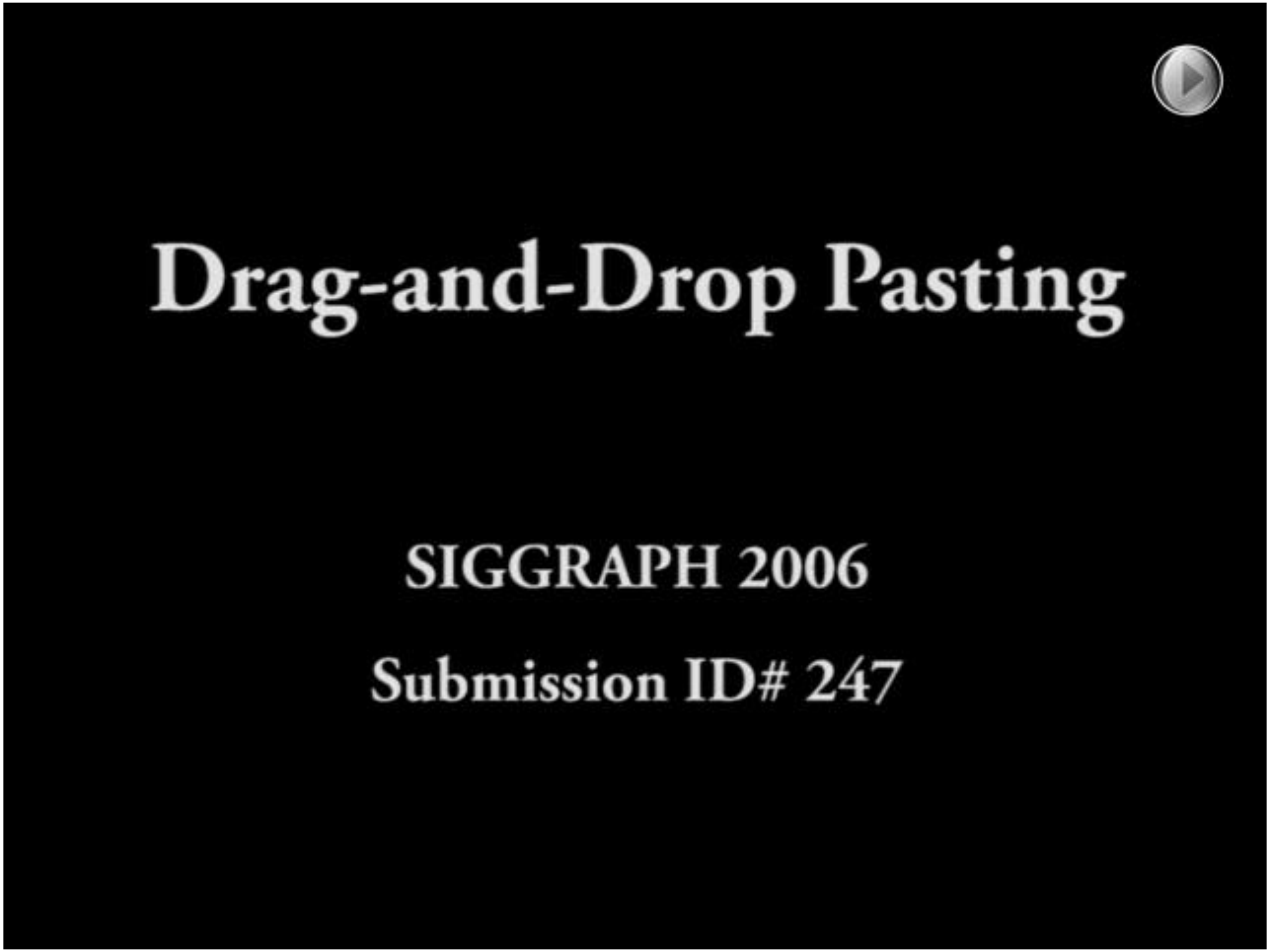

Drag-and-Drop Pasting
SIGGRAPH 2006
Submission ID# 247

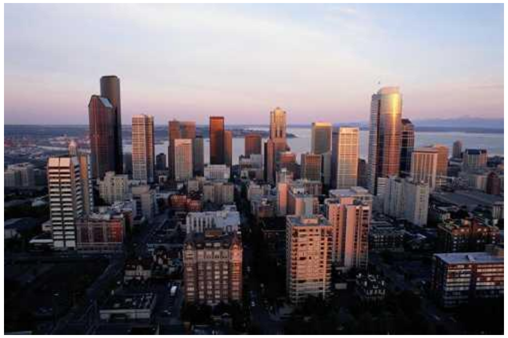
Image Blending + Image Carving

© Kenneth Kwan

Slides Modified from Alexei Efros, CMU,

Compositing Procedure

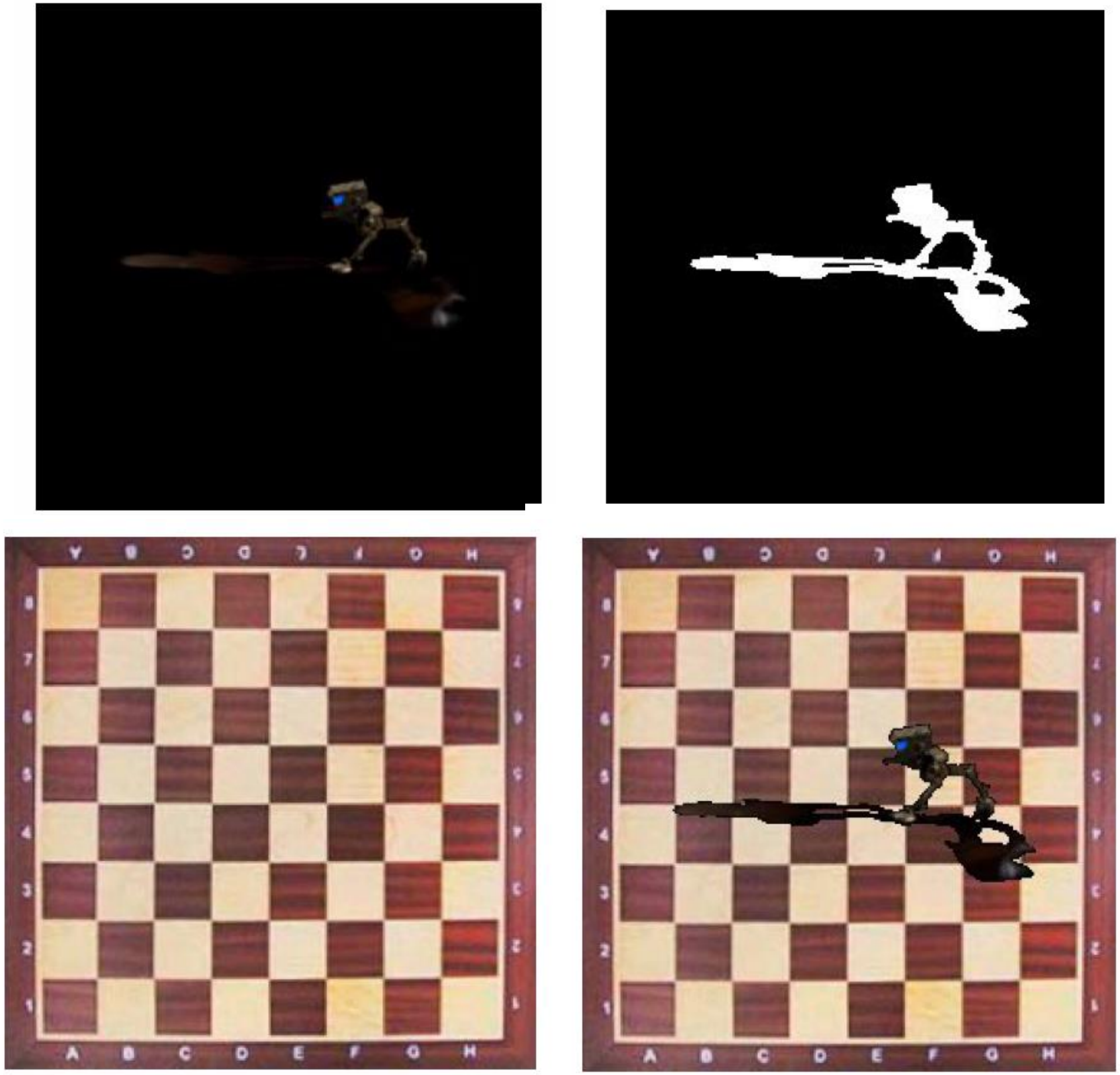
1. Extract Sprites (e.g using *Intelligent Scissors* in Photoshop)



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

Composite by
David Dewey

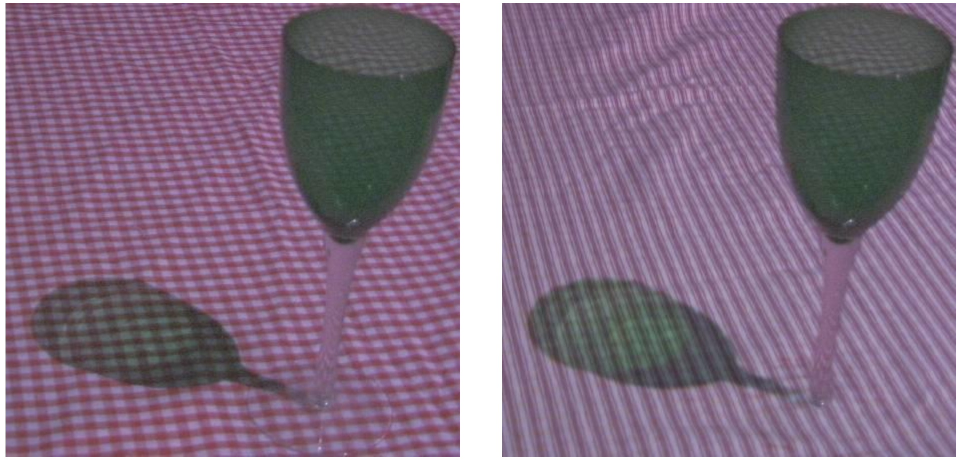
Just replacing pixels rarely works



Binary mask

Problems: boundries & transparency (shadows)

Two Problems:



Semi-transparent objects

Pixels too large

Solution: alpha channel

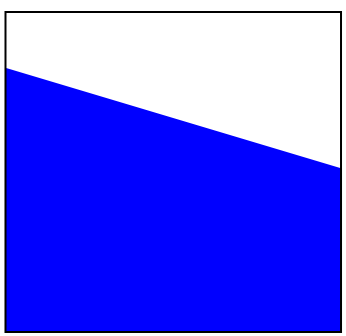
Add one more channel:

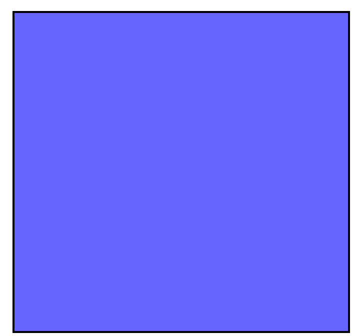
- Image(R,G,B,alpha)

Encodes transparency (or pixel coverage):

- Alpha = 1: opaque object (complete coverage)
- Alpha = 0: transparent object (no coverage)
- 0<Alpha<1: semi-transparent (partial coverage)

Example: alpha = 0.3



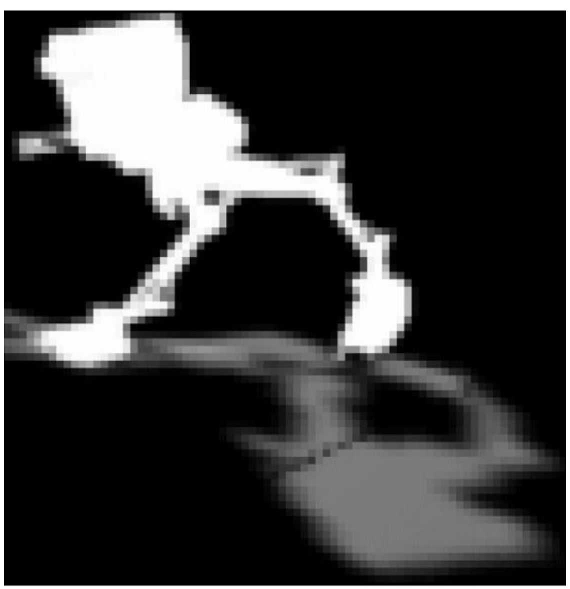


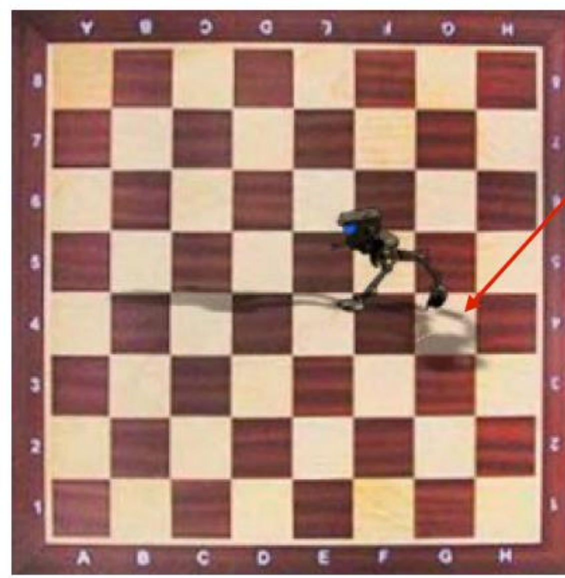
Partial coverage or semi-transparency

Alpha Blending

$$I_{comp} = \alpha I_{fg} + (1-\alpha) I_{bg}$$

alpha mask





shadow


The Woodlands
SPEEDS 25 MPH 45 MPH
Spring
Kingwood
Humble
Jersey Village
Katy
Houston
Bellaire
Pasadena
Sugar Land
6:26 AM
6:24:01
6:25 56°
6:00
7:00
TEXAS WEATHER: DALLAS/FT. WORTH
30% AM SHOWERS, 66
11 NEWS
engadget

Multiple Alpha Blending

So far we assumed that one image (background) is opaque.

If blending semi-transparent sprites (the "A over B" operation):

$$I_{comp} = \alpha_a I_a + (1-\alpha_a)\alpha_b I_b$$

$$\alpha_{comp} = \alpha_a + (1-\alpha_a)\alpha_b$$

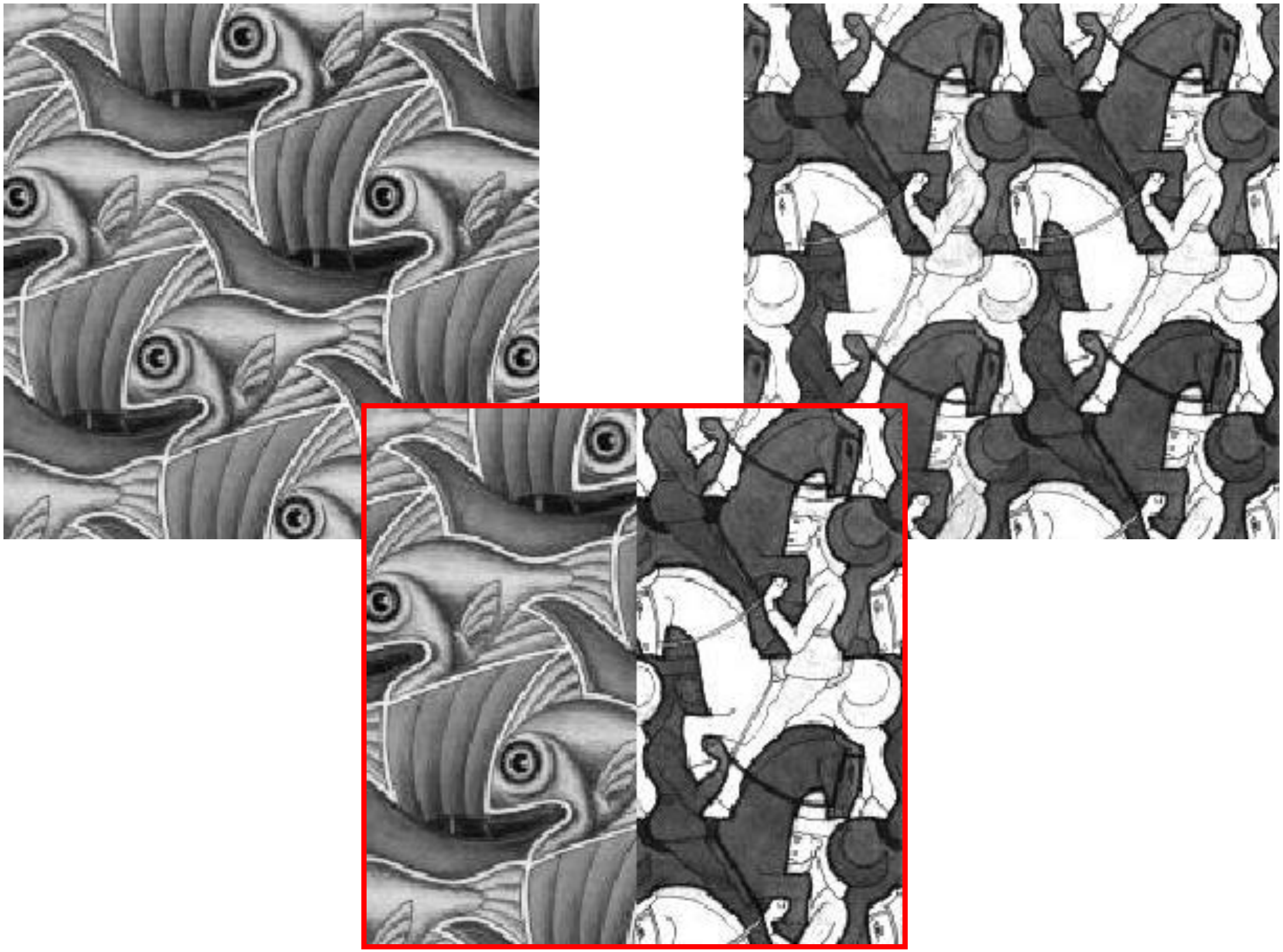
Note: sometimes alpha is premultiplied: im(αR,αG,αB,α):

$$I_{comp} = I_a + (1-\alpha_a)I_b$$

(same for alpha!)

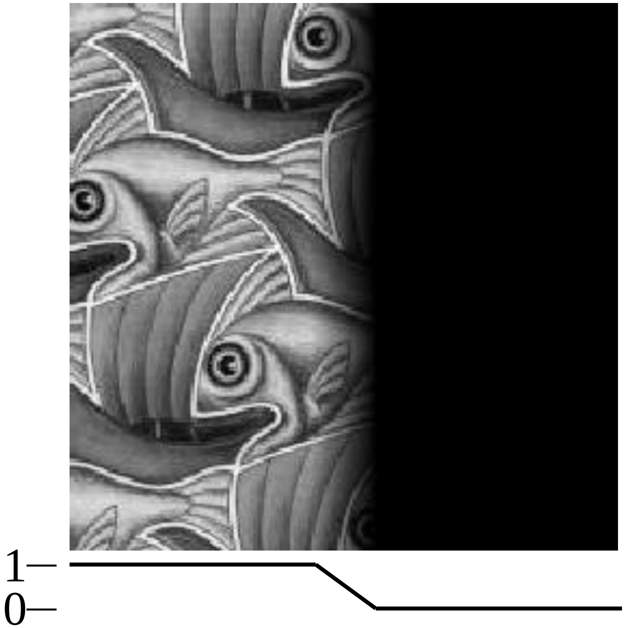

Flash Matting
Siggraph 2006
Submission ID #220

Alpha Hacking…

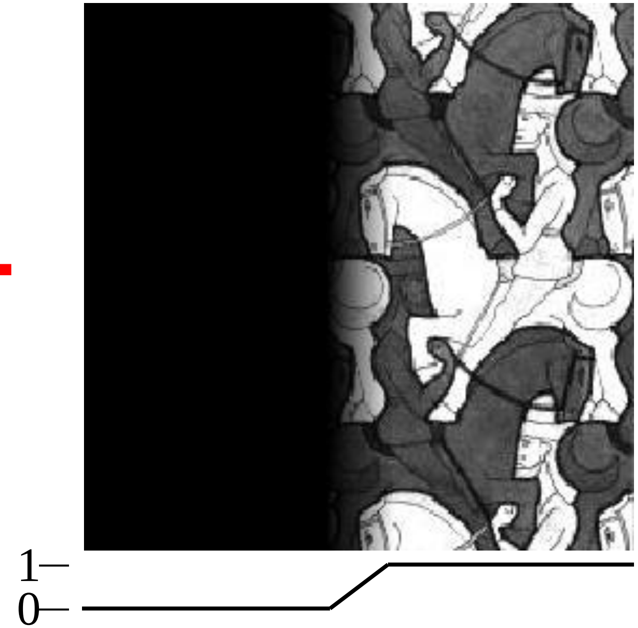


No physical interpretation, but it smoothes the seams

Feathering



+



=

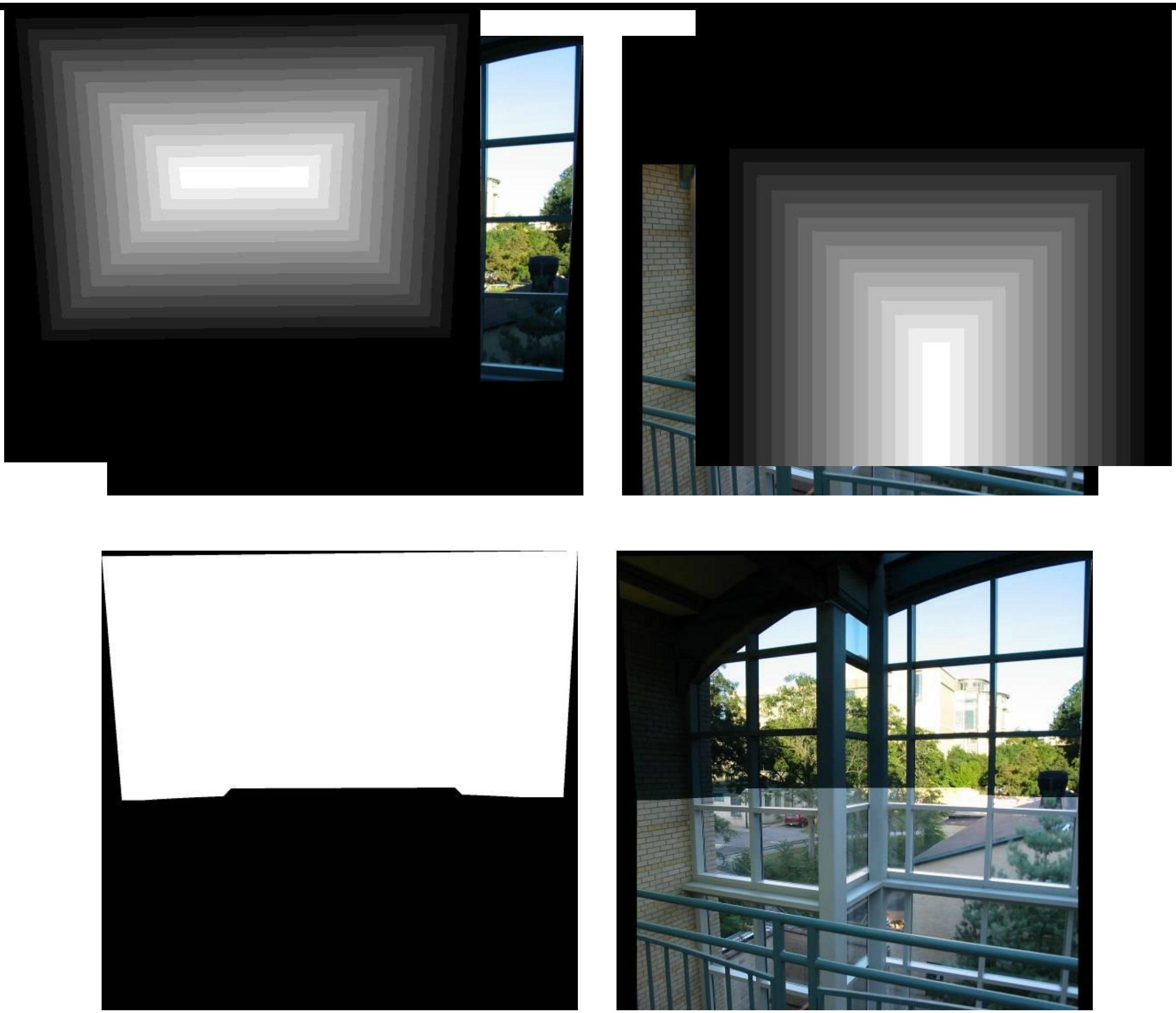
Encoding as transparency

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

Setting alpha: simple averaging

Alpha = .5 in overlap region

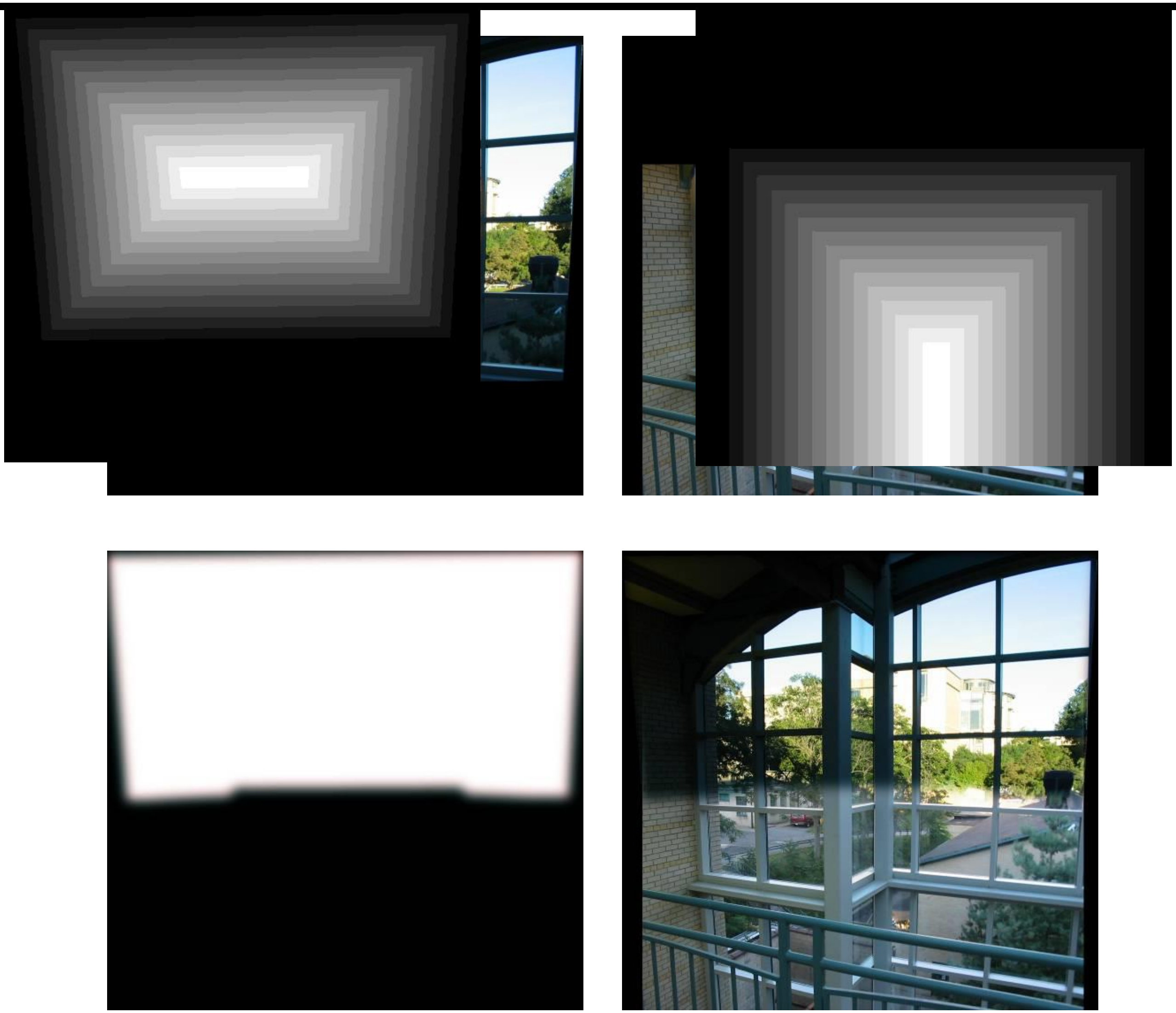
Setting alpha: center seam



Distance transform

Alpha = logical(dtrans1>dtrans2)

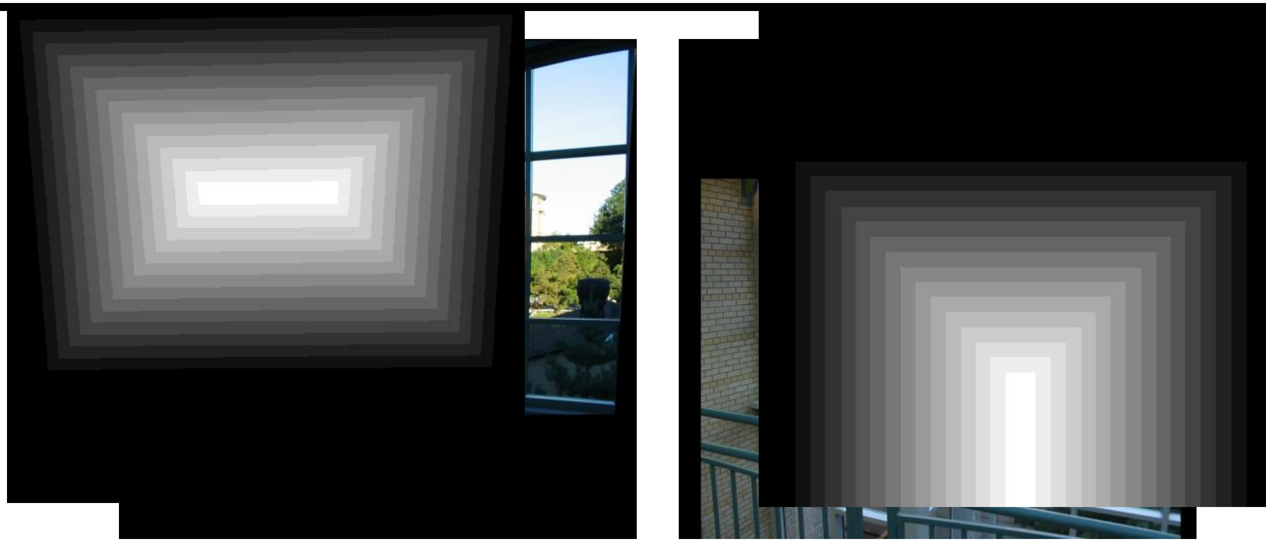
Setting alpha: blurred seam



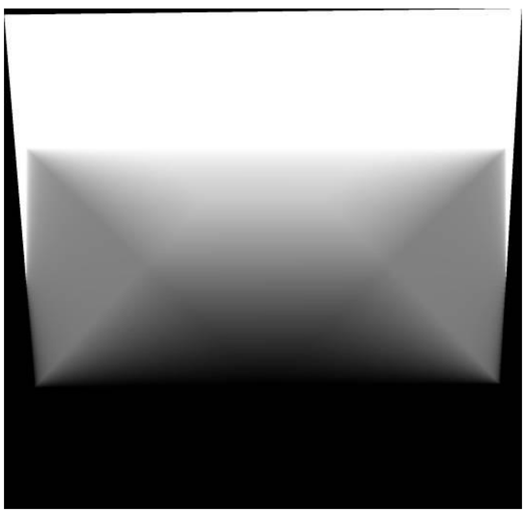
Distance transform

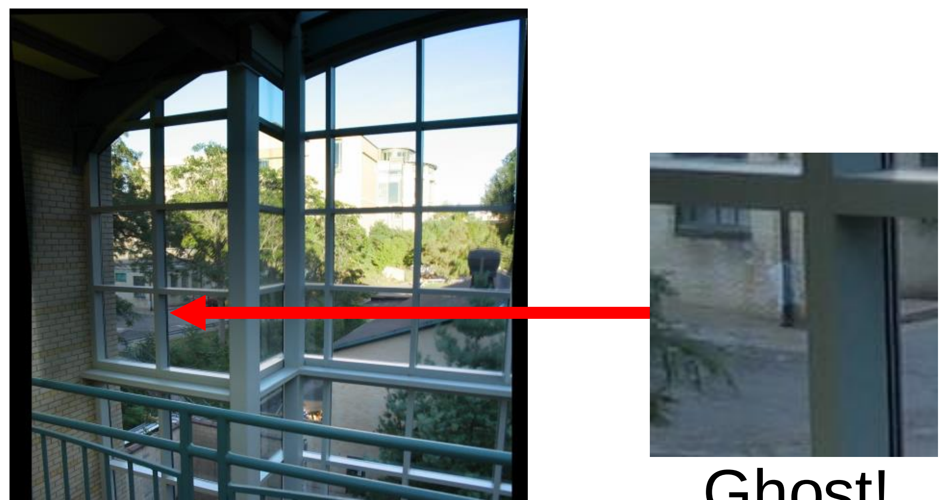
Alpha = blurred

Setting alpha: center weighting



Distance transform

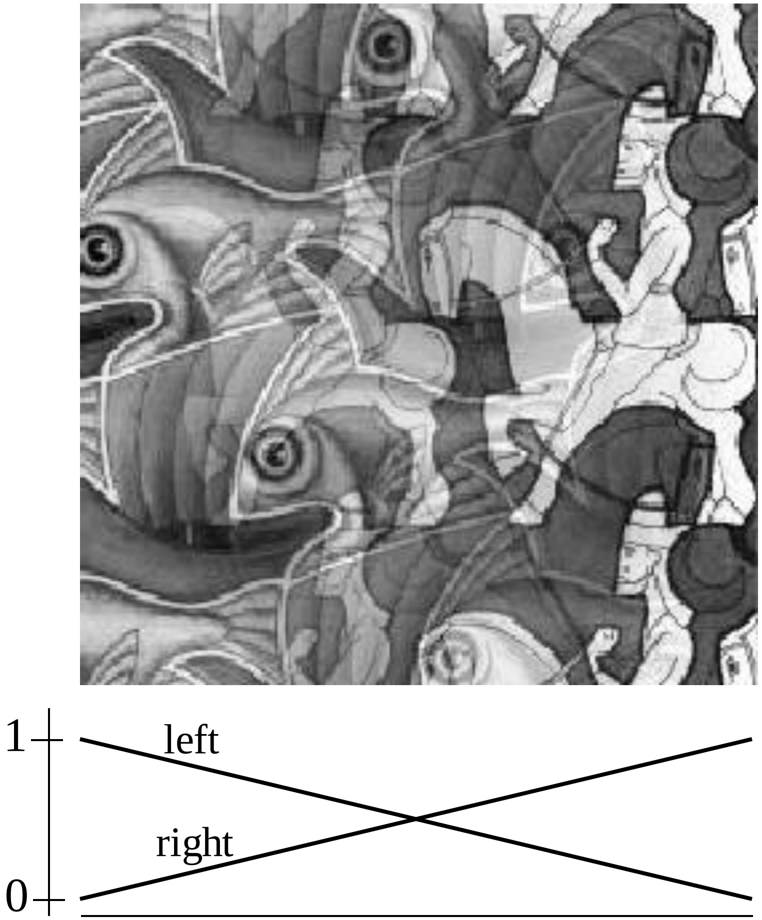


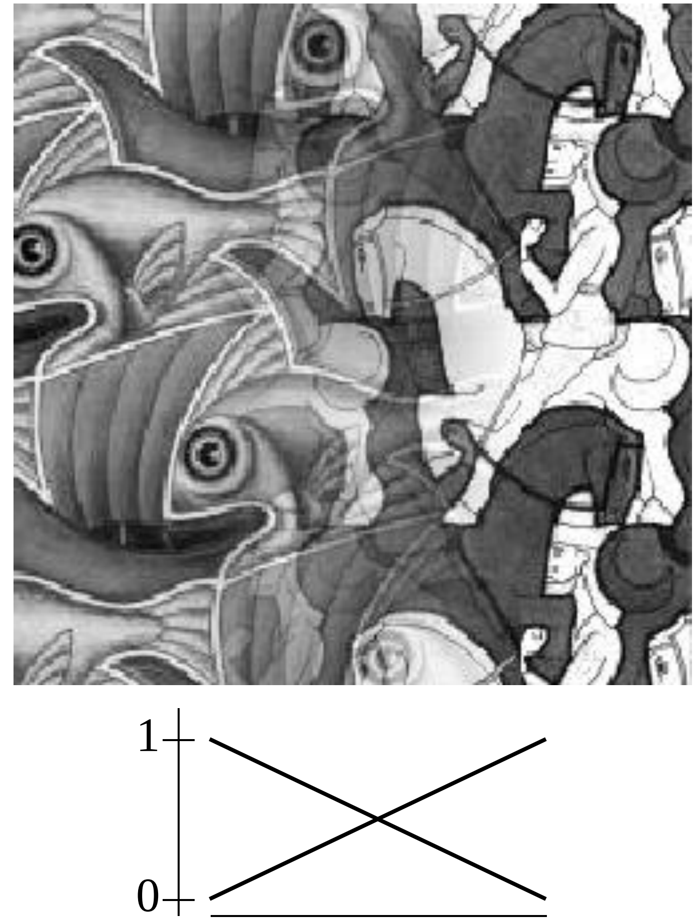


Ghost!

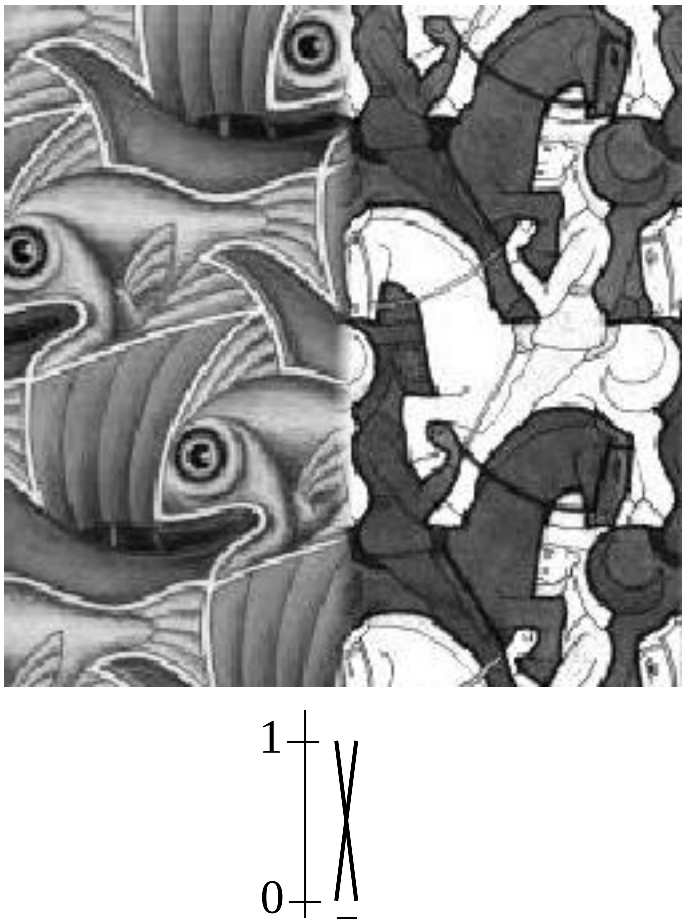
Alpha = dtrans1 / (dtrans1+dtrans2)

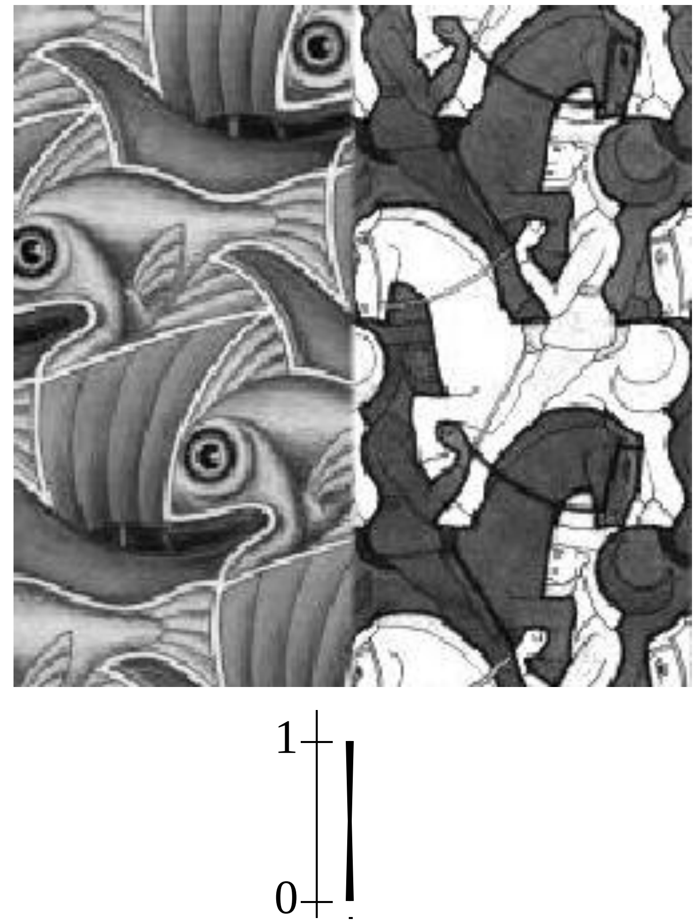
Affect of Window Size



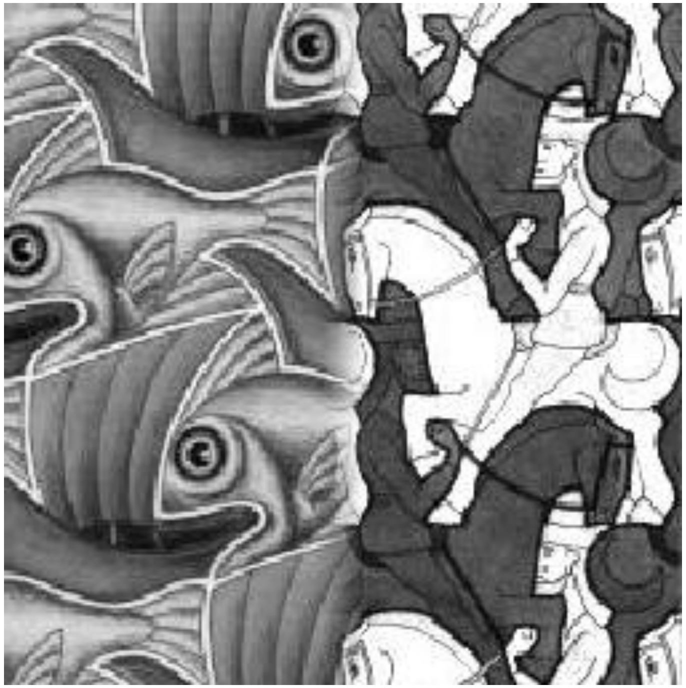


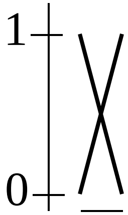
Affect of Window Size





Good Window Size





“Optimal” Window: smooth but not ghosted

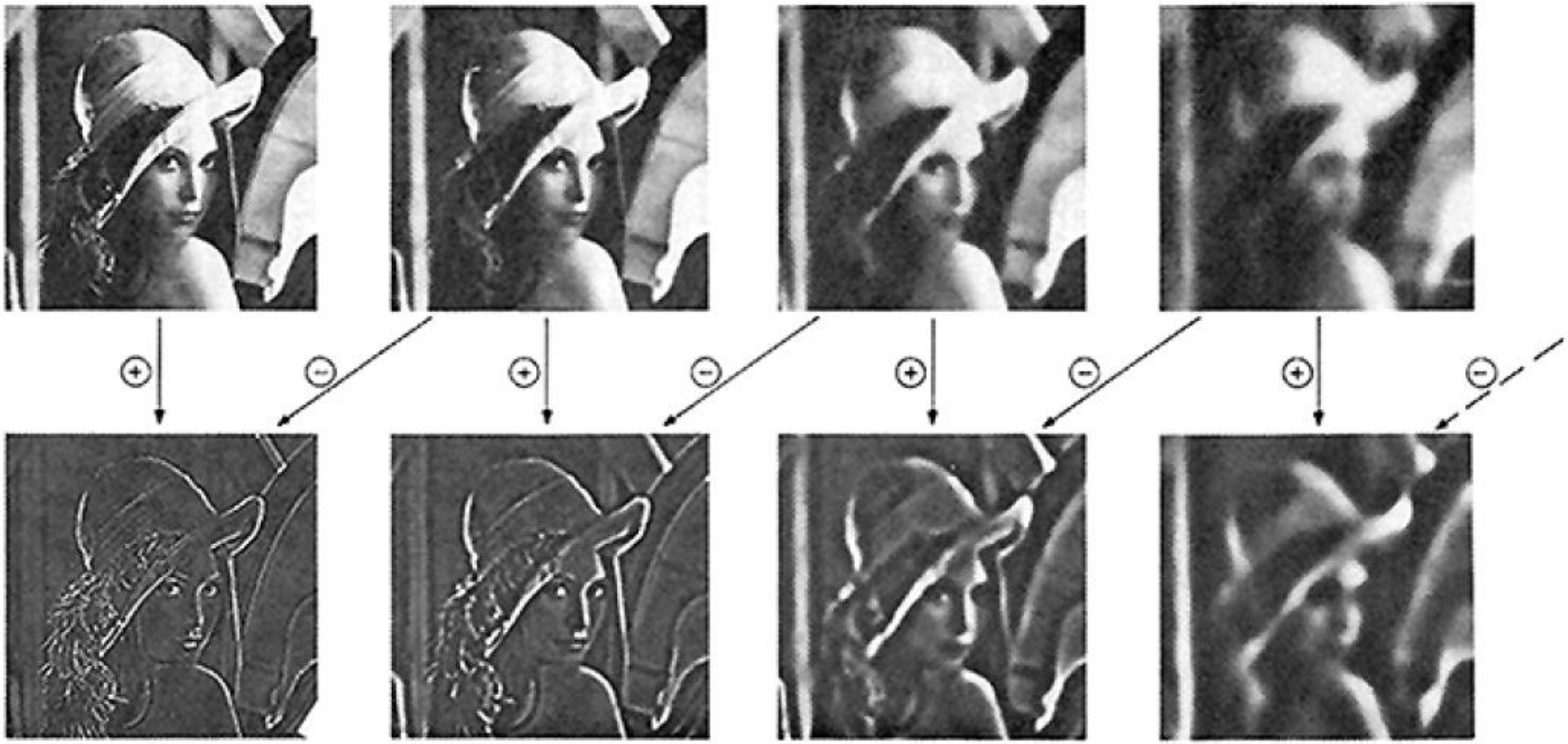
What is the Optimal Window?

To avoid seams

- window >= size of largest prominent feature

To avoid ghosting

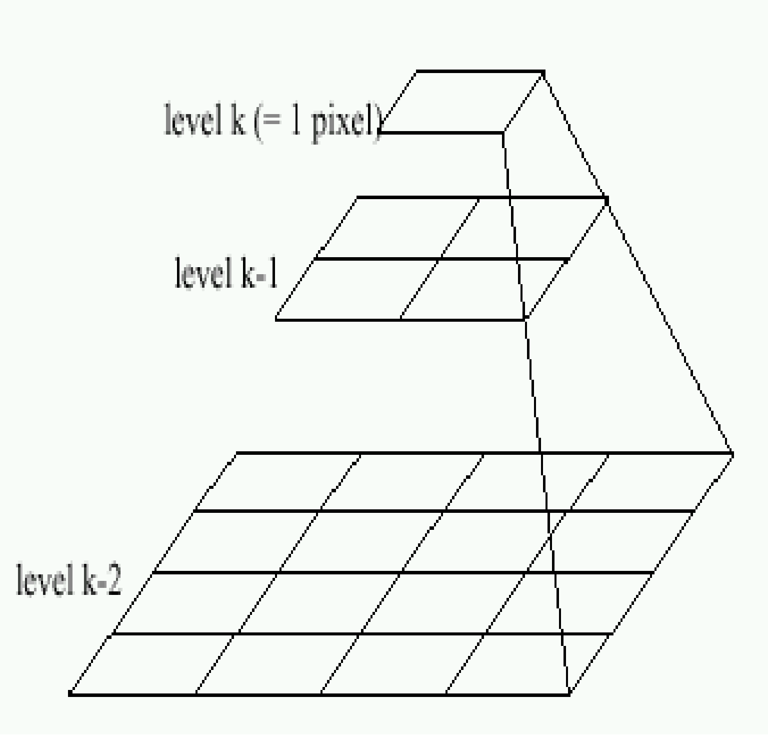
- window <= 2*size of smallest prominent feature

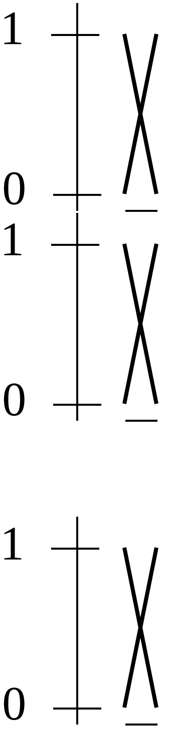


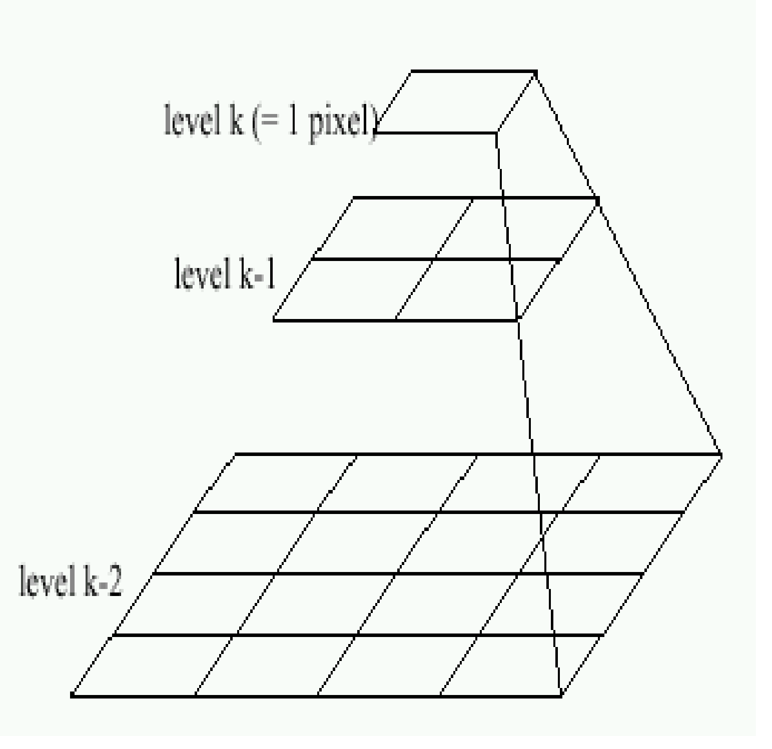
Gaussian pyramid is smooth=> can be subsampled

Laplacian pyramid has narrow band of frequency=> compressed

Pyramid Blending

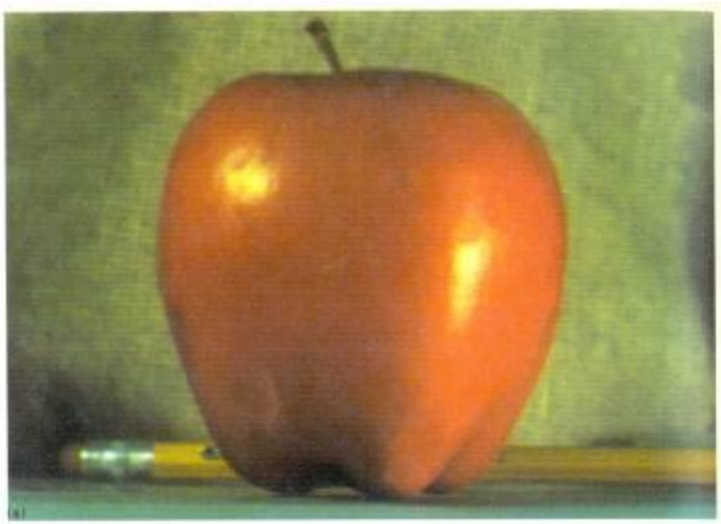


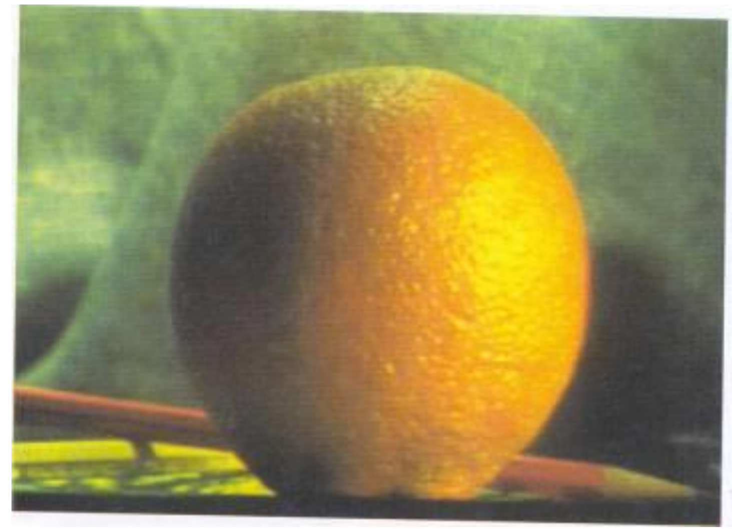


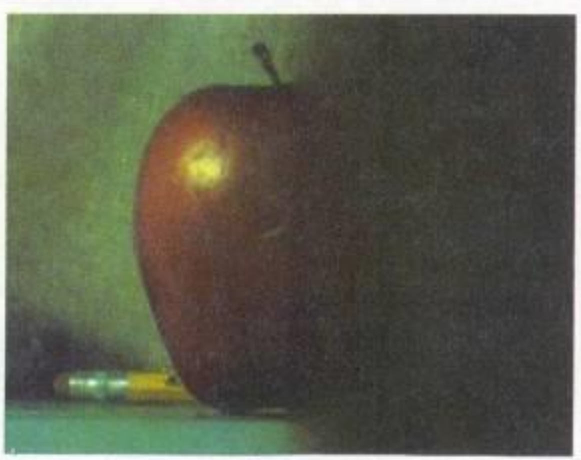


Left pyramid | blend | Right pyramid

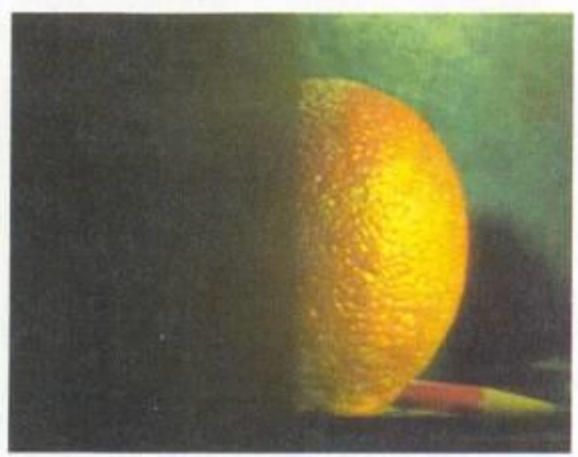
Pyramid Blending



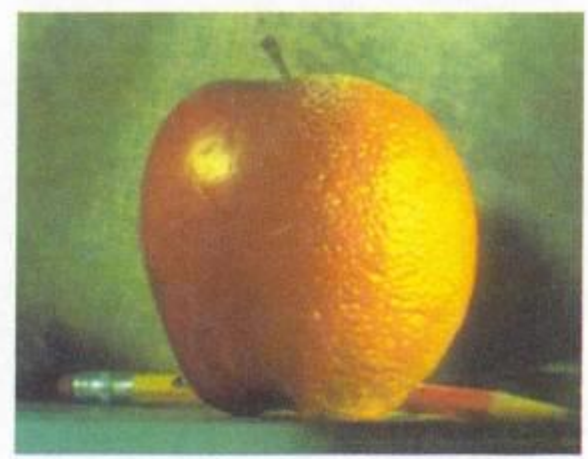




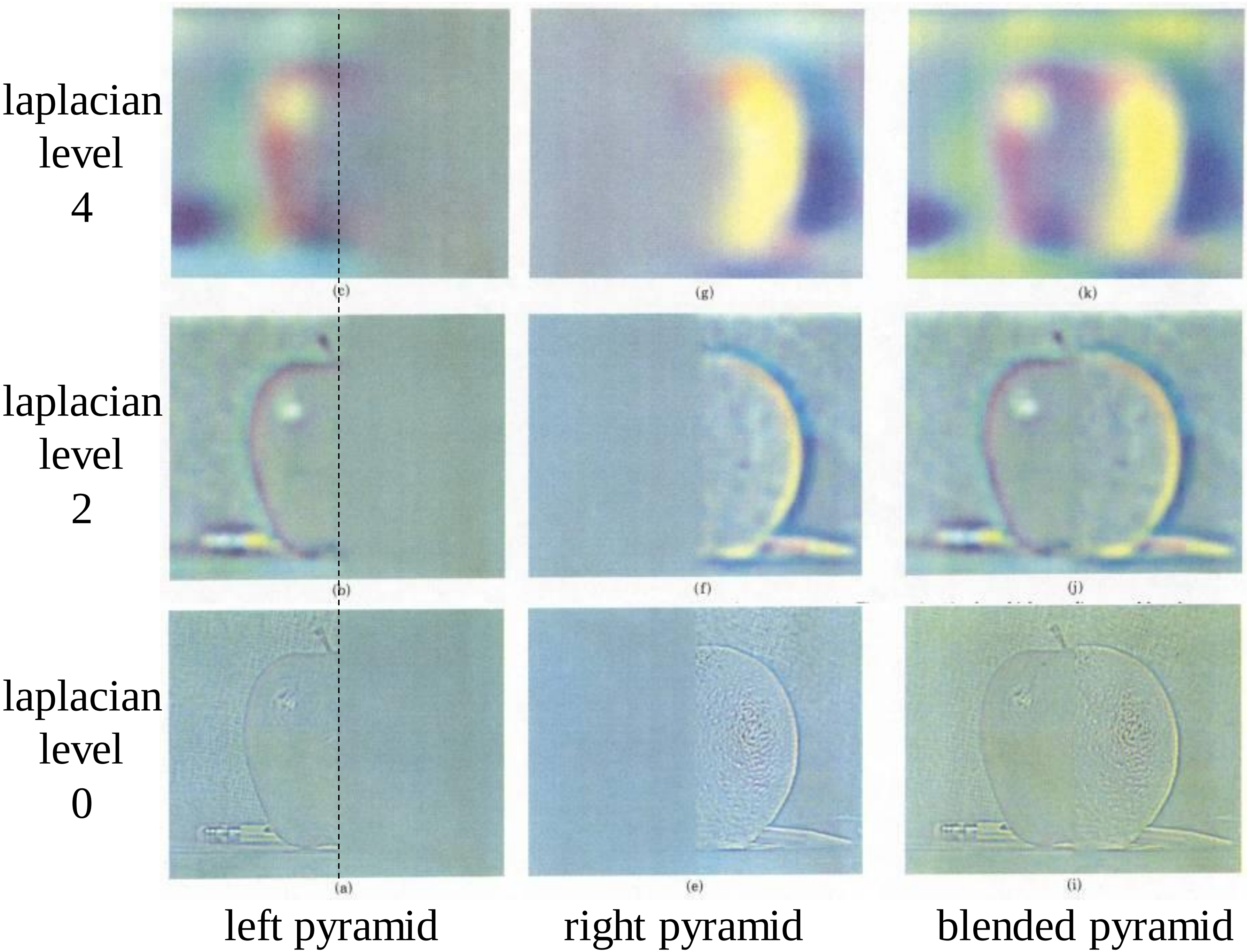
(d)



(h)



(l)

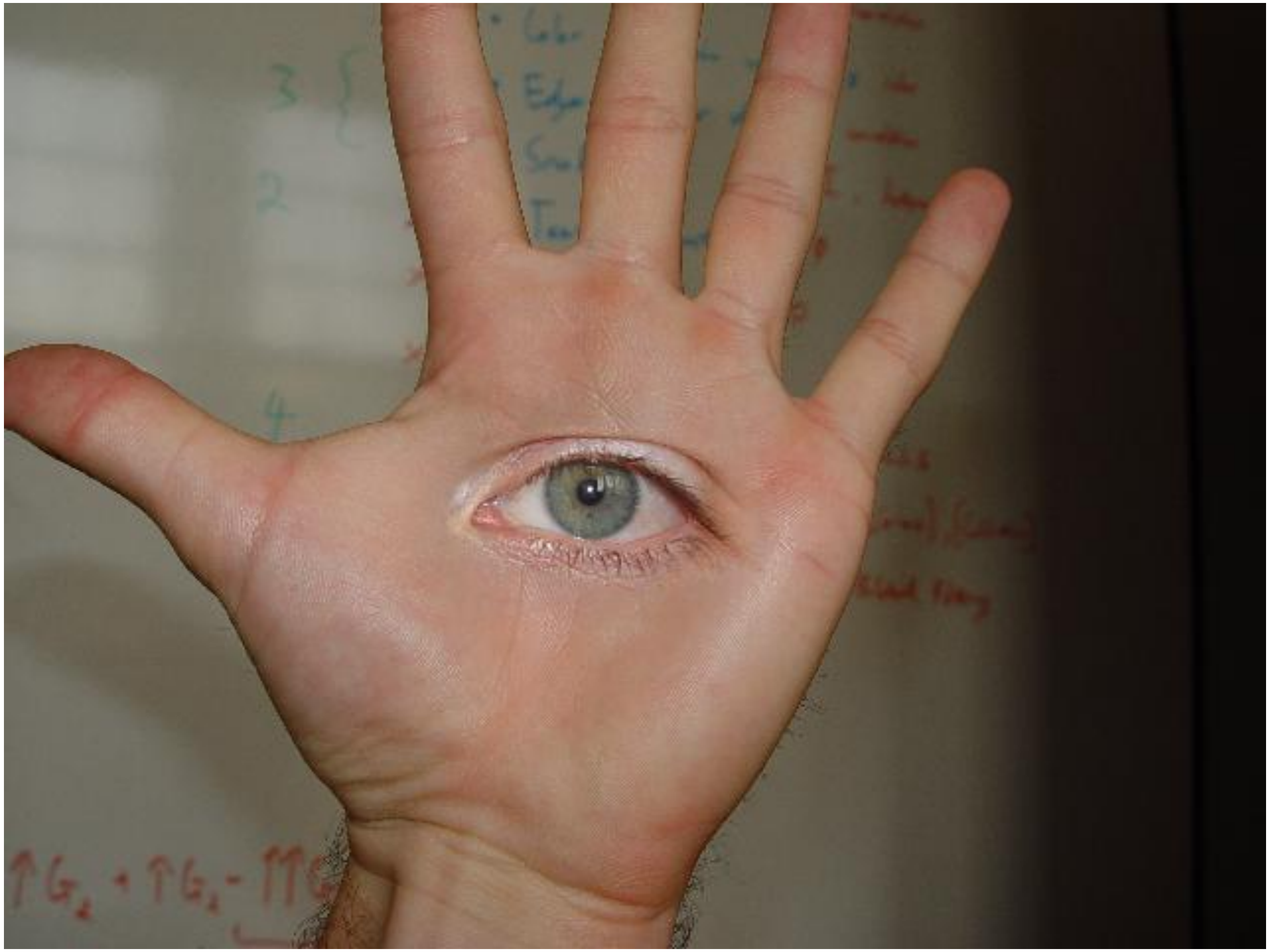

laplacian level 4
laplacian level 2
laplacian level 0
(c)
(g)
(k)
(b)
(f)
(j)
(a)
(e)
(i)
left pyramid
right pyramid
blended pyramid

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

Horror Photo



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

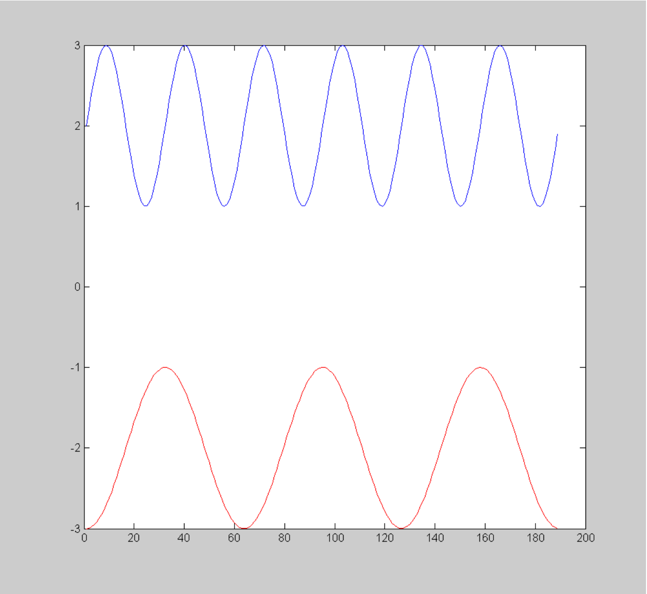
In Pyramid Blending, we decomposed our image into 2nd derivatives (Laplacian) and a low-res image

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

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

Gradient Domain blending (1D)

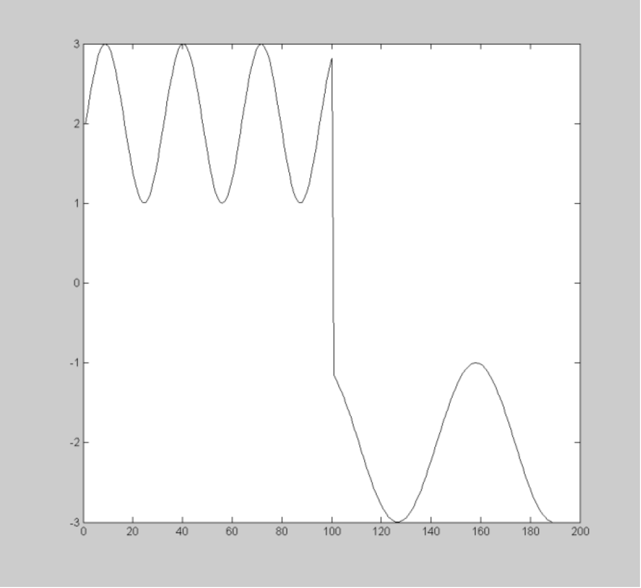
Two signals



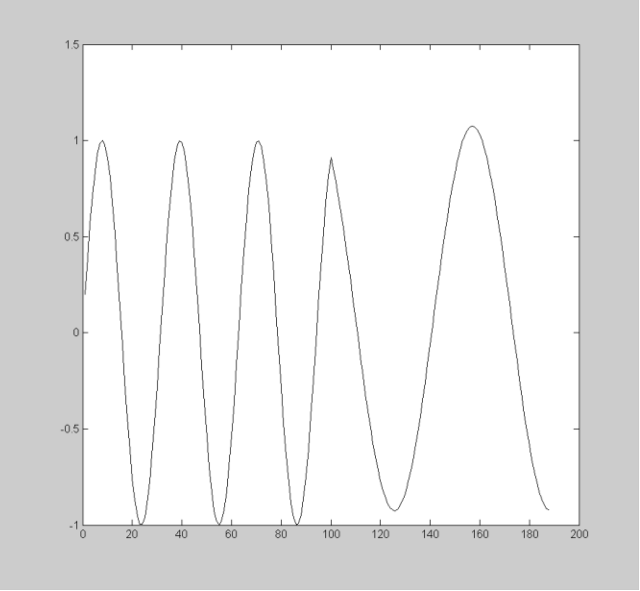
bright

dark

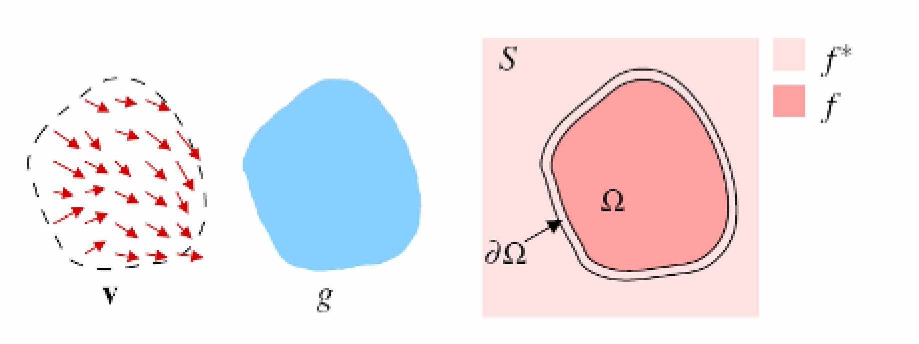
Regular blending



Blending derivatives



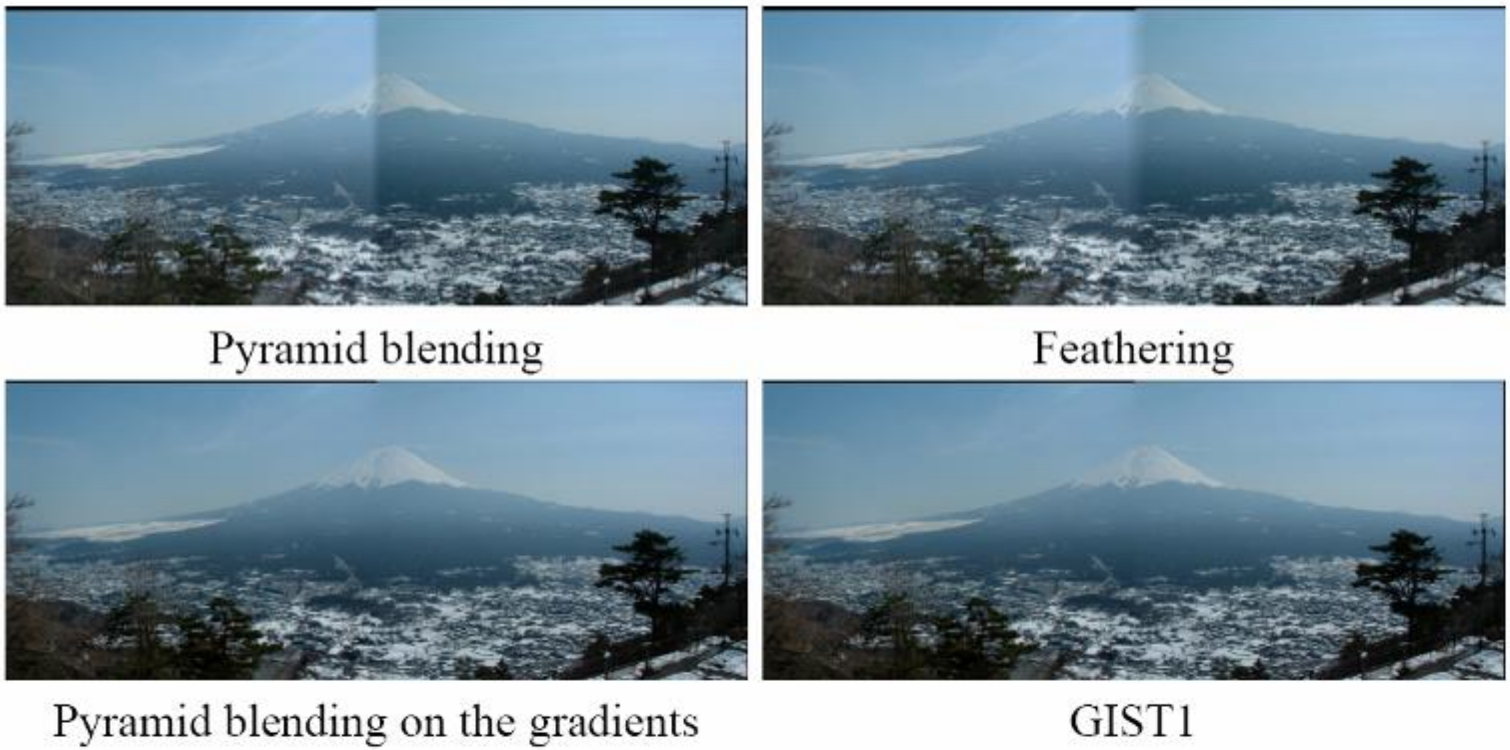
Gradient Domain Blending (2D)



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

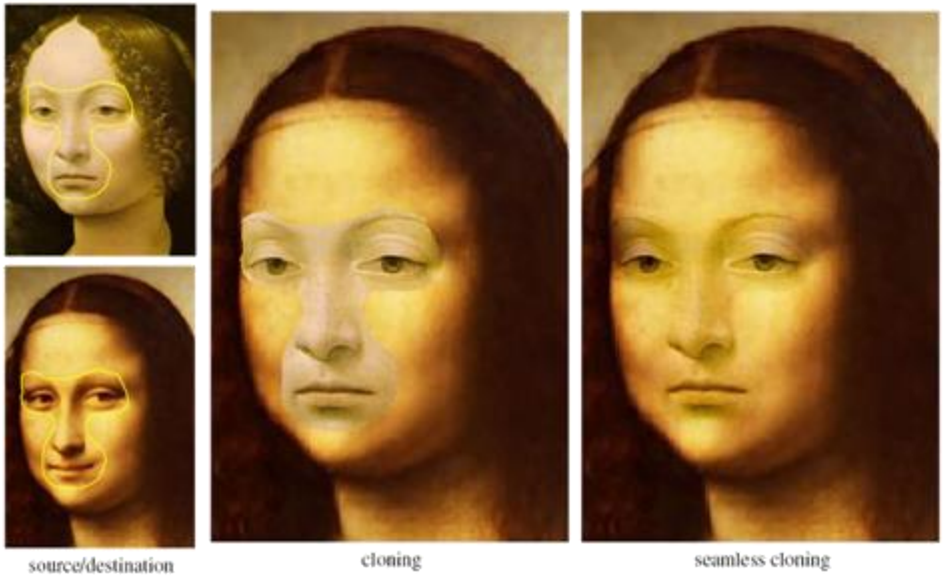


Perez et al., 2003

sources destinations cloning seamless cloning

sources/destinations cloning seamless cloning

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