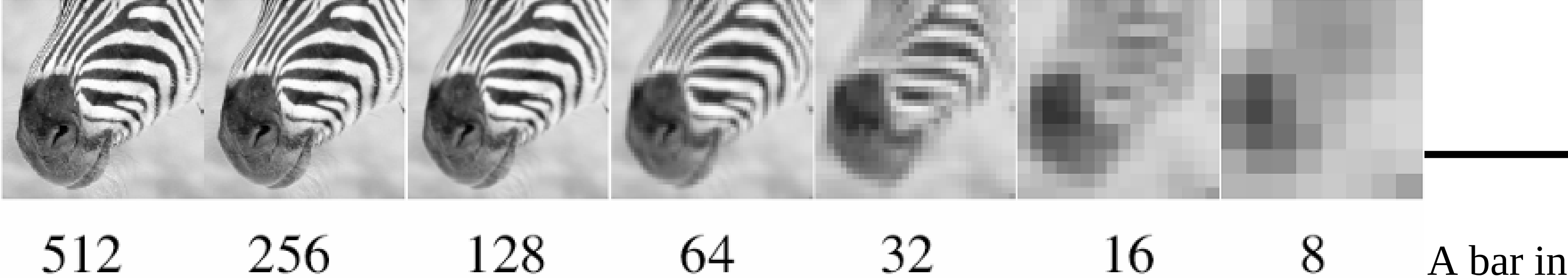
15-463: Computational Photography
Alexei Efros, CMU, Fall 2005

Image Pyramids



Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*



A bar in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose

Figure from David Forsyth

What are they good for?

Improve Search

- Search over translations
  - Like homework
  - Classic coarse-to-fine strategy
- Search over scale
  - Template matching
  - E.g. find a face at different scales

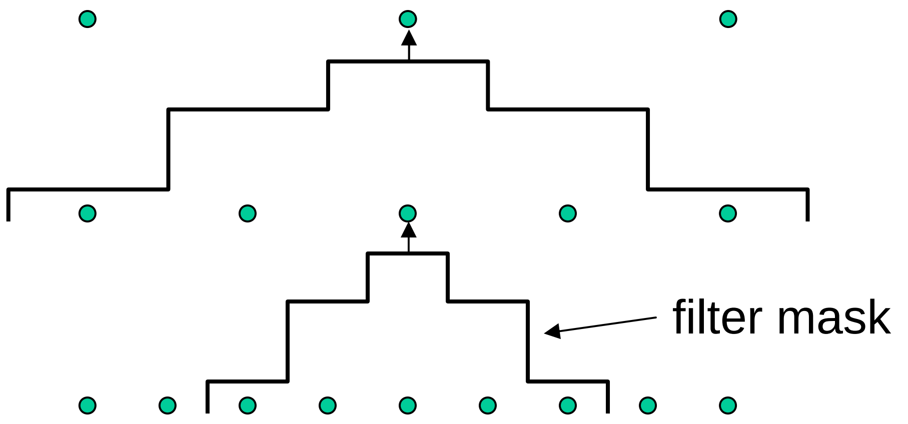
Precomputation

- Need to access image at different blur levels
- Useful for texture mapping at different resolutions (called mip-mapping)

Image Processing

- Editing frequency bands separately
- E.g. image blending…

Gaussian pyramid construction



Repeat

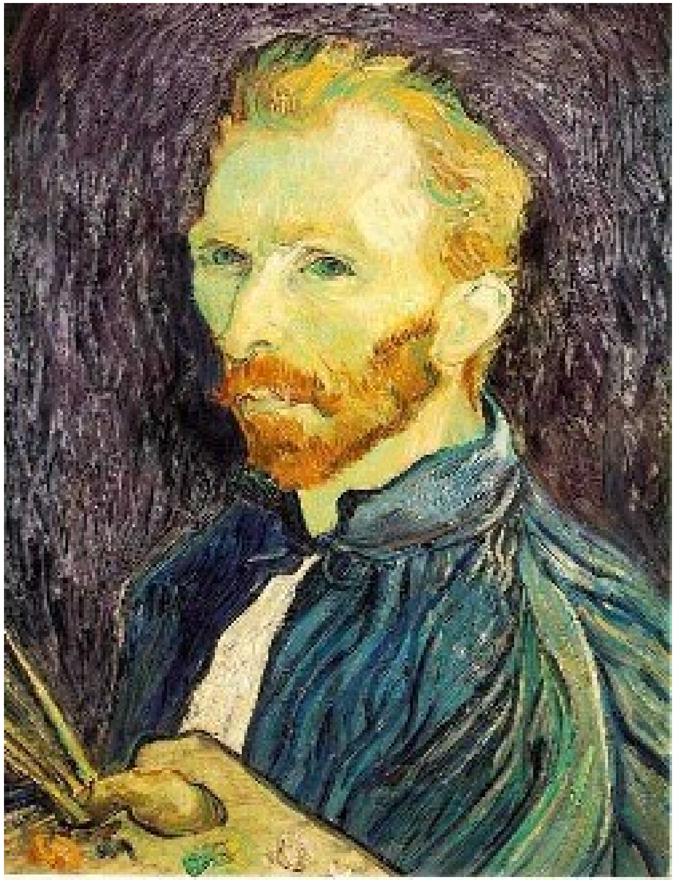
- Filter
- Subsample

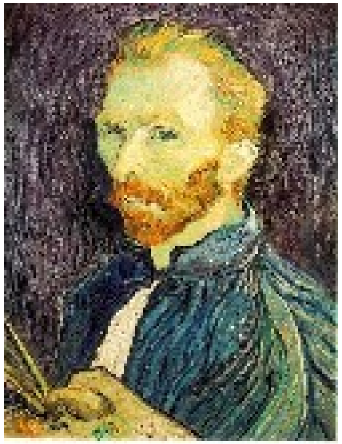
Until minimum resolution reached

- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

Image sub-sampling



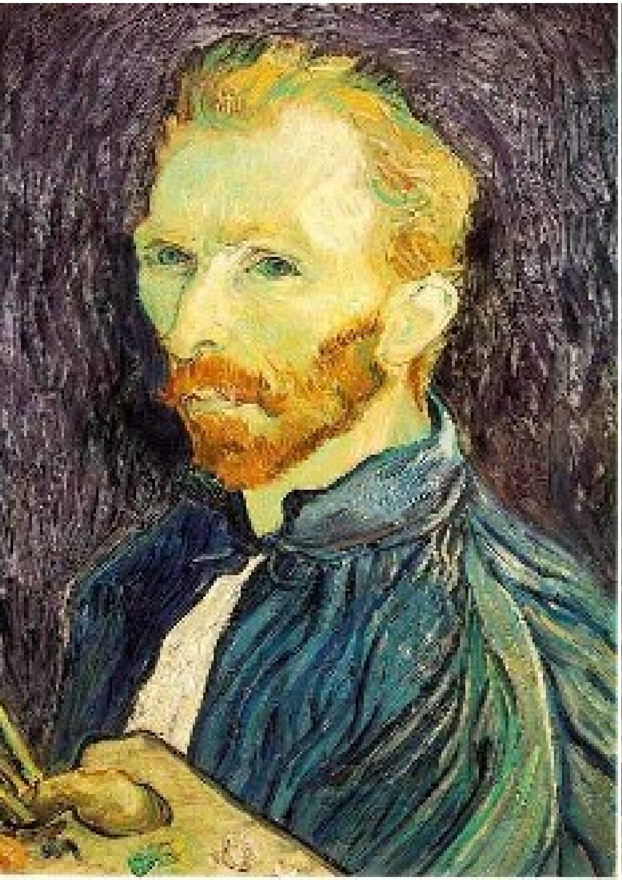


1/4

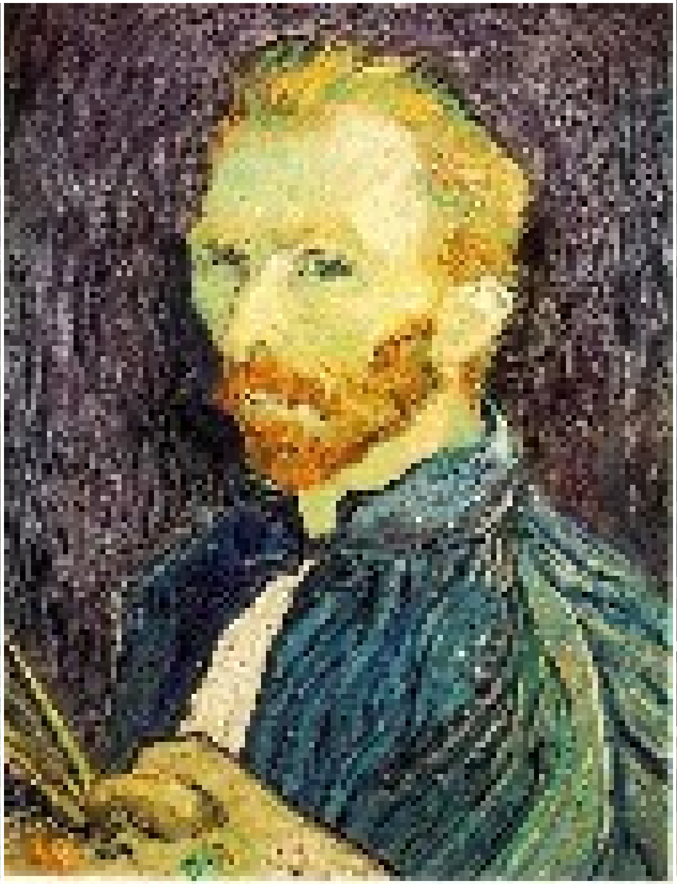
1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Image sub-sampling



1/2

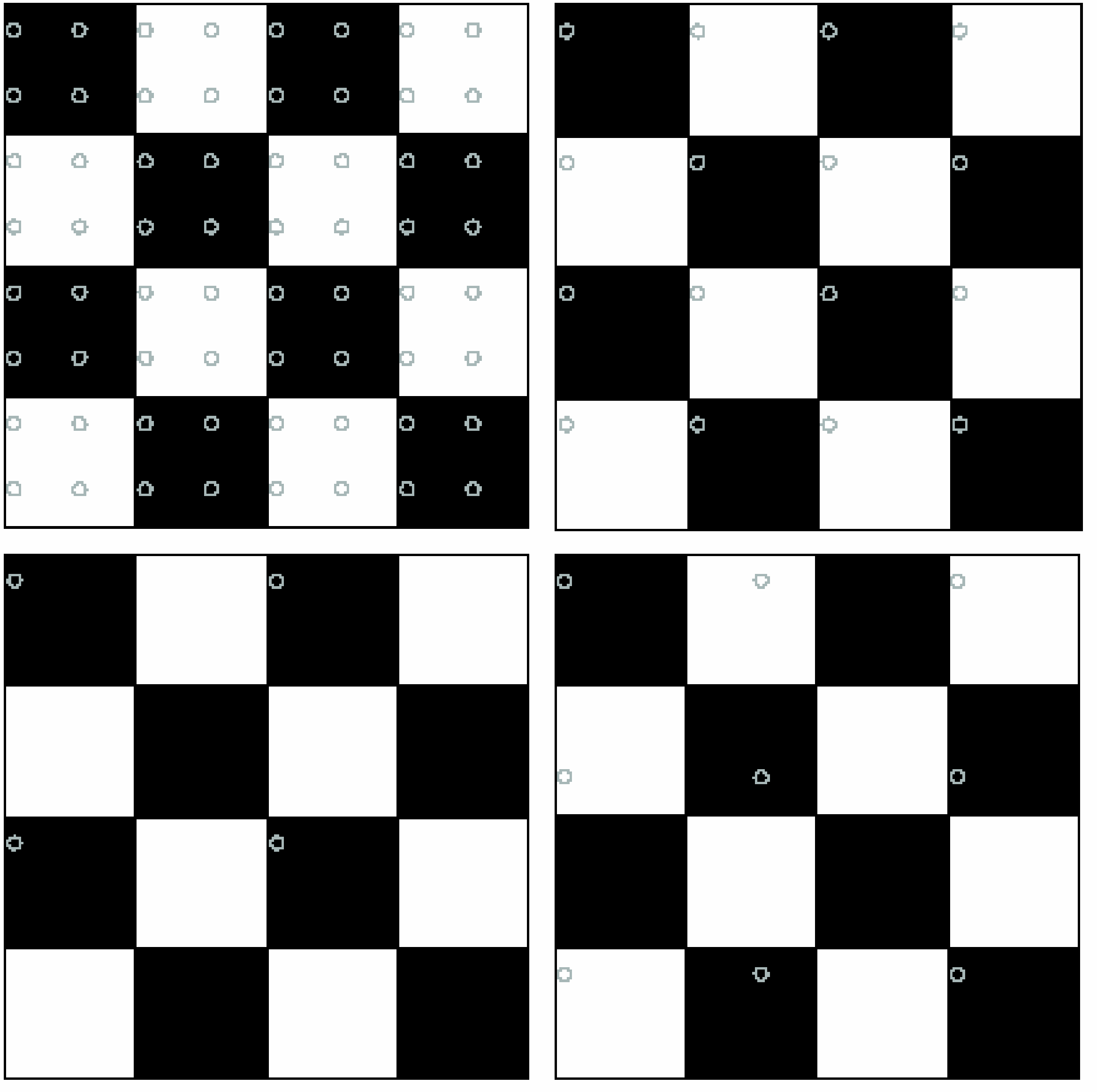


1/4 (2x zoom)

1/8 (4x zoom)

Why does this look so bad?

Sampling



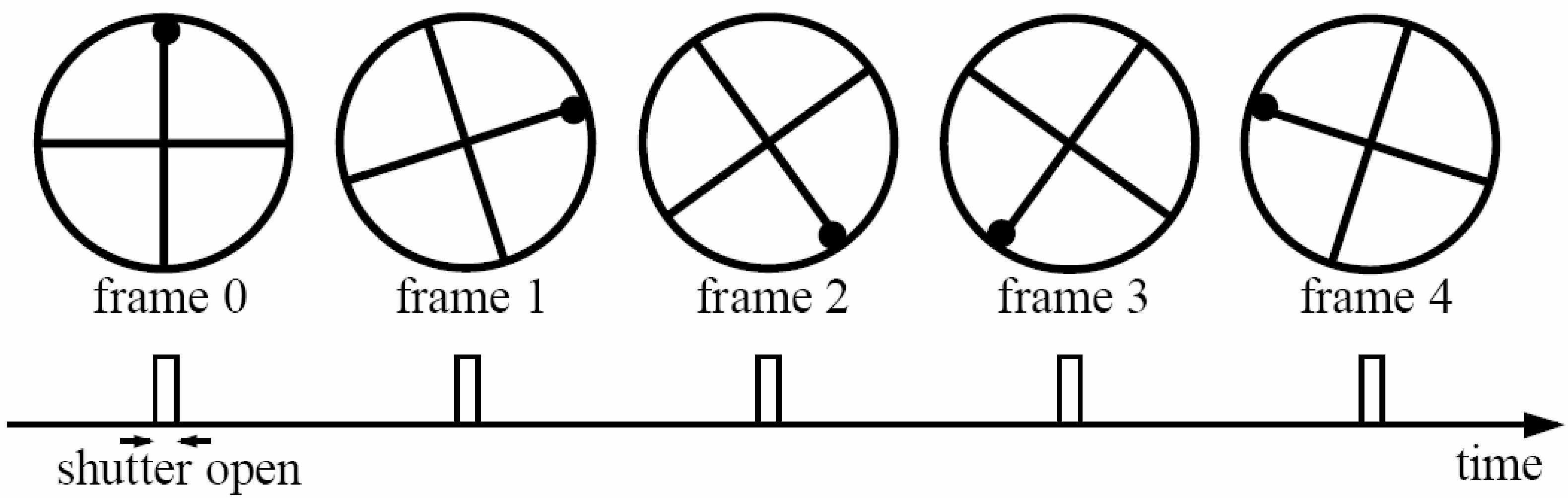
Good sampling:
- •Sample often or,
- •Sample wisely

Bad sampling:
- •see aliasing in action!

Really bad in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

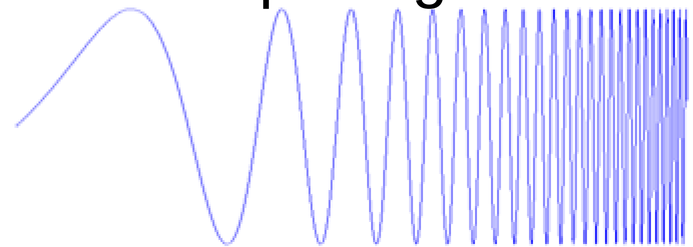
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



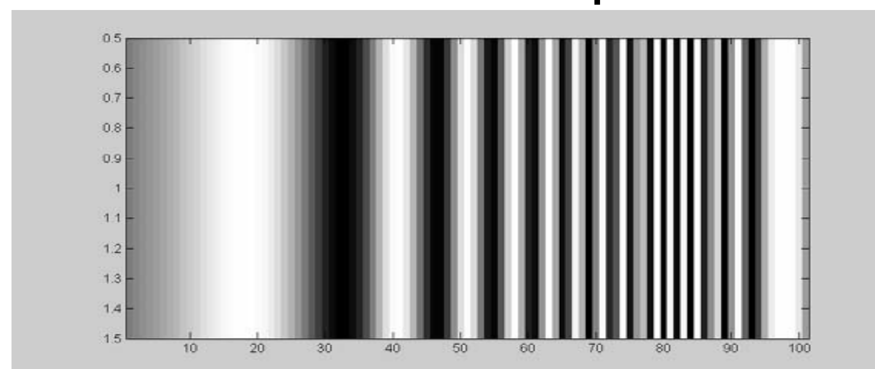
Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Alias: n., an assumed name

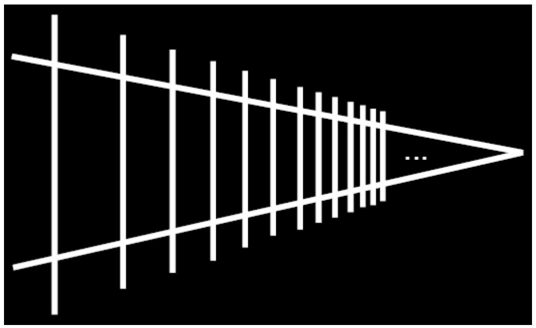
Input signal:



Matlab output:

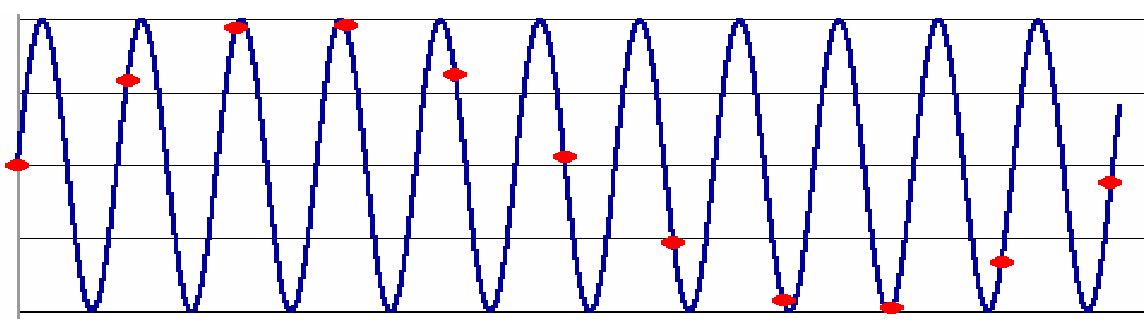


x = 0:.05:5; imagesc(sin((2.^x).*x))



Picket fence receding
Into the distance will
produce aliasing…

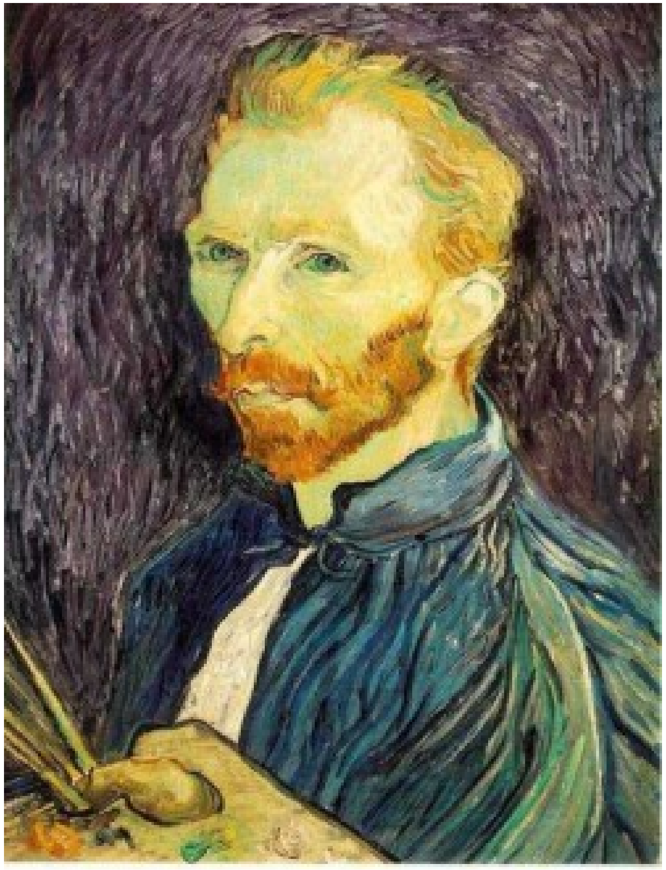
WHY?



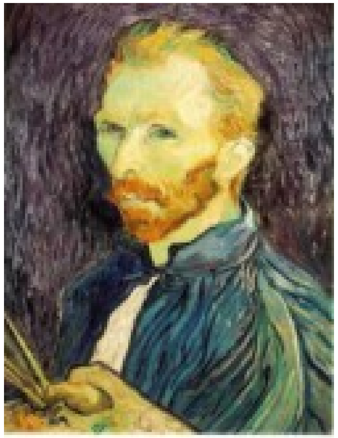
Aj-aj-aj:
Alias!

Not enough samples

Gaussian pre-filtering



Gaussian 1/2



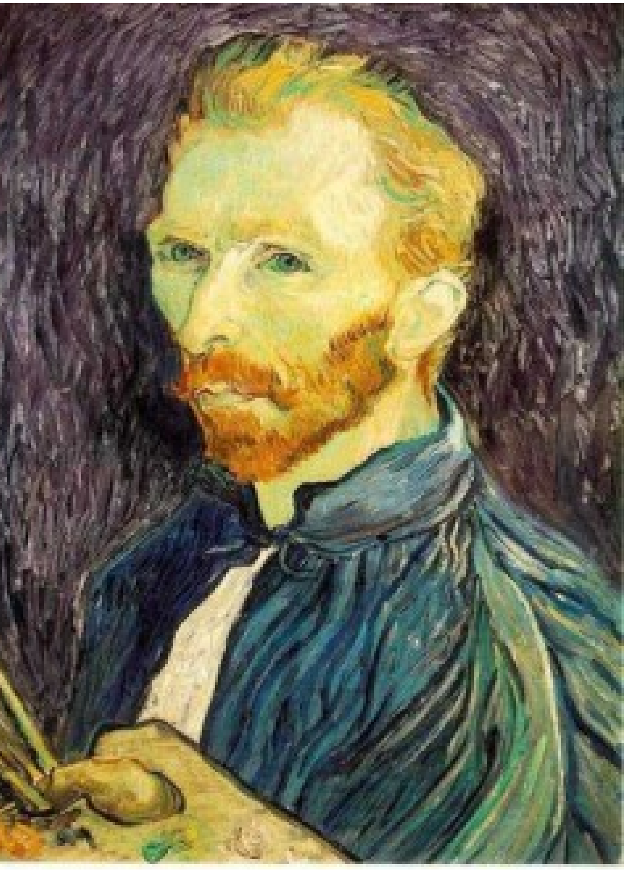
G 1/4

G 1/8

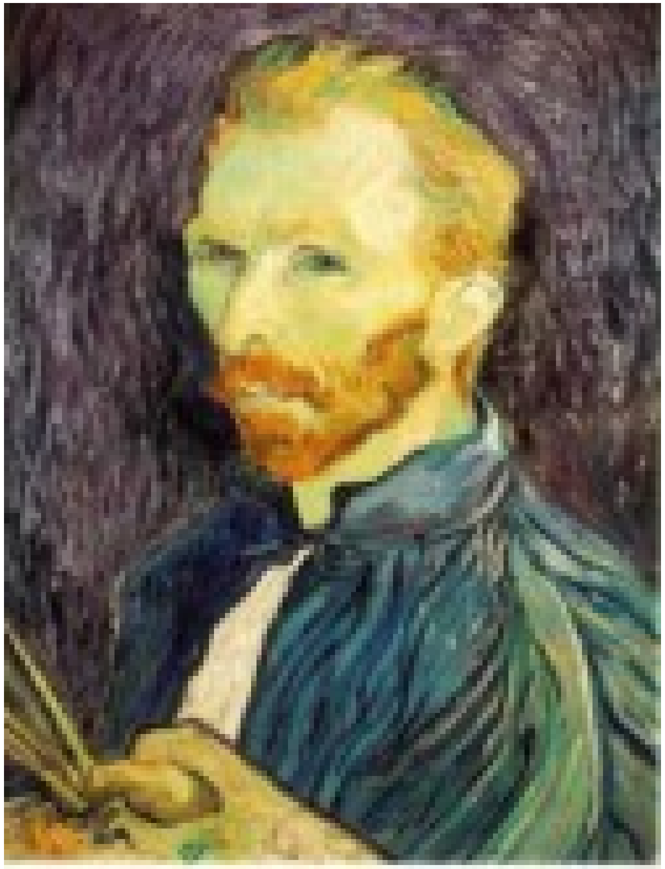
Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?

Subsampling with Gaussian pre-filtering



Gaussian 1/2



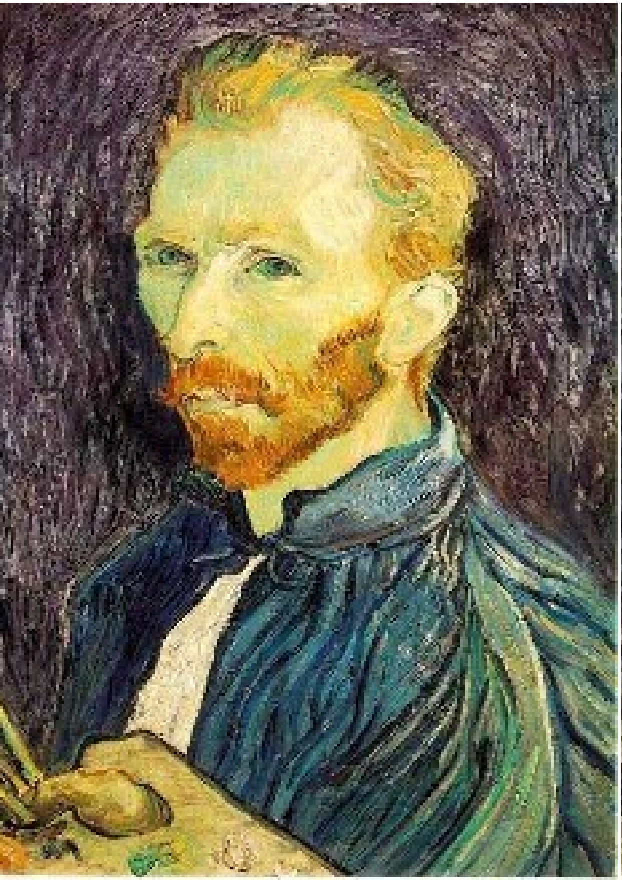
G 1/4

G 1/8

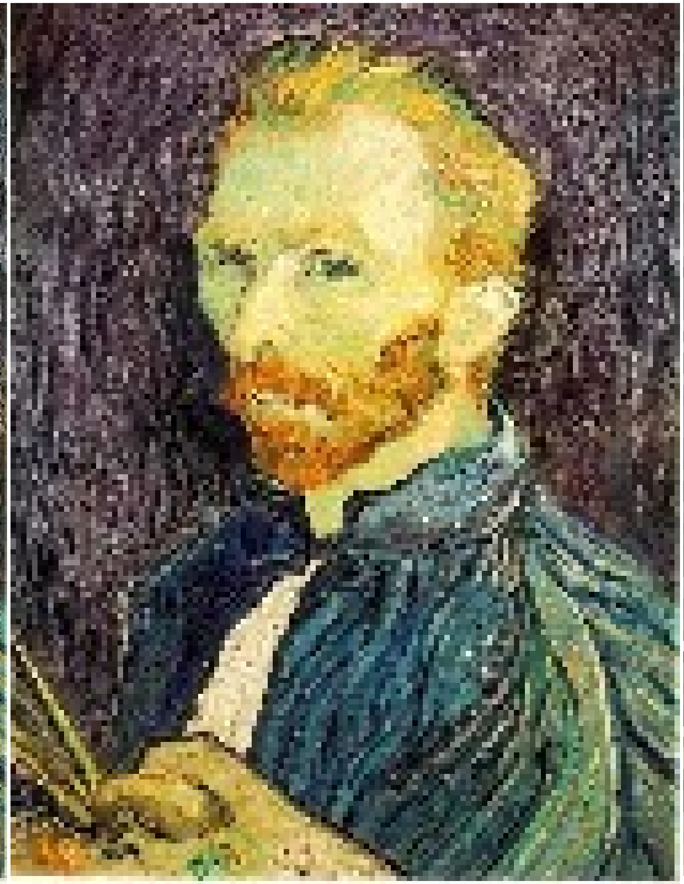
Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Compare with...



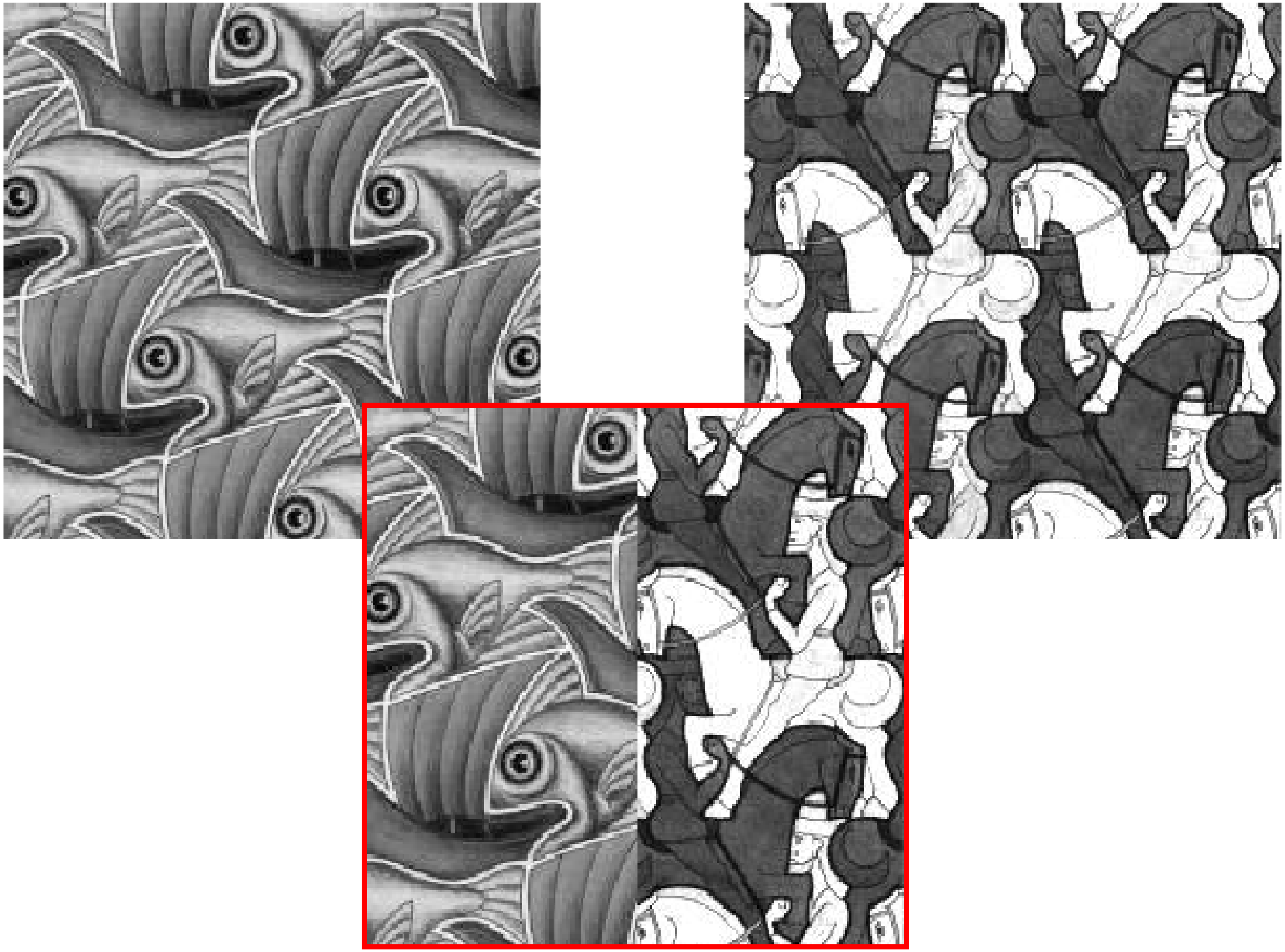
1/2



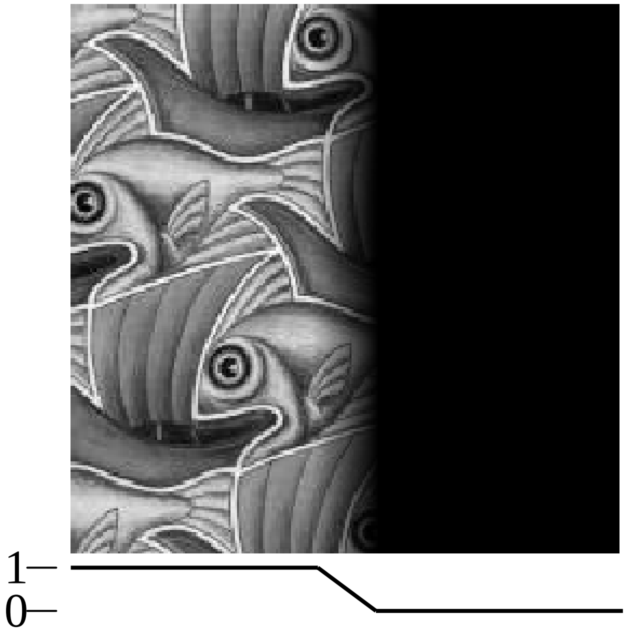
1/4 (2x zoom)

1/8 (4x zoom)

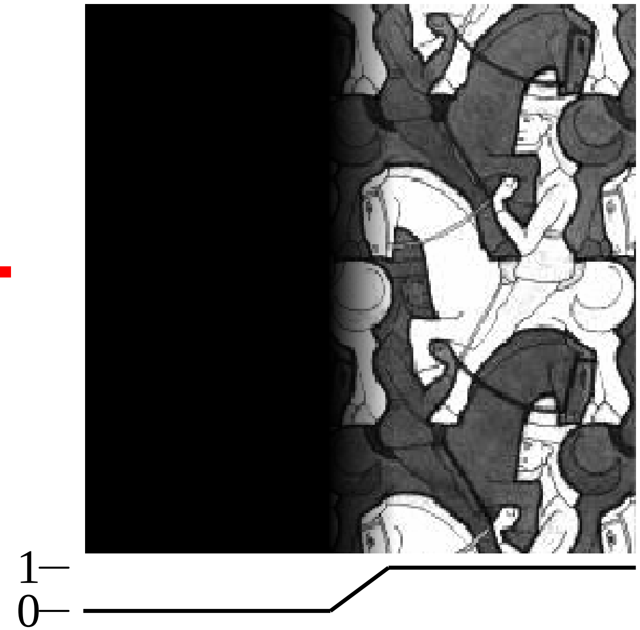
Image Blending



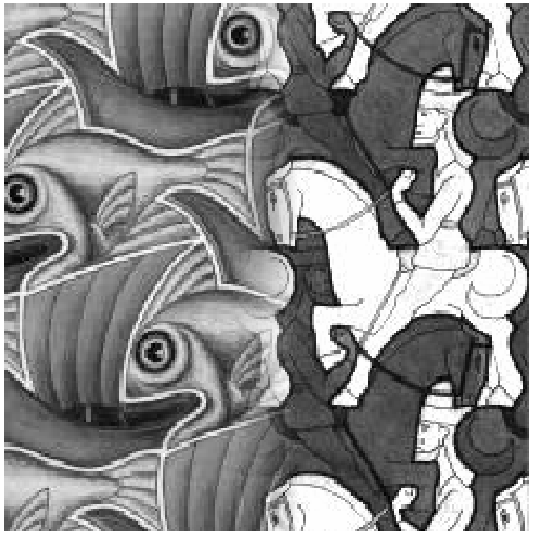
Feathering



+



=

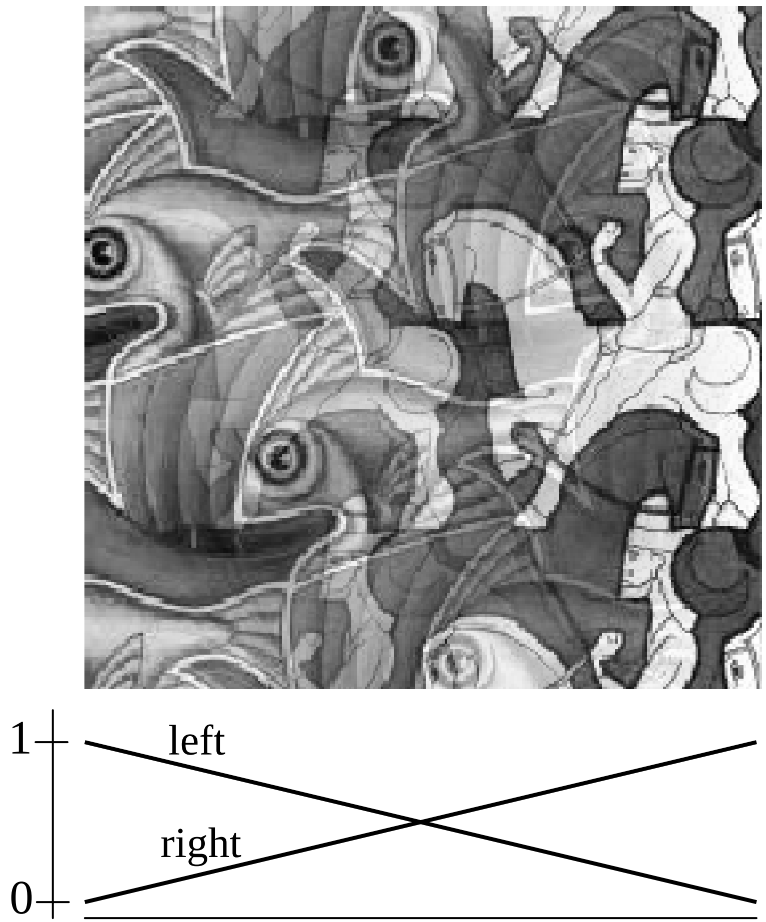


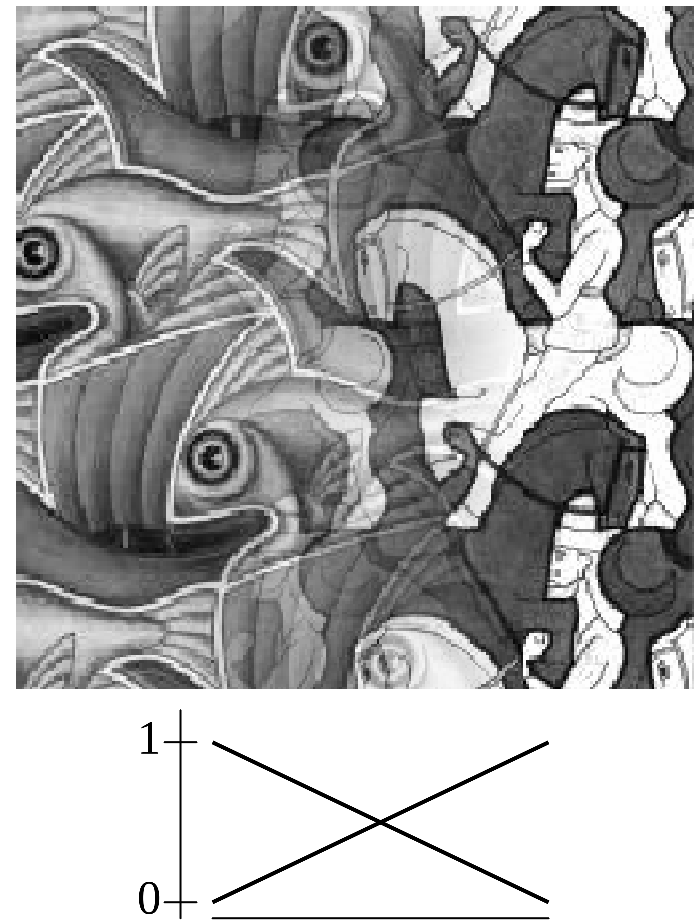
Encoding transparency

$I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha)$

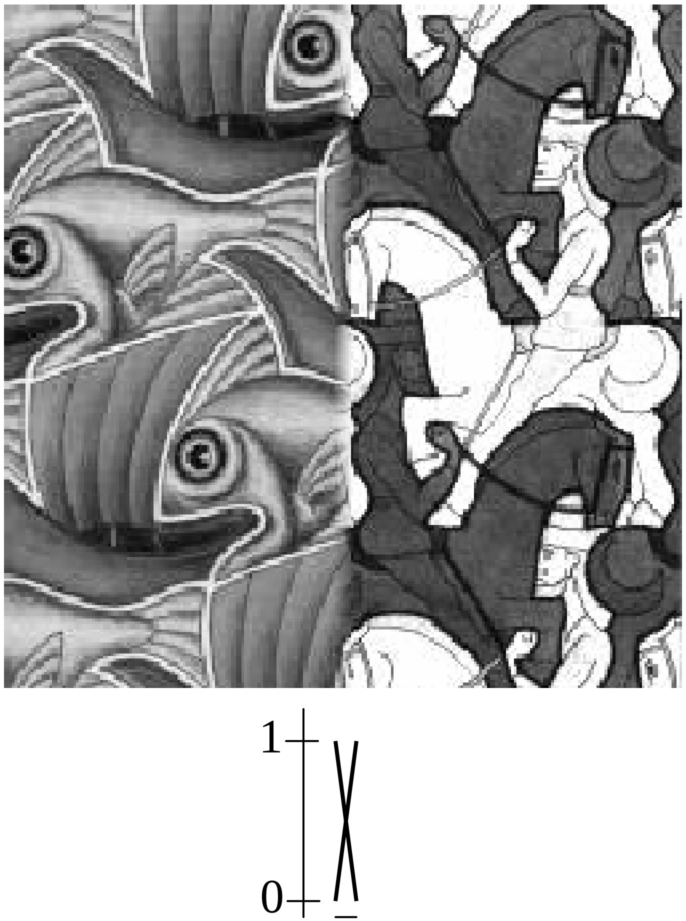
$I_{blend} = I_{left} + I_{right}$

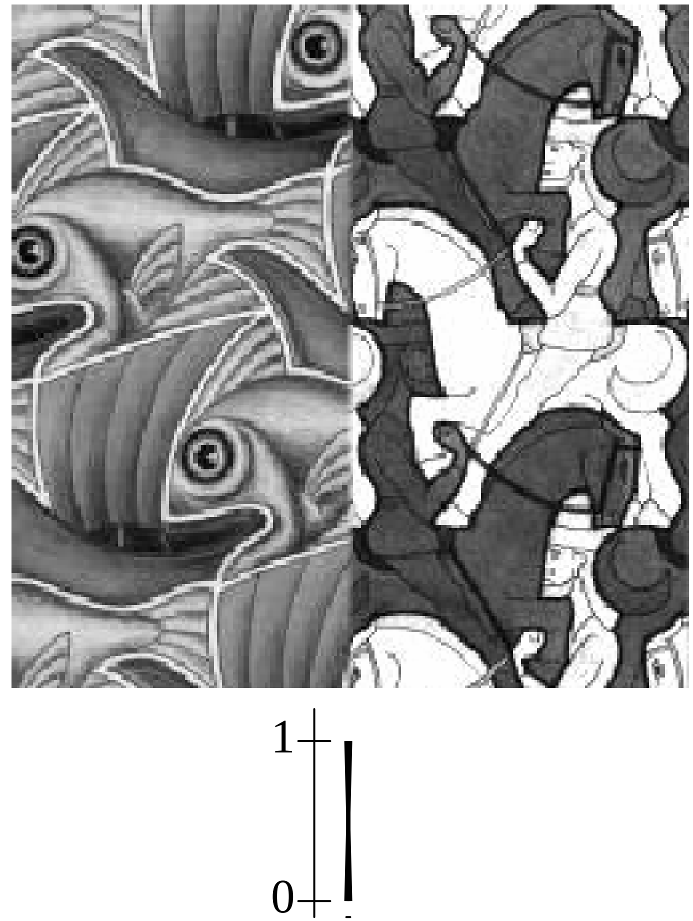
Affect of Window Size



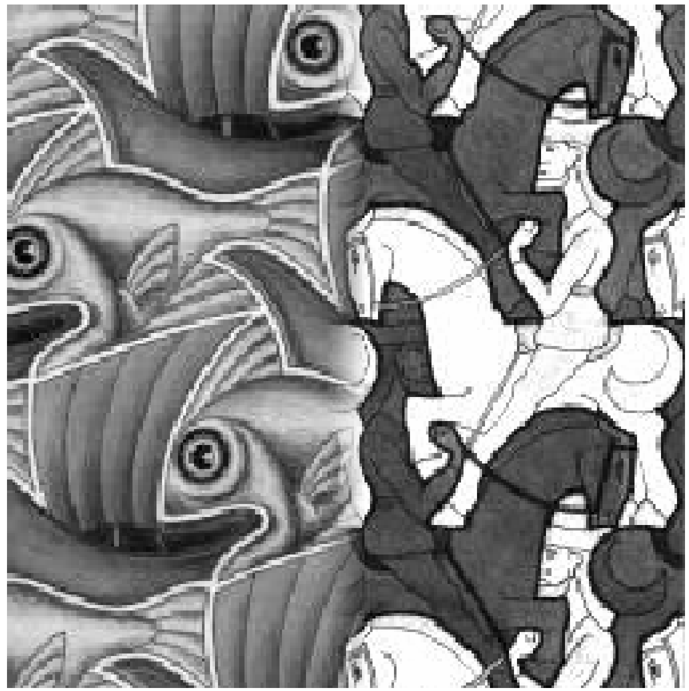


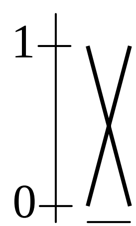
Affect of Window Size





Good Window Size





"Optimal" Window: smooth but not ghosted

What is the Optimal Window?

To avoid seams

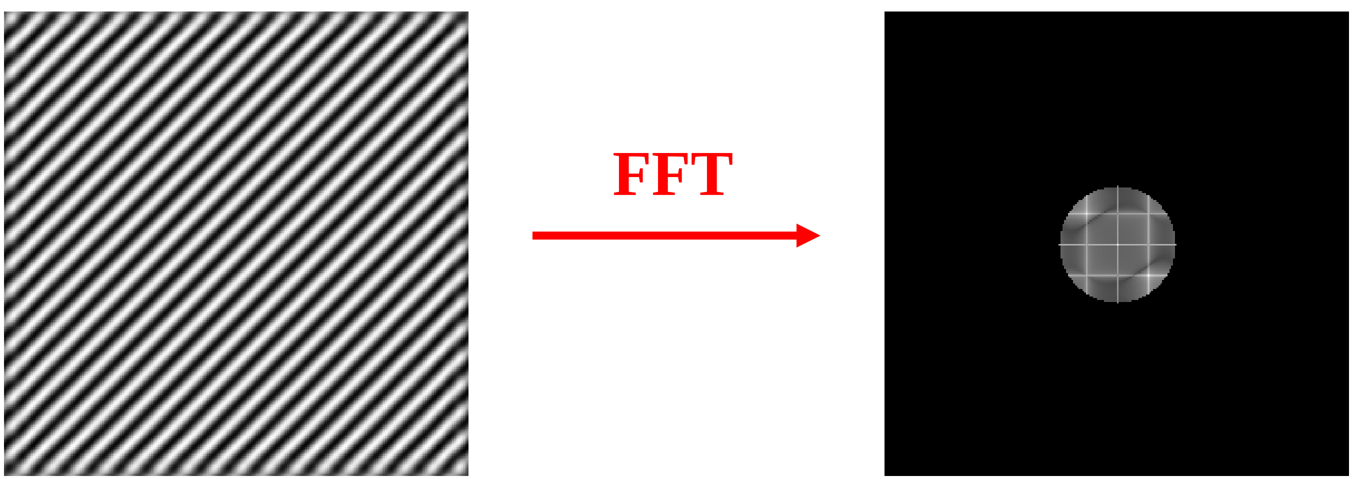
- window >= size of largest prominent feature

To avoid ghosting

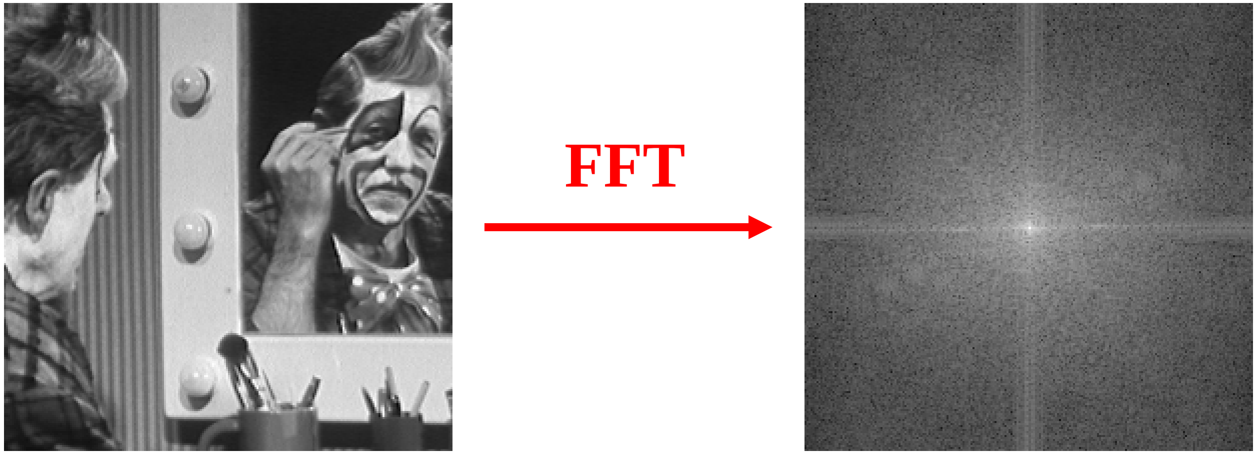
- window <= 2*size of smallest prominent feature

Natural to cast this in the *Fourier domain*

- largest frequency <= 2*size of smallest frequency
- image frequency content should occupy one “octave” (power of two)



What if the Frequency Spread is Wide



Idea (Burt and Adelson)

- Compute $F_{left} = FFT(I_{left})$, $F_{right} = FFT(I_{right})$
- Decompose Fourier image into octaves (bands)
  - $F_{left} = F_{left}^1 + F_{left}^2 + \ldots$
- Feather corresponding octaves F_{left}^i with F_{right}^i
  - Can compute inverse FFT and feather in spatial domain
- Sum feathered octave images in frequency domain

Better implemented in *spatial domain*

What does blurring take away?

original

What does blurring take away?

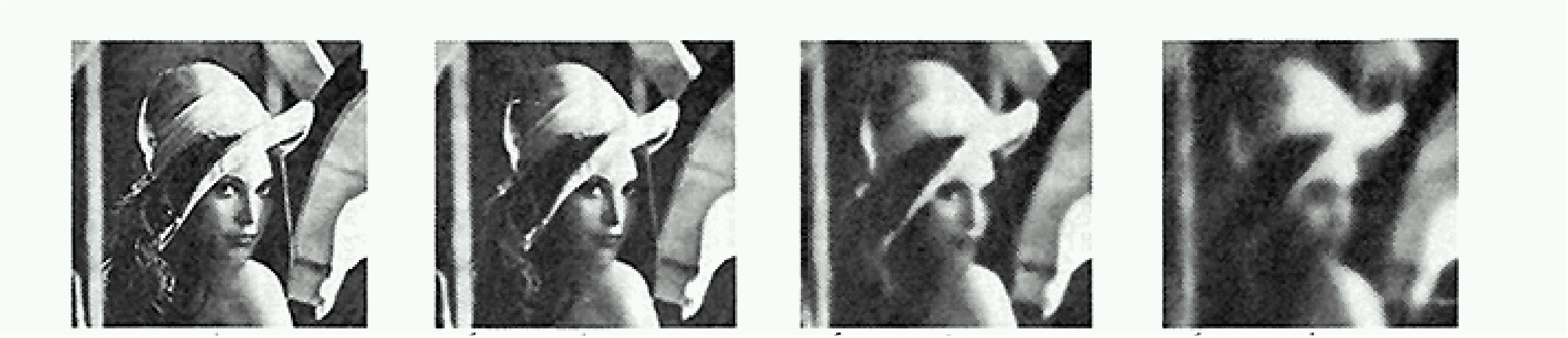
smoothed (5x5 Gaussian)

High-Pass filter

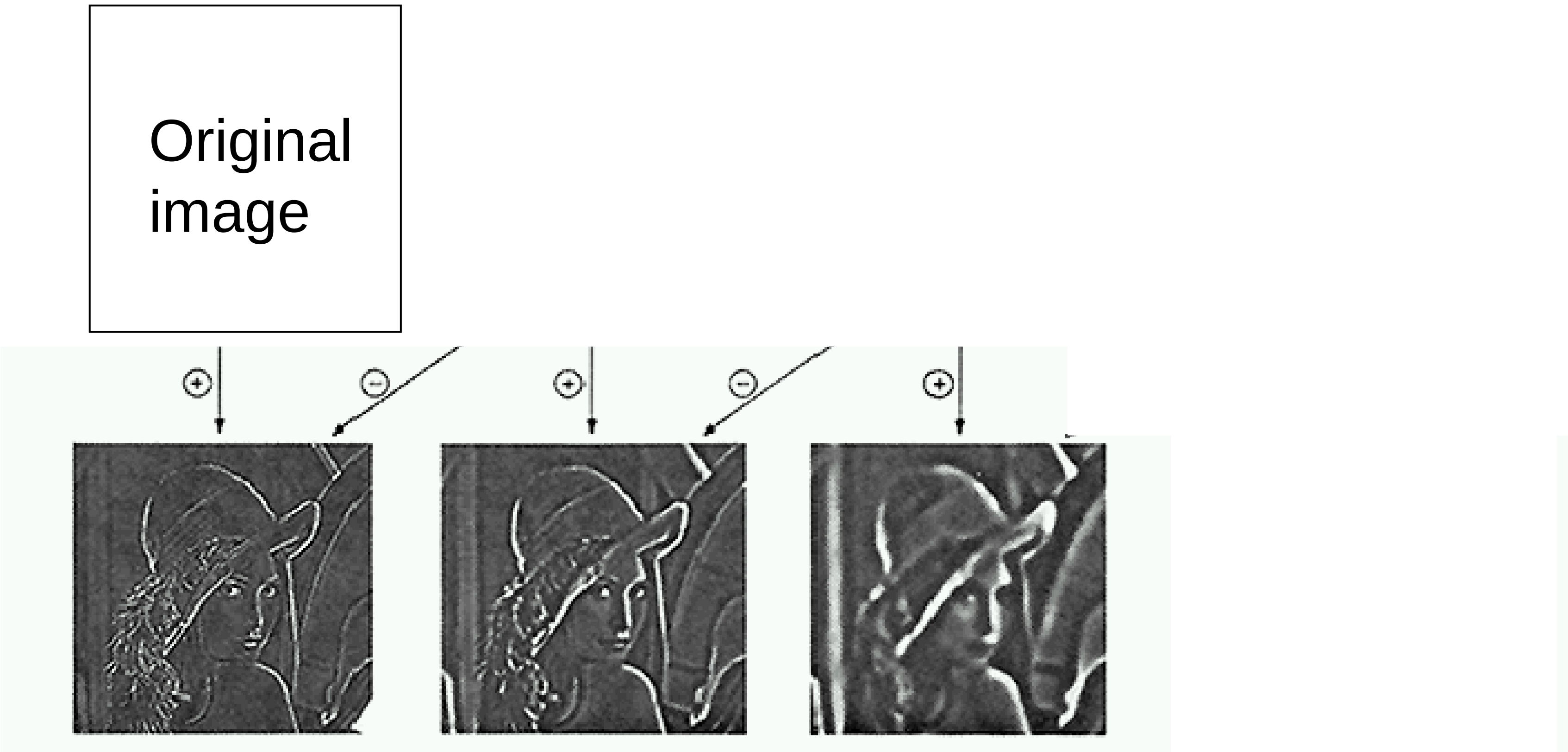
smoothed – original

Band-pass filtering

Gaussian Pyramid (low-pass images)

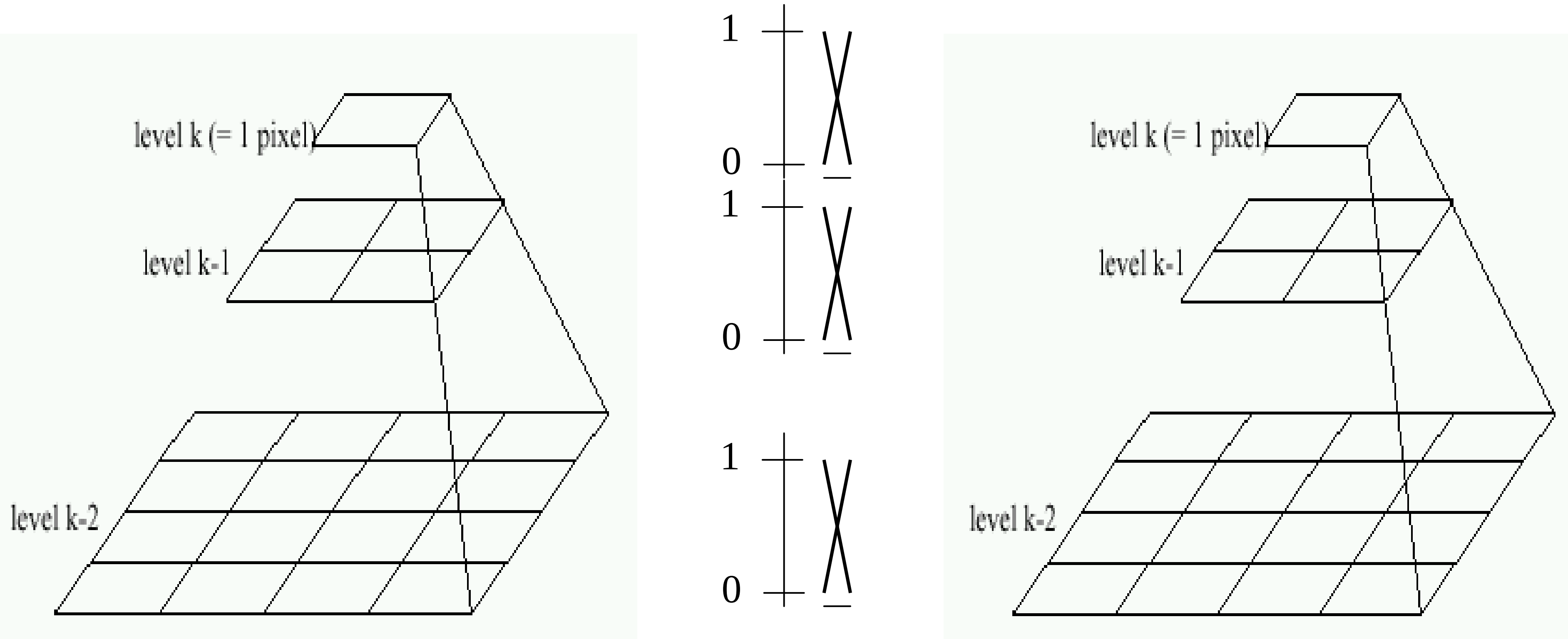


Laplacian Pyramid



How can we reconstruct (collapse) this pyramid into the original image?

Pyramid Blending

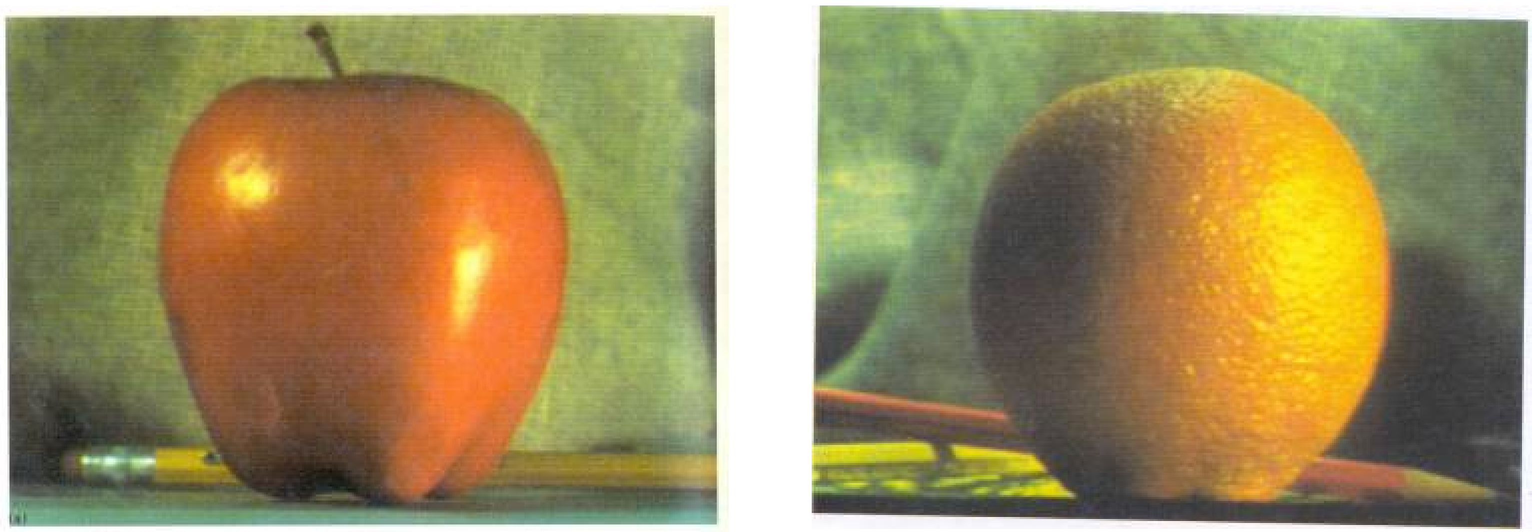


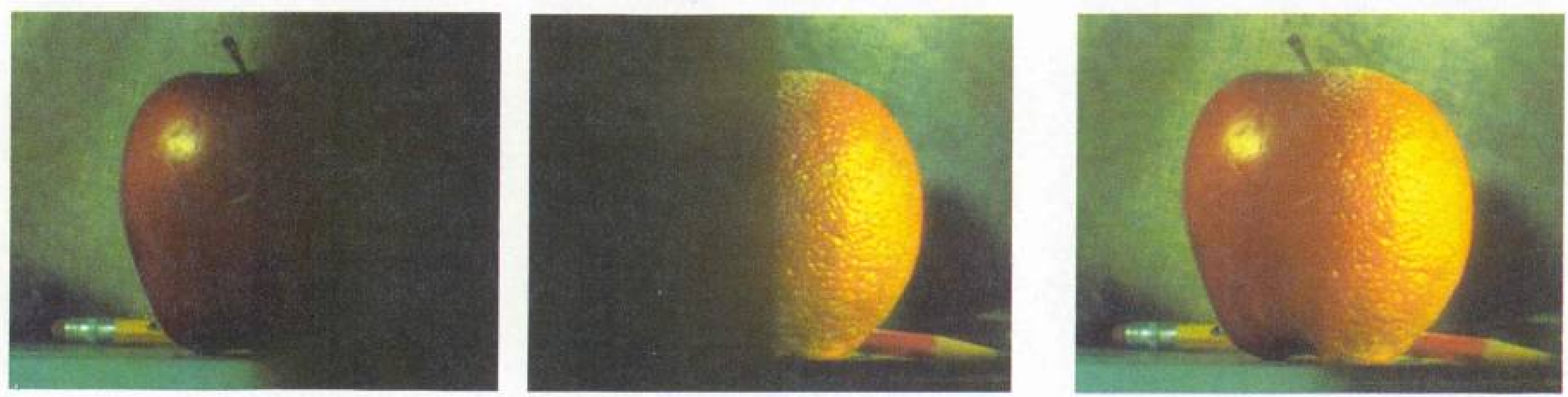
Left pyramid

blend

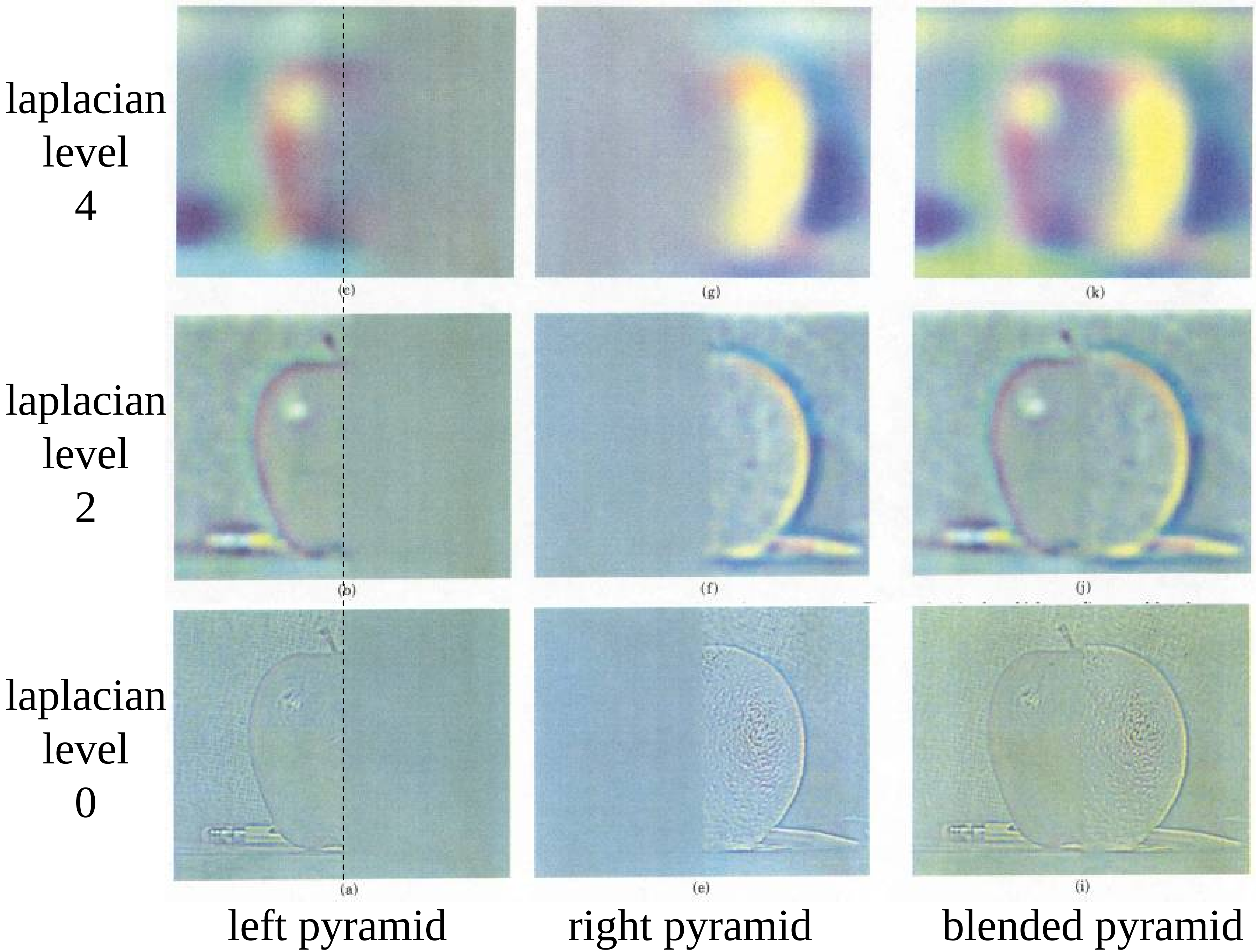
Right pyramid

Pyramid Blending





(d) (h) (l)



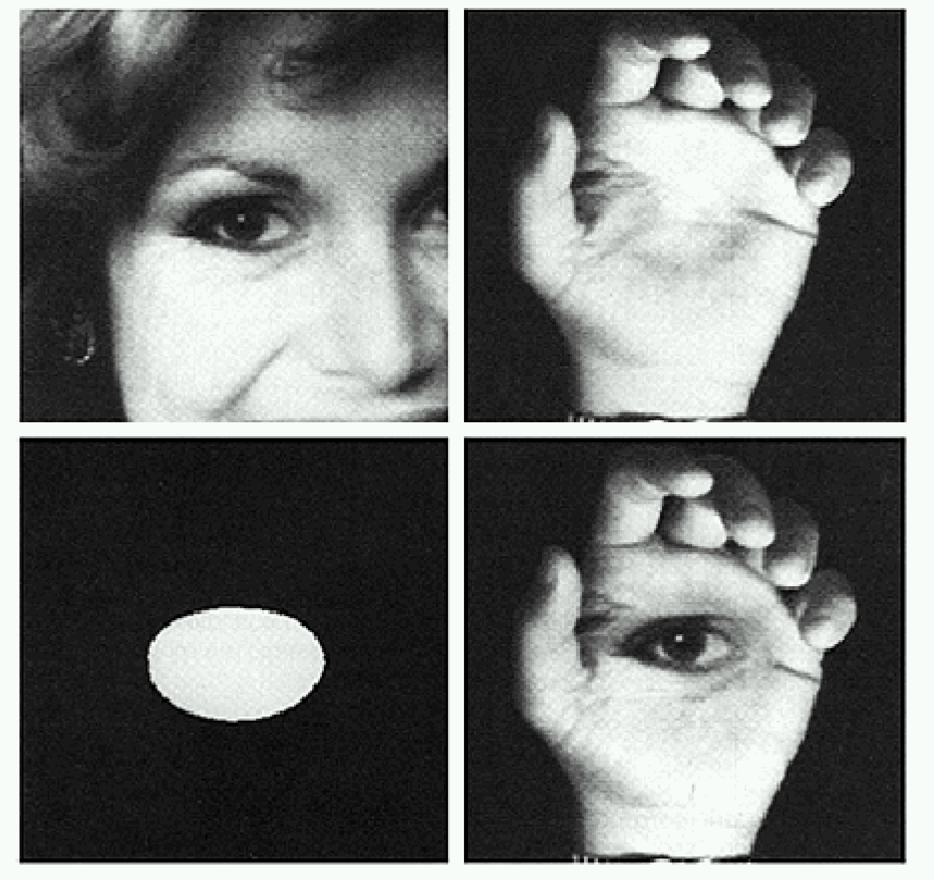
laplacian level 4
laplacian level 2
laplacian level 0
(c)
(g)
(k)
(b)
(f)
(j)
(a)
(e)
(i)
left pyramid
right pyramid
blended pyramid

Laplacian Pyramid: Blending

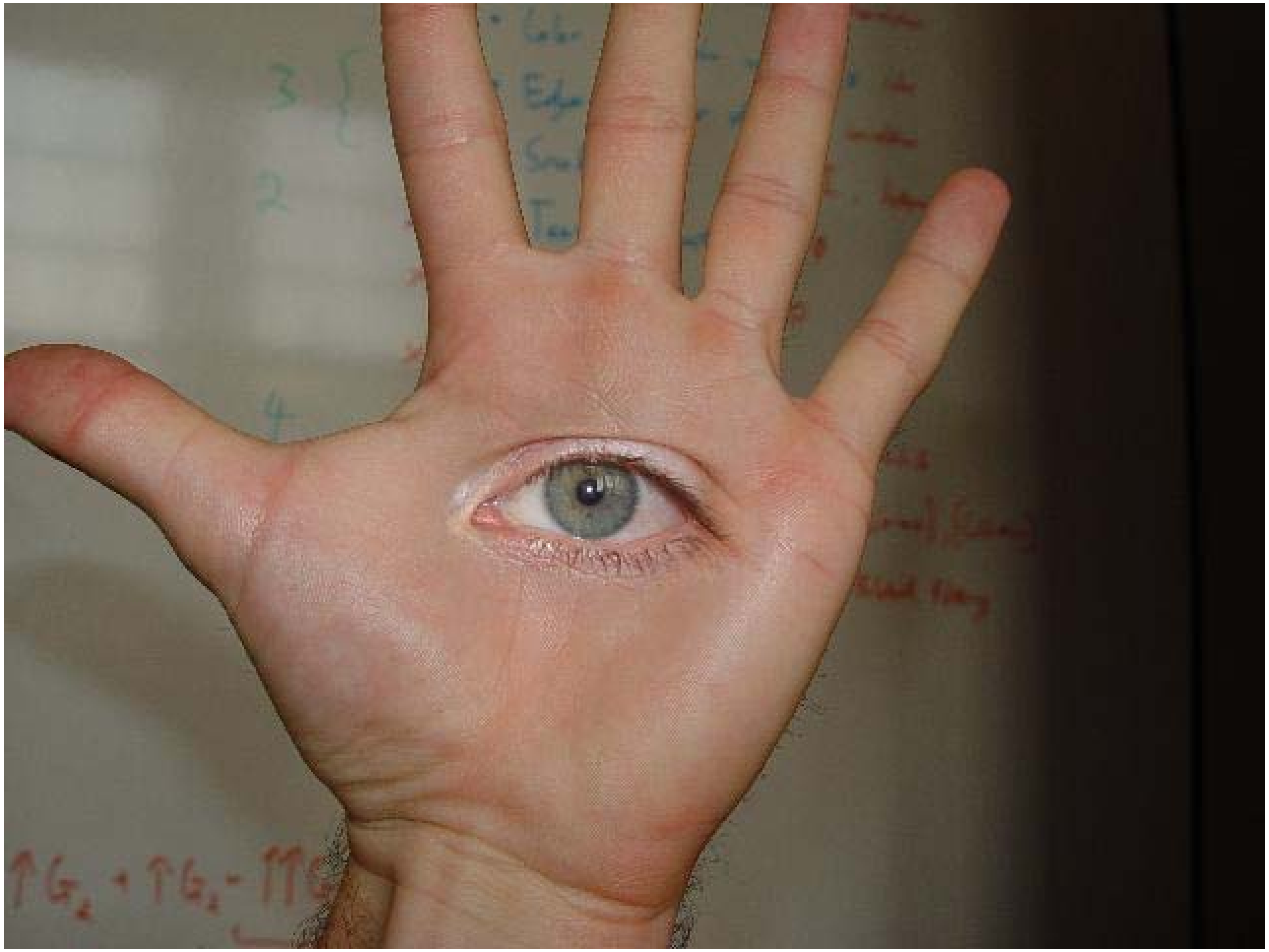
General Approach:

1. Build Laplacian pyramids *LA* and *LB* from images *A* and *B*
2. Build a Gaussian pyramid *GR* from selected region *R*
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

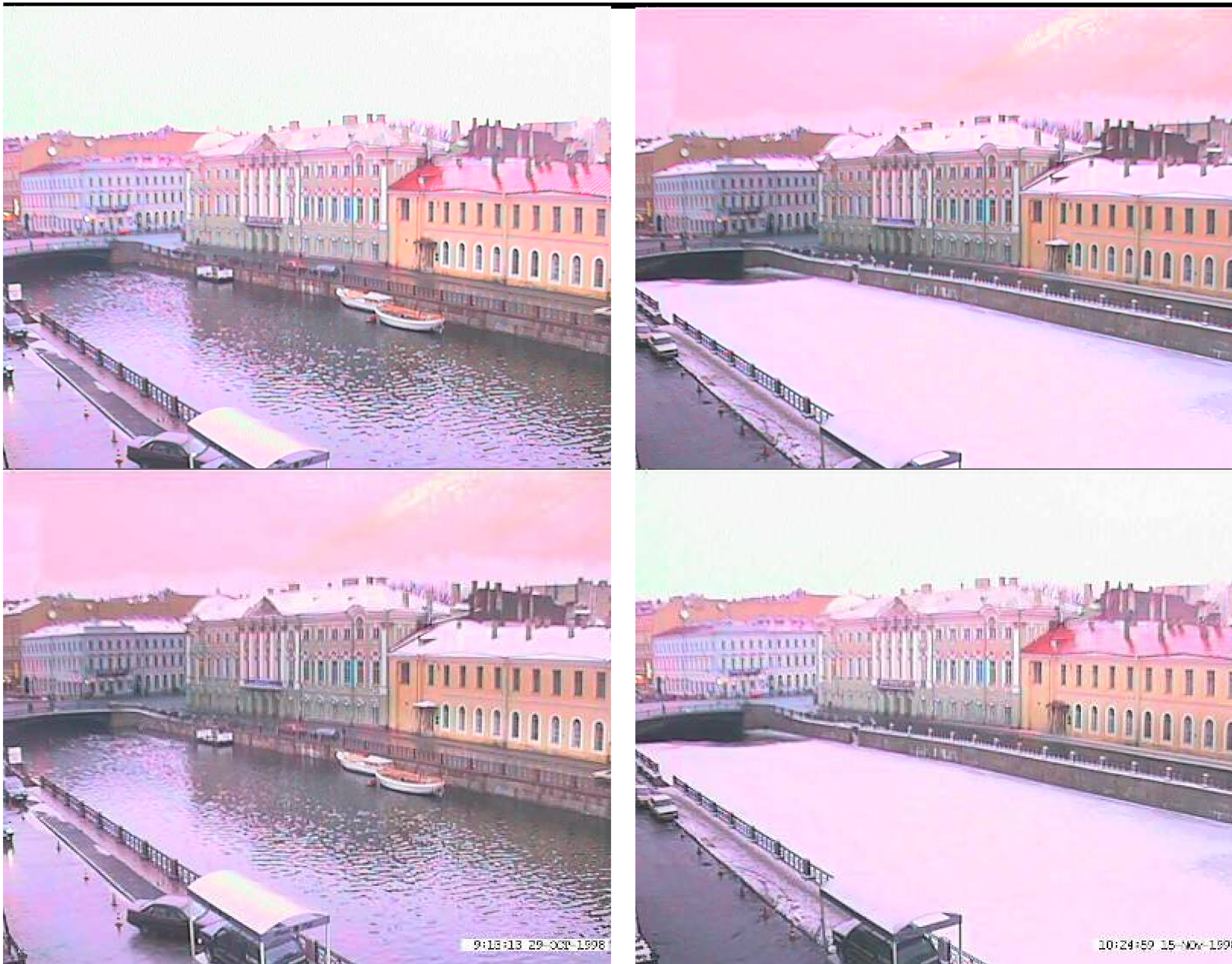


Horror Photo

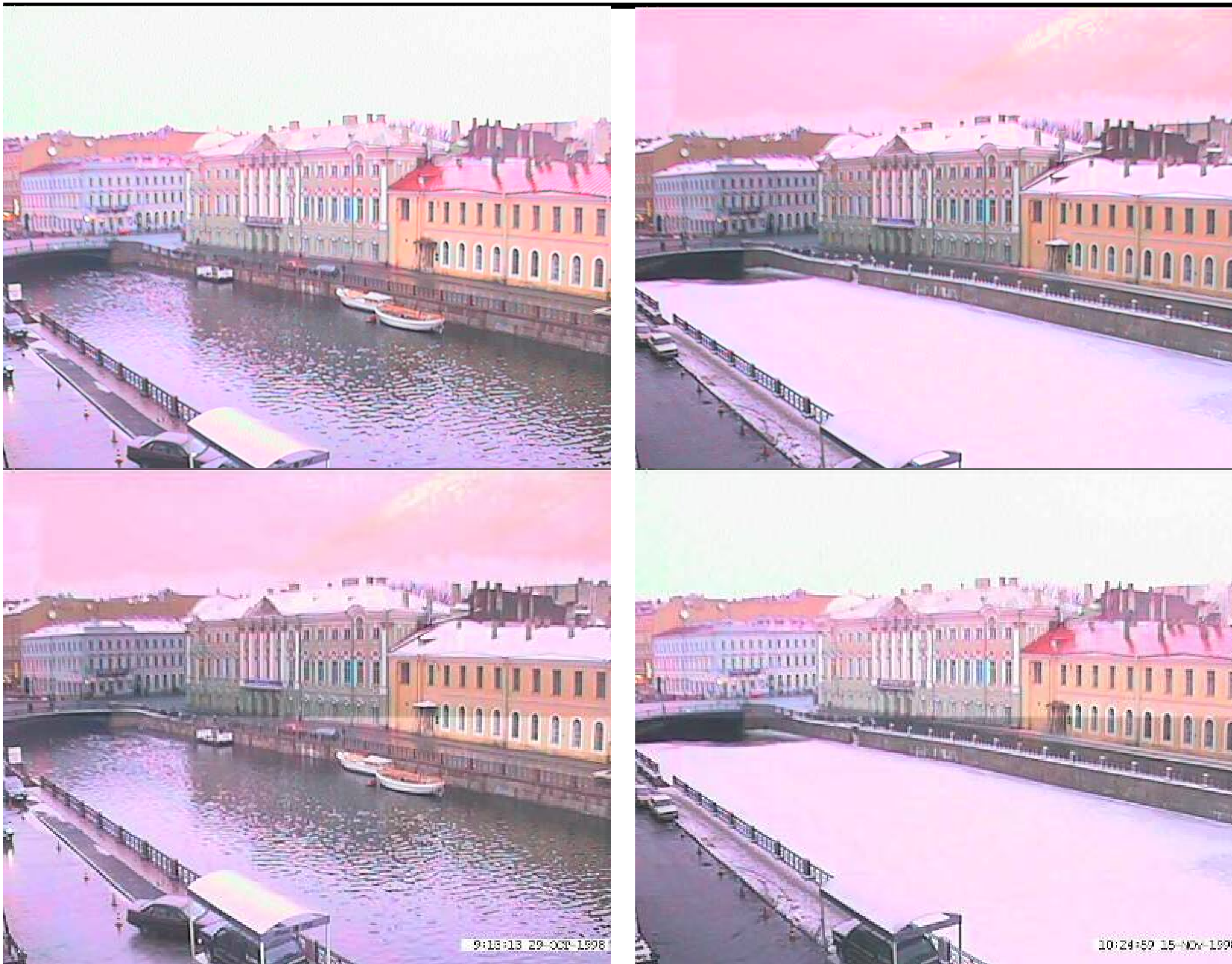


© prof. dmartin

Season Blending (St. Petersburg)



Season Blending (St. Petersburg)



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 mask

2-band Blending

Low frequency (λ > 2 pixels)

High frequency (λ < 2 pixels)

Linear Blending

2-band Blending

Gradient Domain

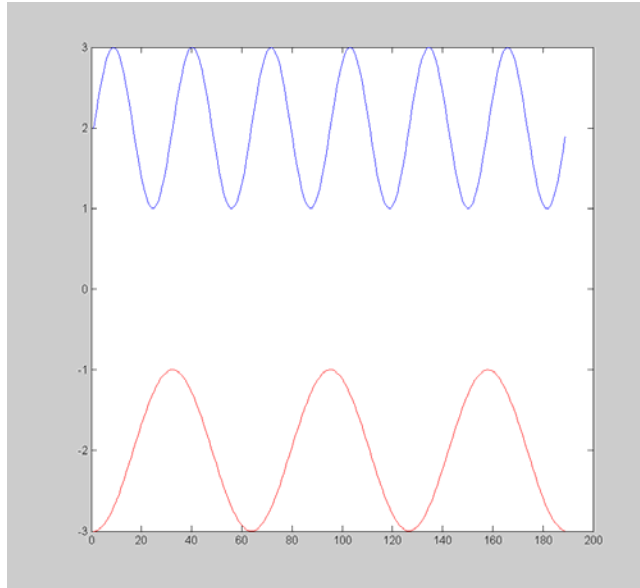
In Pyramid Blending, we decomposed our image into 2^{nd} derivatives (Laplacian) and a low-res image

Let us now look at 1^{st} derivatives (gradients):

- No need for low-res image
  - captures everything (up to a constant)
- Idea:
  - Differentiate
  - Blend
  - Reintegrate

Gradient Domain blending (1D)

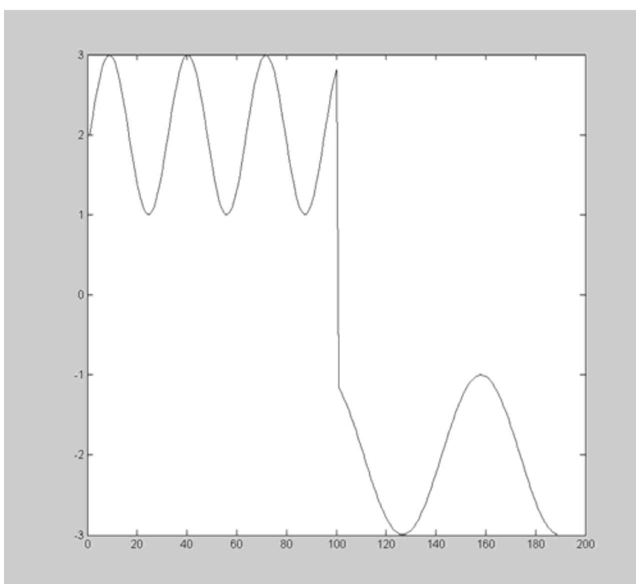
Two signals

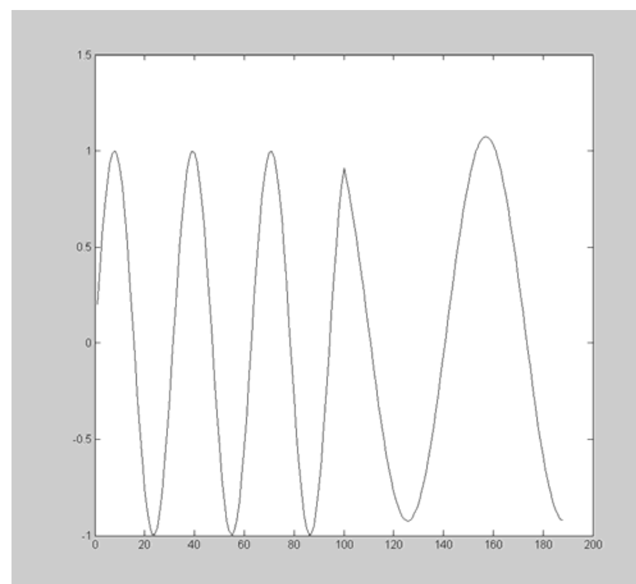


bright

dark

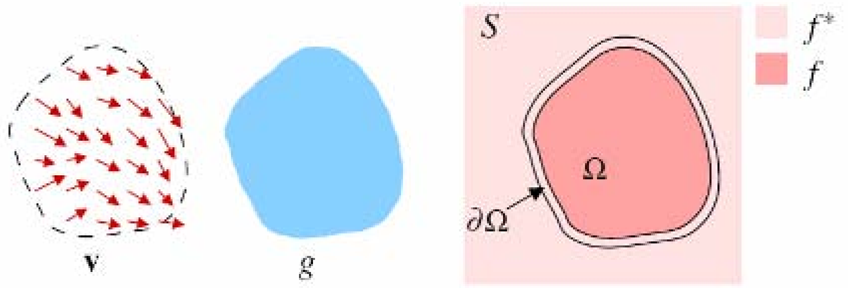
Regular blending





Blending derivatives

Gradient Domain Blending (2D)



Trickier in 2D:

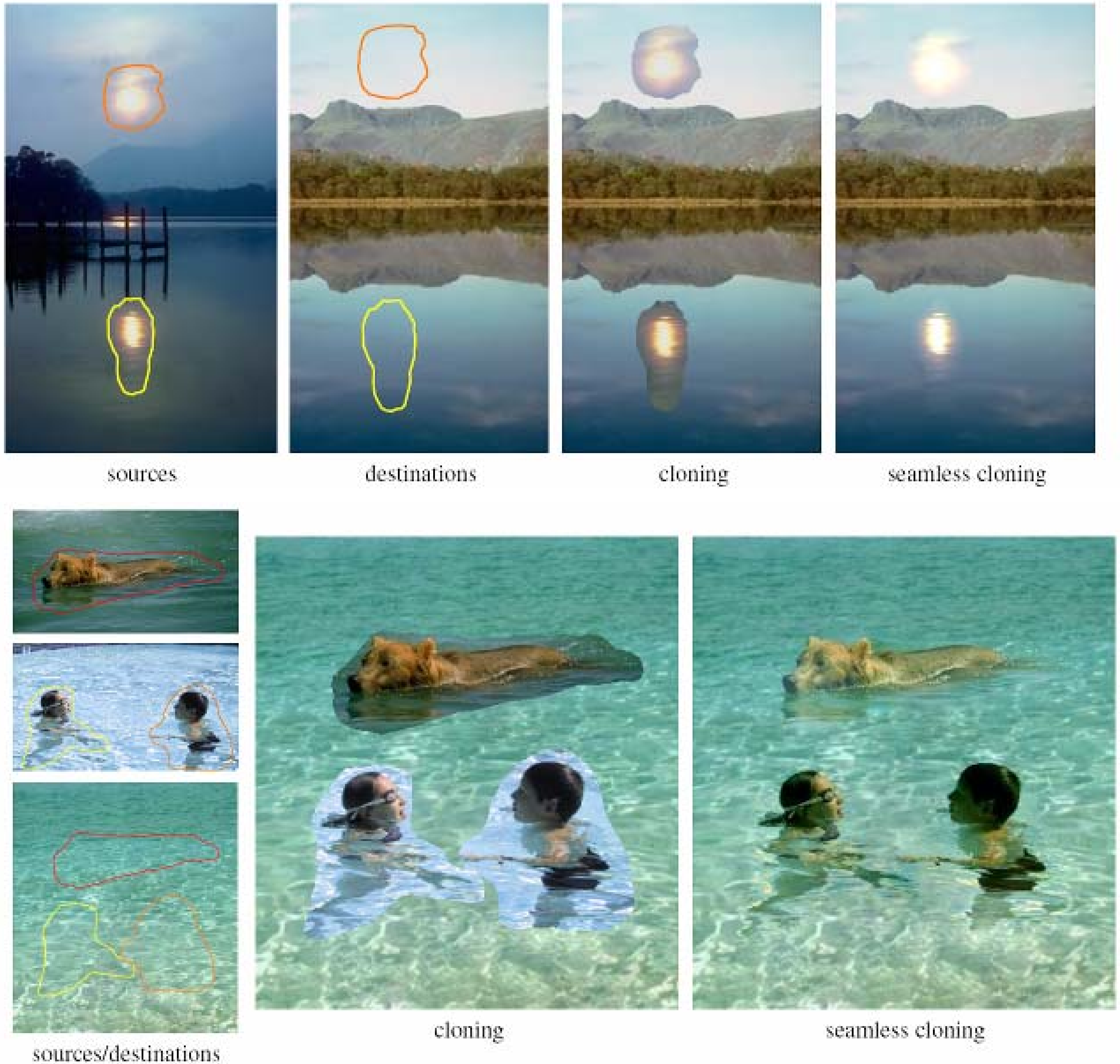
- Take partial derivatives dx and dy (the gradient field)
- Fidle around with them (smooth, blend, feather, etc)
- Reintegrate
  - But now integral(dx) might not equal integral(dy)
- Find the most agreeable solution
  - Equivalent to solving Poisson equation
  - Can use FFT, deconvolution, multigrid solvers, etc.

Comparisons: Levin et al, 2004

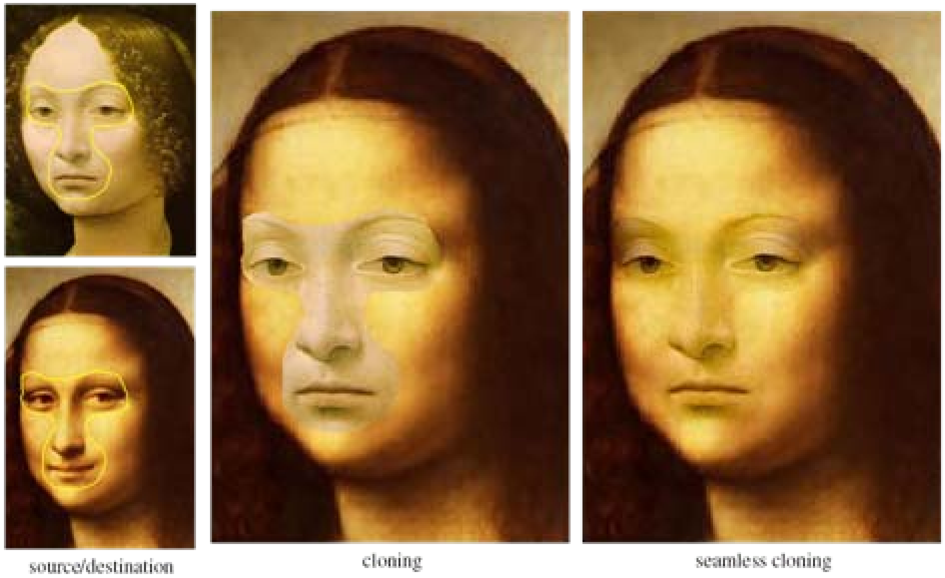
Pyramid blending — Feathering

Pyramid blending on the gradients — GIST1

Perez et al., 2003



Perez et al, 2003



editing

Limitations:

- Can’t do contrast reversal (gray on black -> gray on white)
- Colored backgrounds “bleed through”
- Images need to be very well aligned

Don't blend, CUT!

Moving objects become ghosts

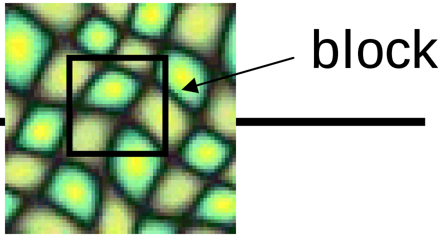
So far we only tried to blend between two images.
What about finding an optimal seam?

Davis, 1998

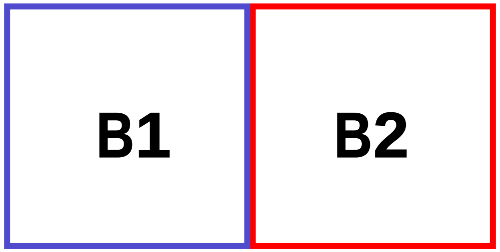
Segment the mosaic

- Single source image per segment
- Avoid artifacts along boundries
  - Dijkstra's algorithm

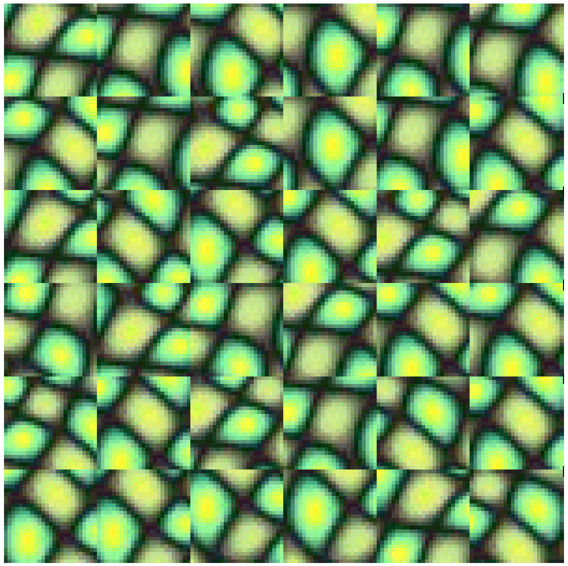
Efros & Freeman, 2001

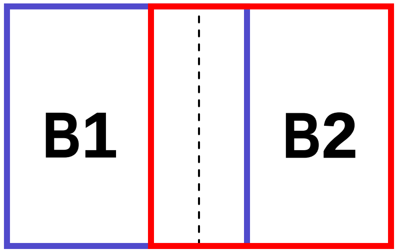


Input texture

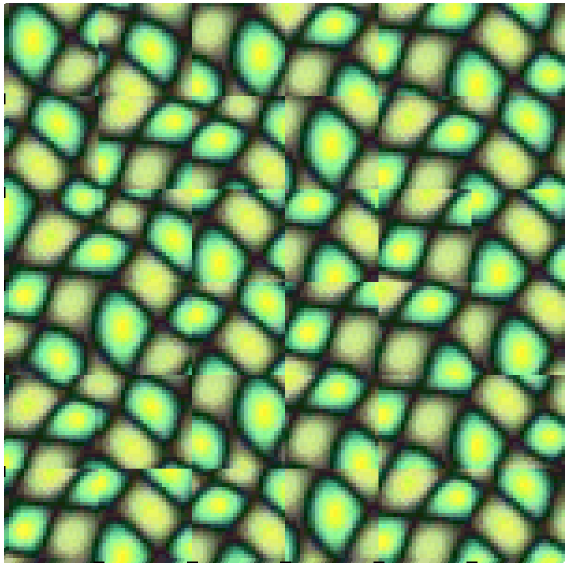


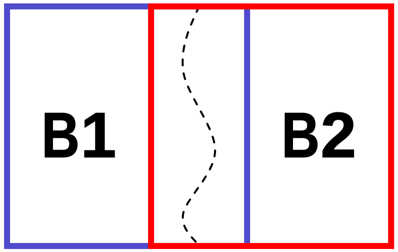
Random placement of blocks



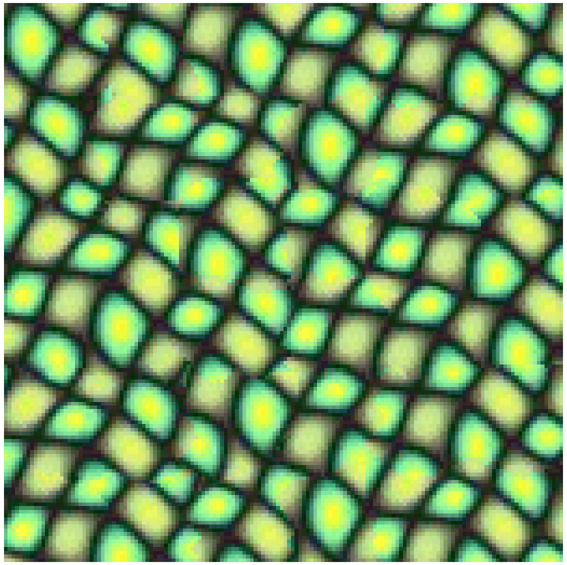


Neighboring blocks constrained by overlap

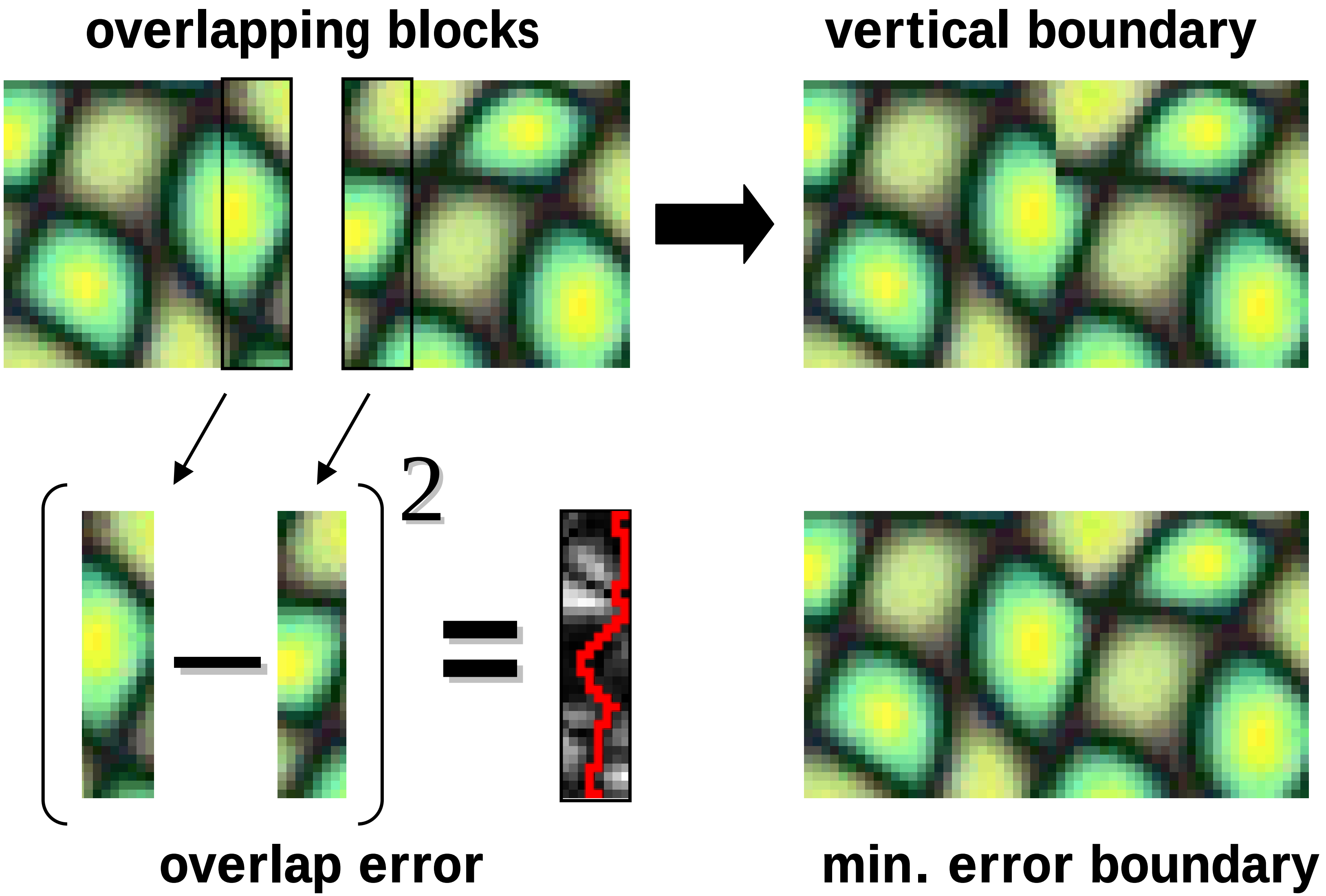




Minimal error boundary cut



Minimal error boundary



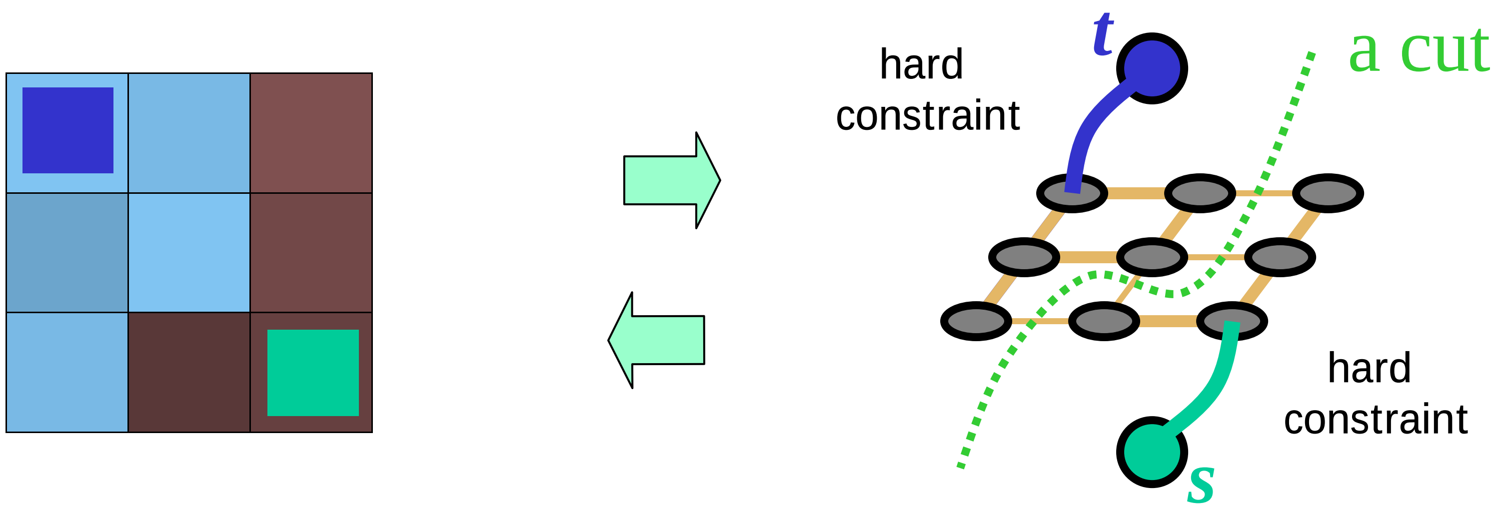
Graphcuts

What if we want similar “cut-where-things-agree” idea, but for closed regions?

- Dynamic programming can’t handle loops

Graph cuts

(simple example à la Boykov&Jolly, ICCV'01)



Minimum cost cut can be computed in polynomial time

(max-flow/min-cut algorithms)

Kwatra et al, 2003

Actually, for this example, DP will work just as well…

Lazy Snapping (Li el al., 2004)

Interactive segmentation using graphcuts

Putting it all together

Compositing images

- Have a clever blending function
  - Feathering
  - blend different frequencies differently
  - Gradient based blending
- Choose the right pixels from each image
  - Dynamic programming – optimal seams
  - Graph-cuts

Now, let’s put it all together:

- Interactive Digital Photomontage, 2004 (video)

Interactive Digital Photomontage

Aseem Agarwala, Mira Dontcheva
Maneesh Agrawala, Steven Drucker, Alex Colburn
Brian Curless, David Salesin, Michael Cohen