Image Blending

© NASA

15-463: Computational Photography
Alexei Efros, CMU, Fall 2008

Image Compositing

Compositing Procedure

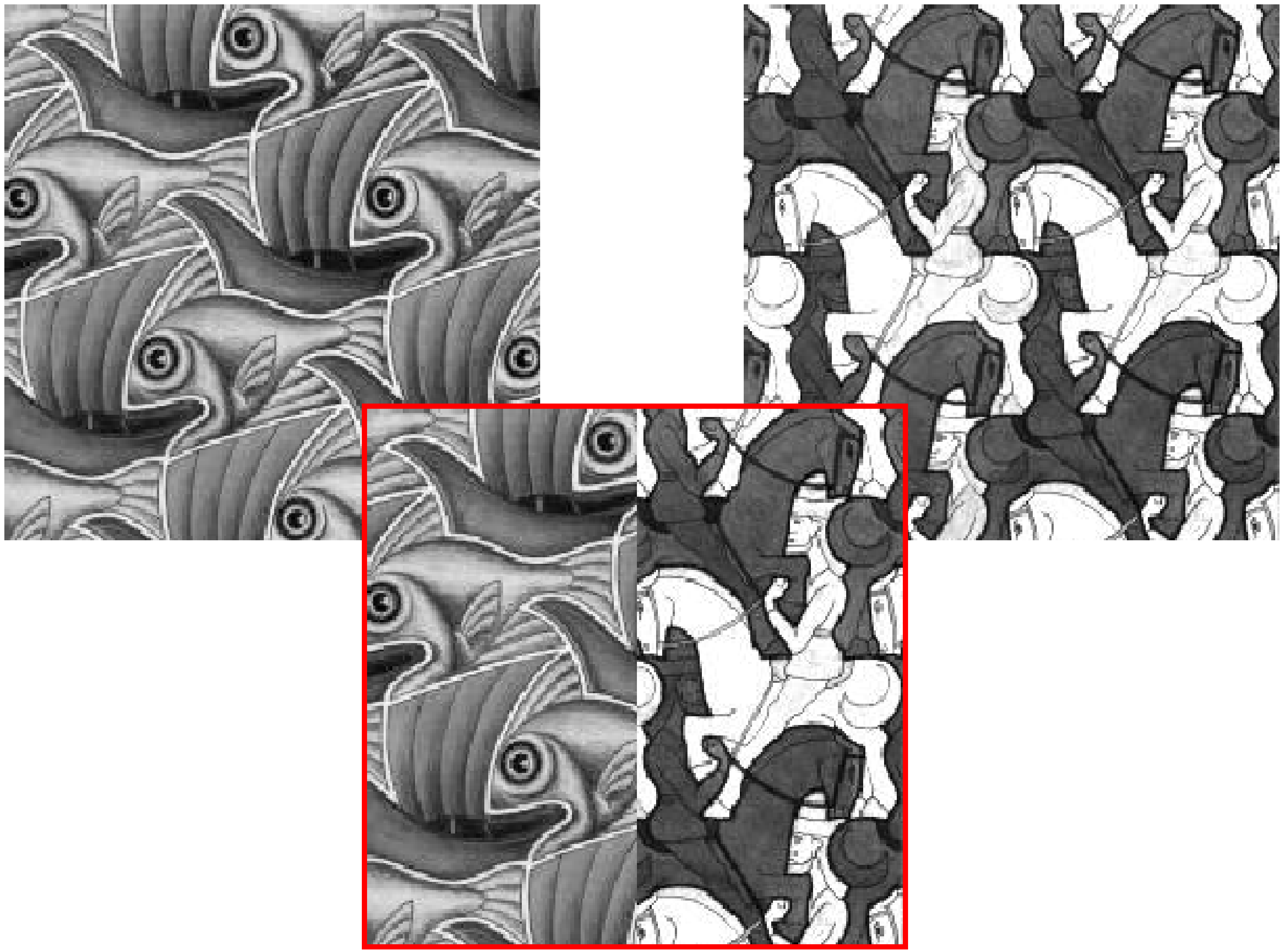
1. Extract Sprites (e.g using *Intelligent Scissors* in Photoshop)



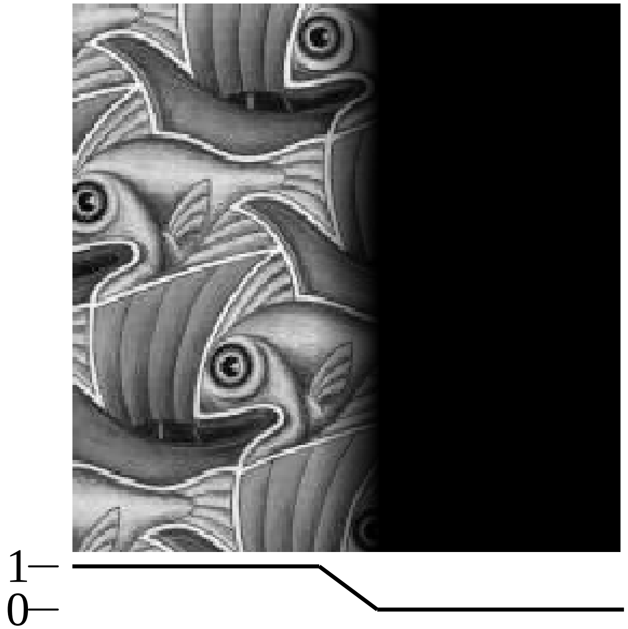
2. Blend them into the composite (in the right order)

Composite by
David Dewey

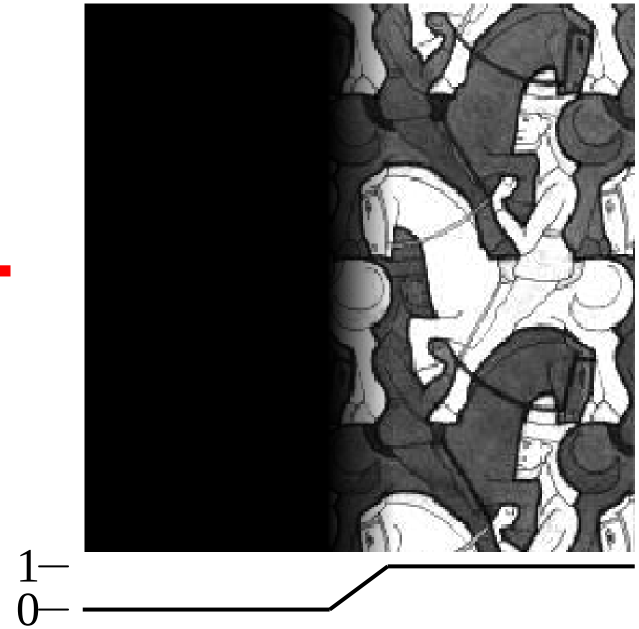
Need blending



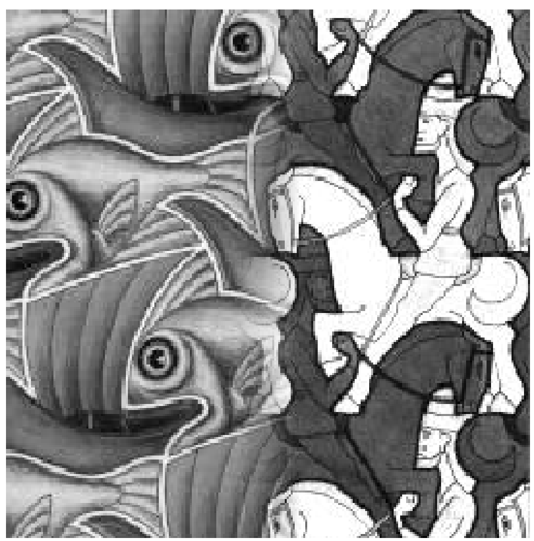
Alpha Blending / Feathering



+



=

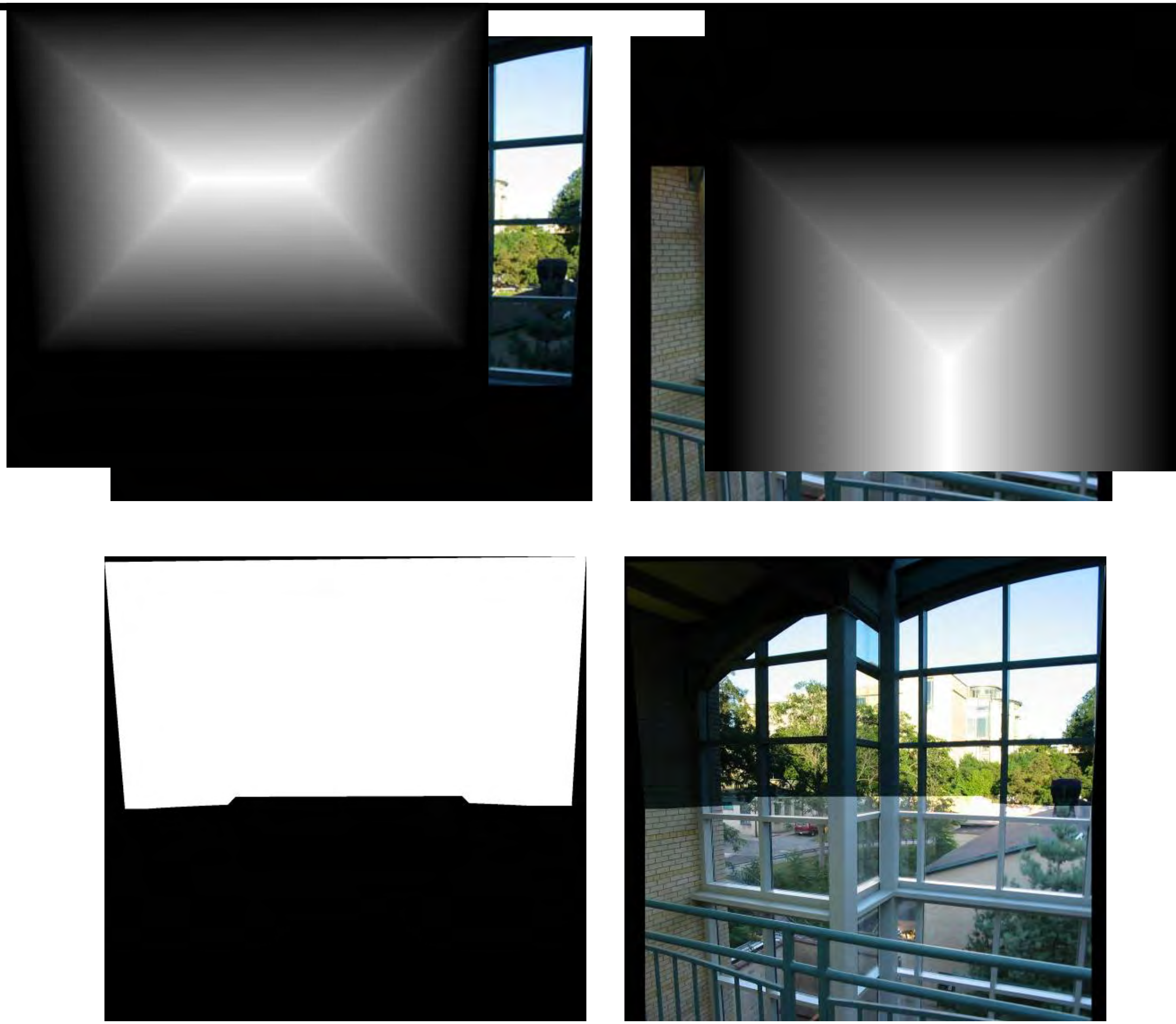


$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

Setting alpha: simple averaging

Alpha = .5 in overlap region

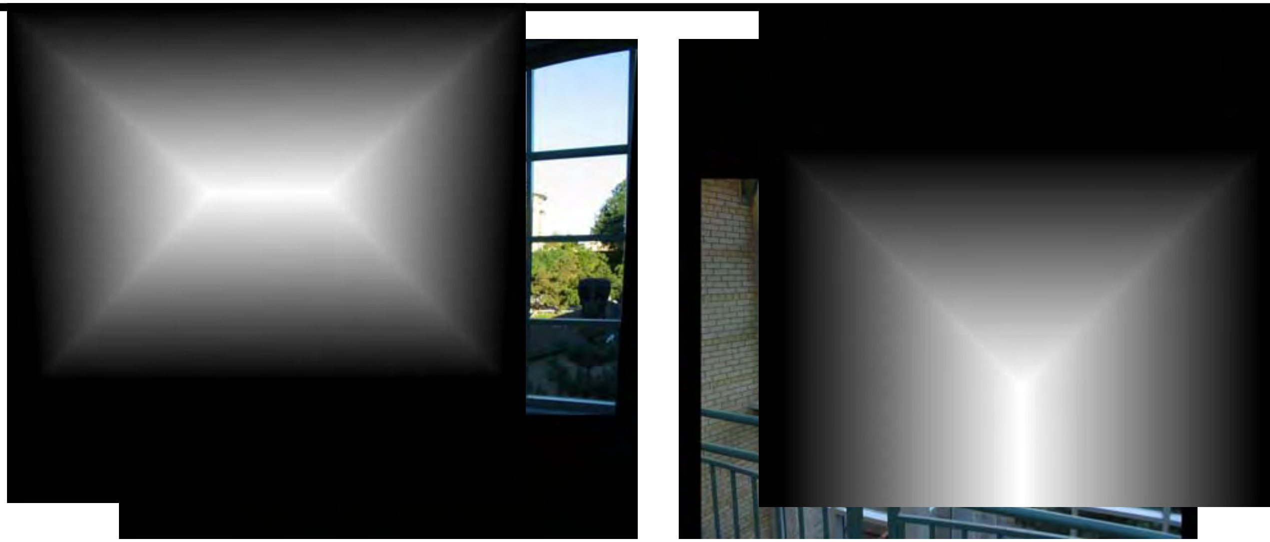
Setting alpha: center seam



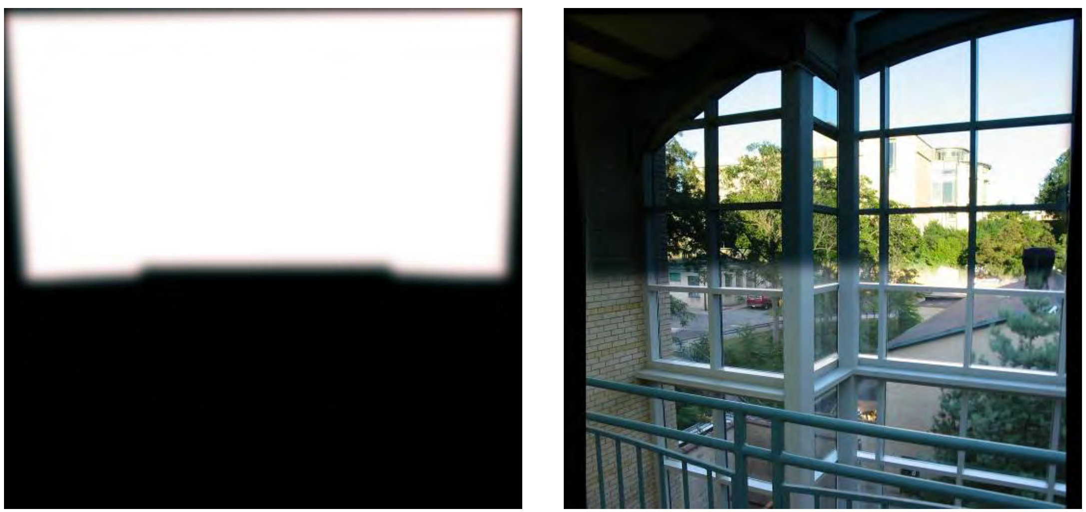
Distance
Transform
`bwdist`

Alpha = logical(dtrans1>dtrans2)

Setting alpha: blurred seam

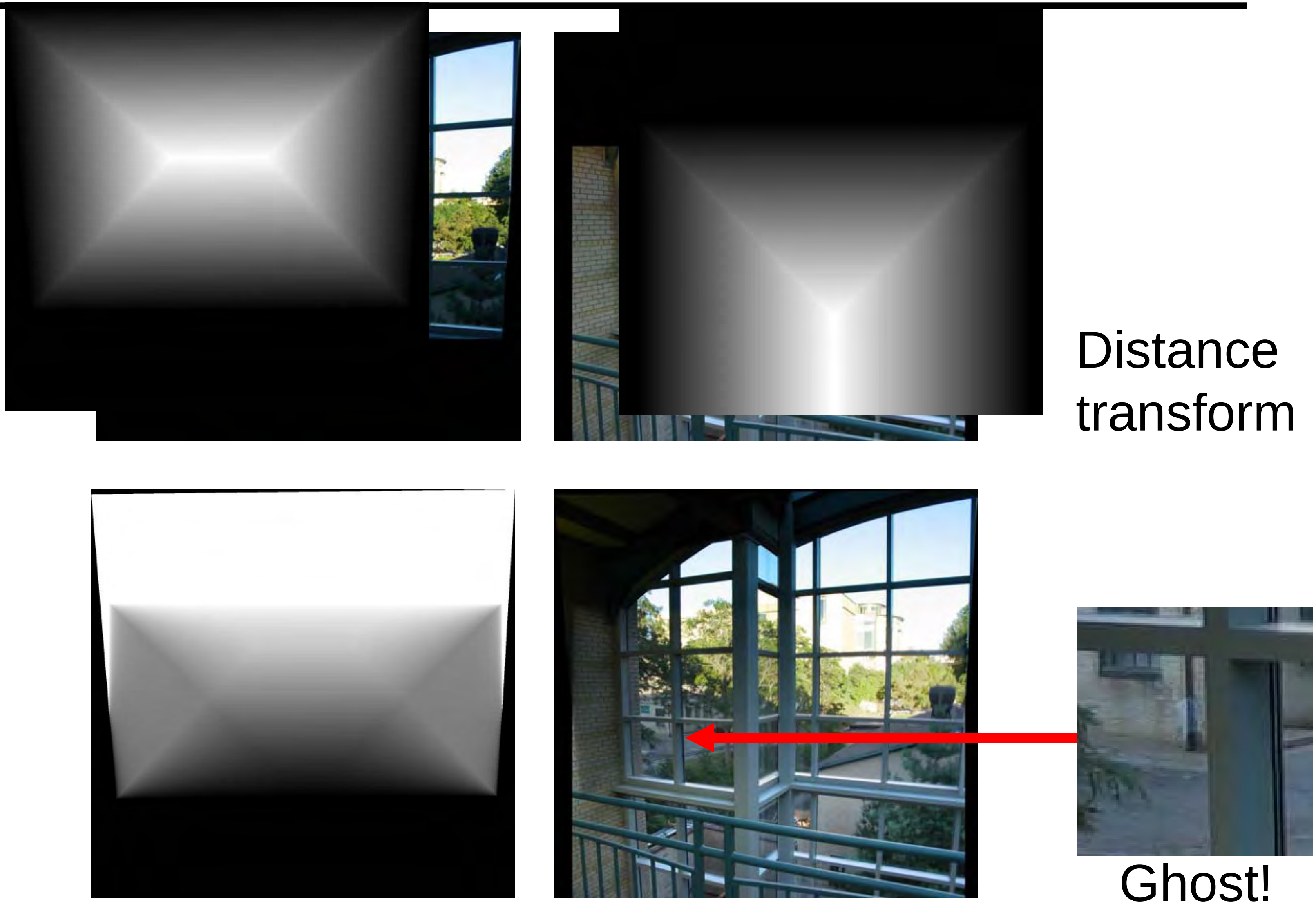


Distance transform



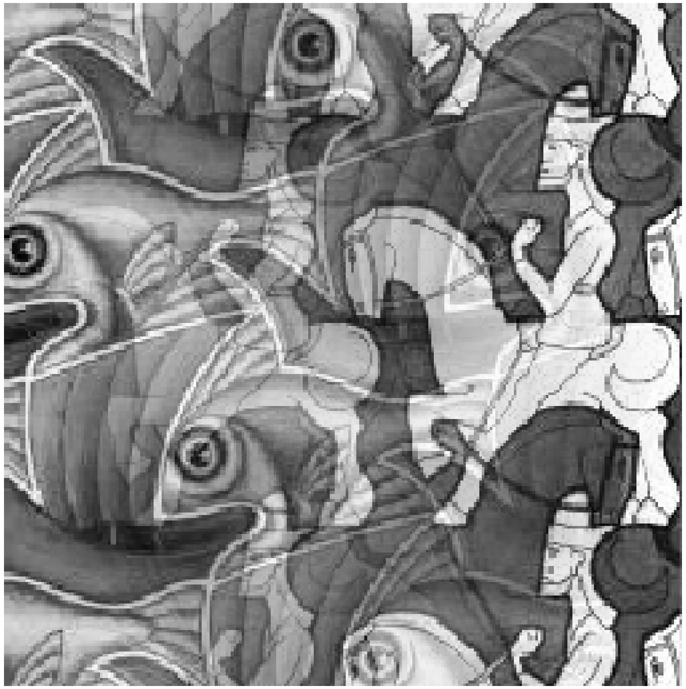
Alpha = blurred

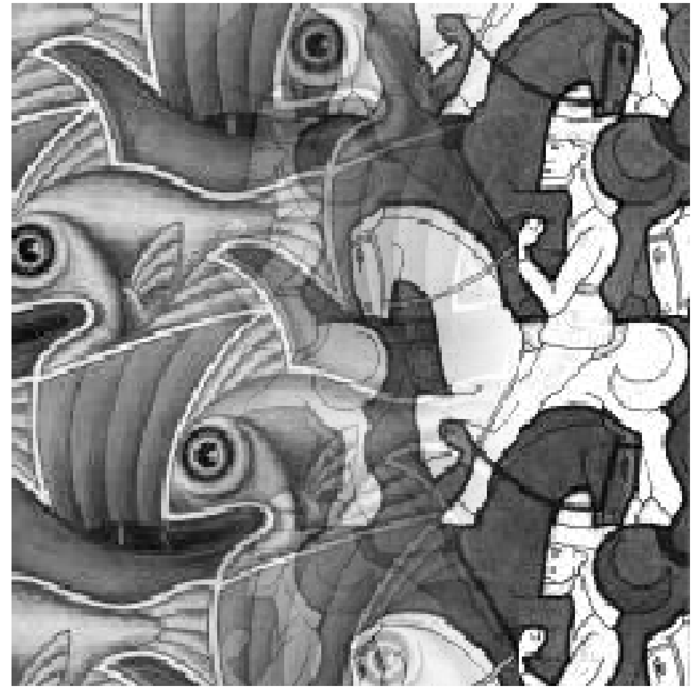
Setting alpha: center weighting

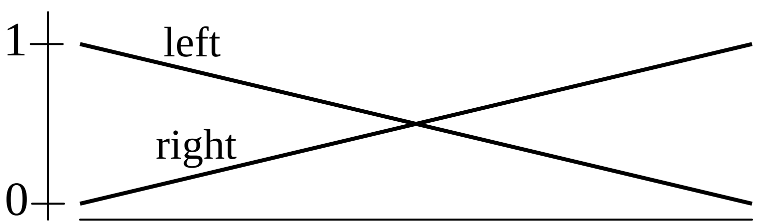


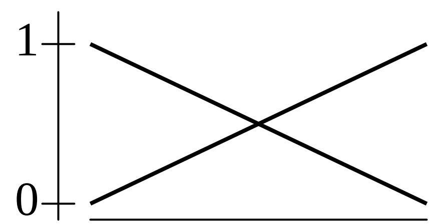
Alpha = dtrans1 / (dtrans1+dtrans2)

Affect of Window Size

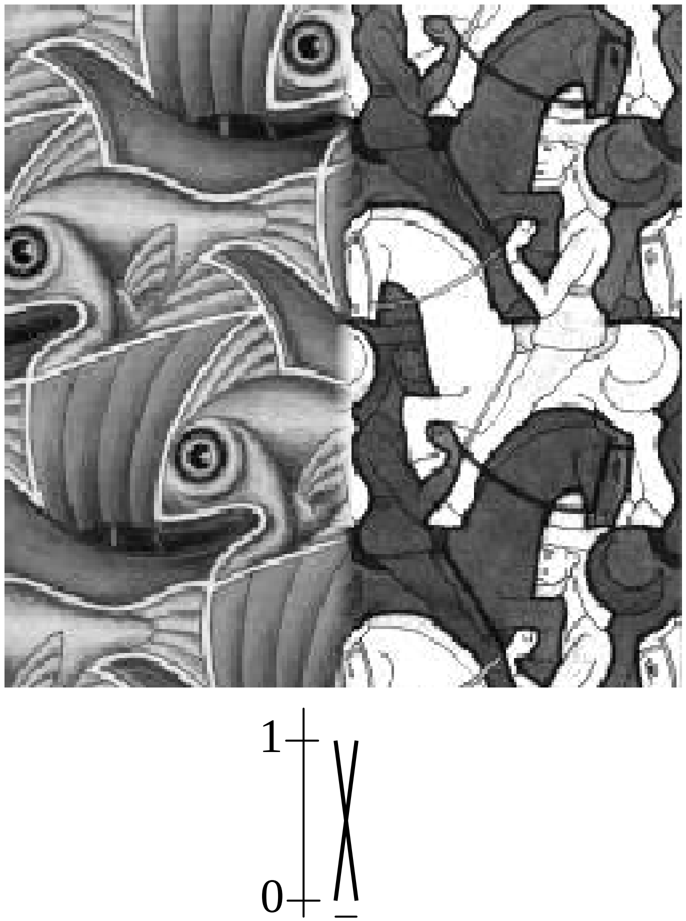


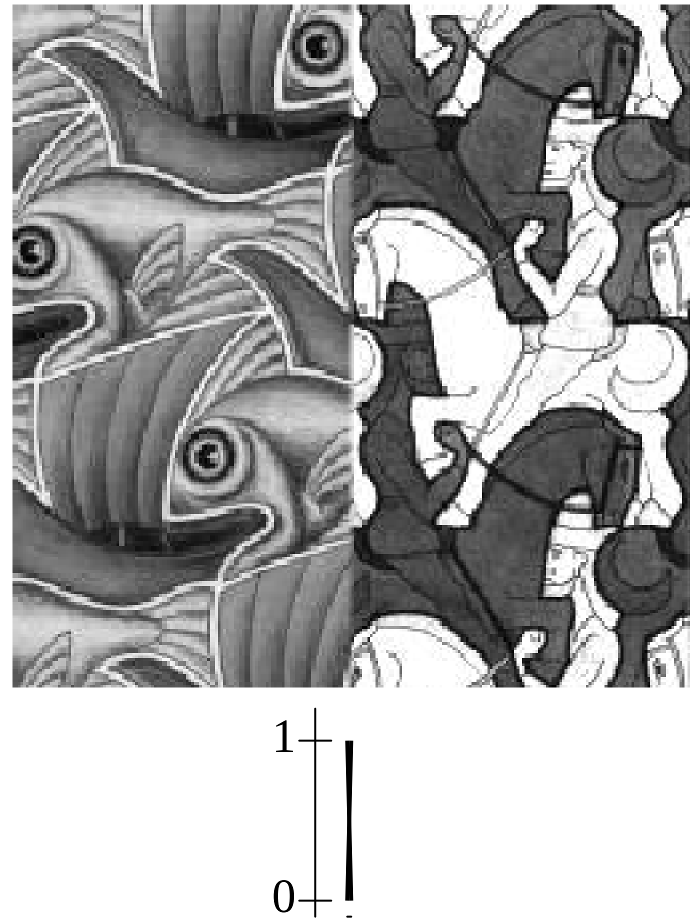




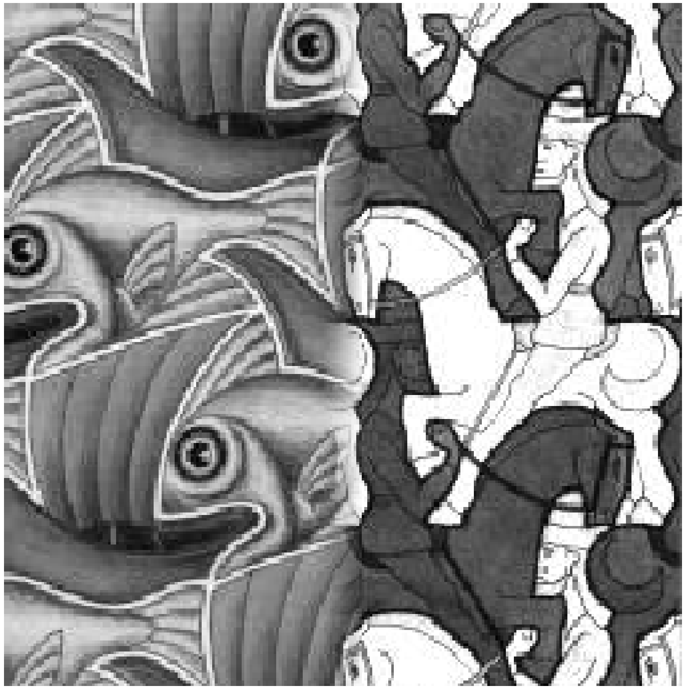


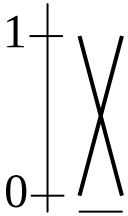
Affect of Window Size





Good Window Size





"Optimal" Window: smooth but not ghosted

What is the Optimal Window?

To avoid seams

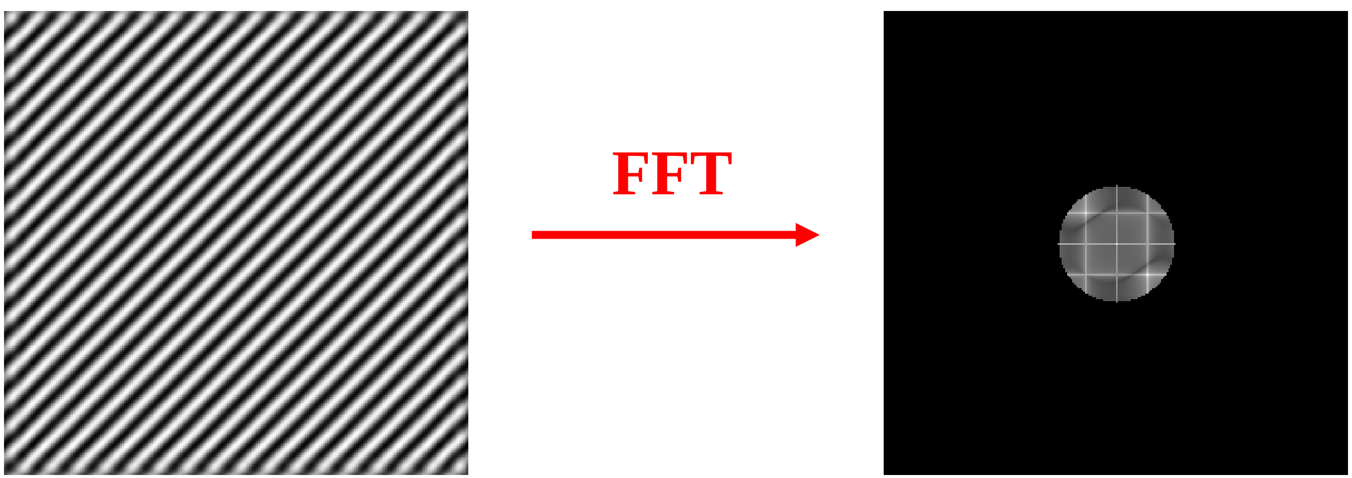
- window = size of largest prominent feature

To avoid ghosting

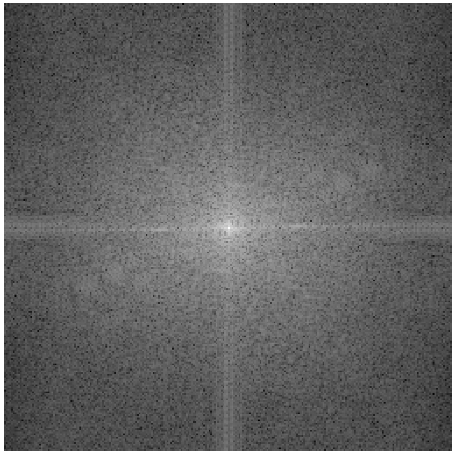
- window <= 2*size of smallest prominent feature

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



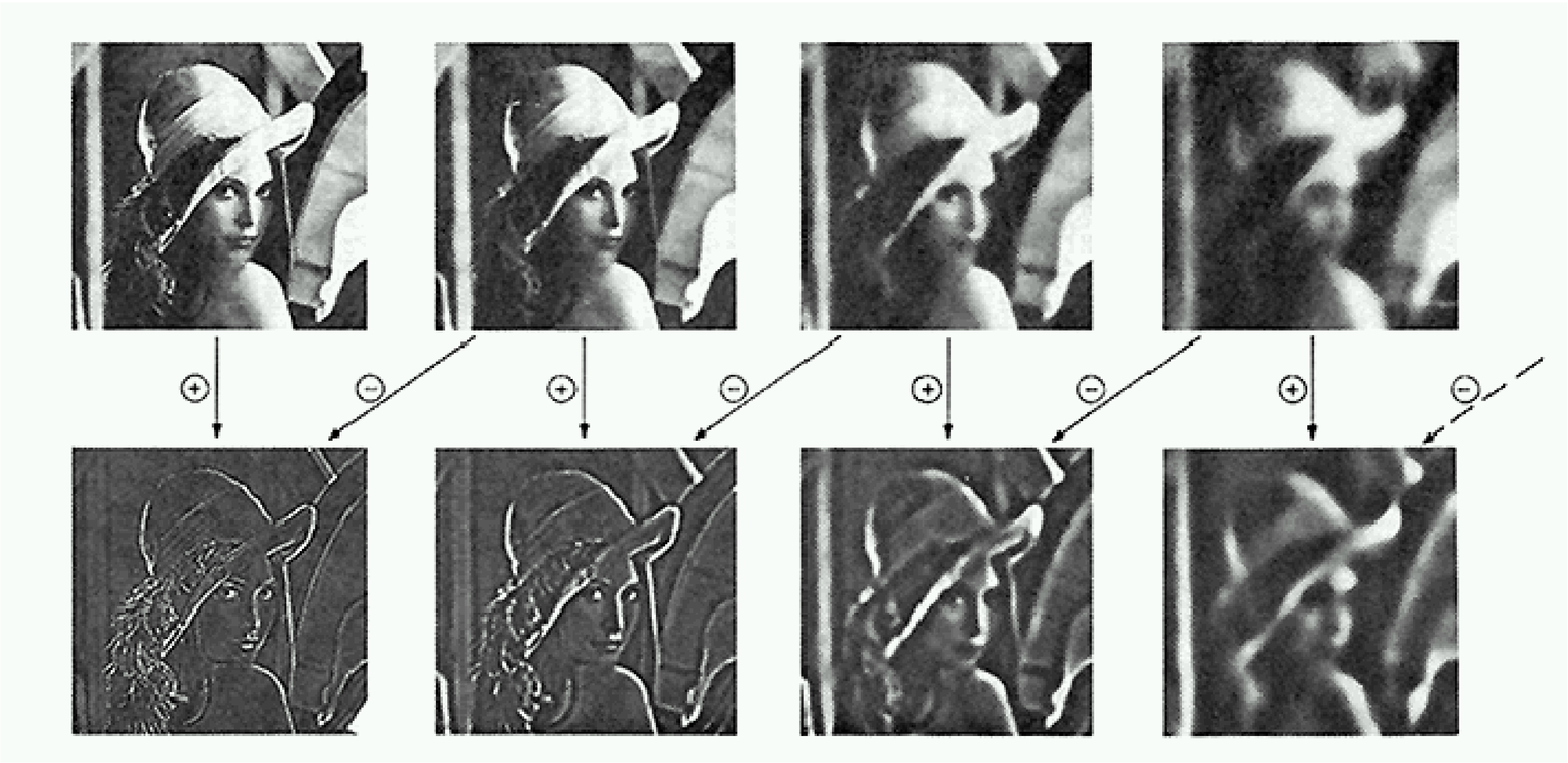
Idea (Burt and Adelson)

- 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*

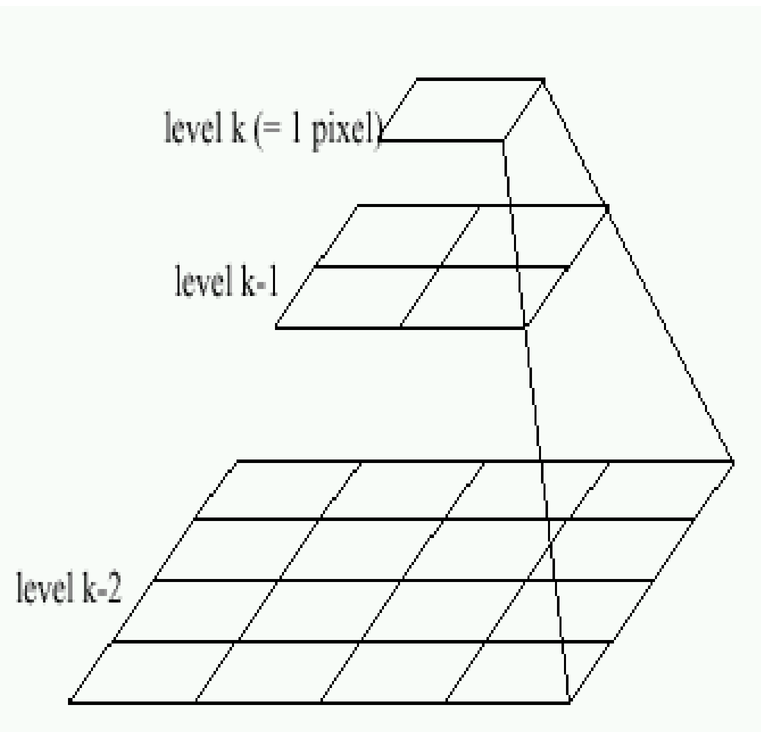
Octaves in the Spatial Domain

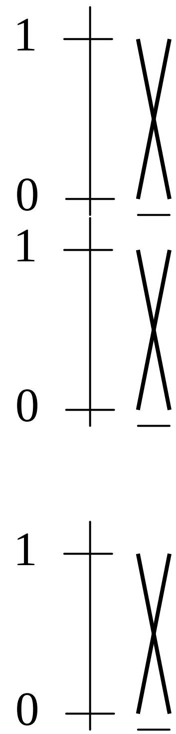
Lowpass Images

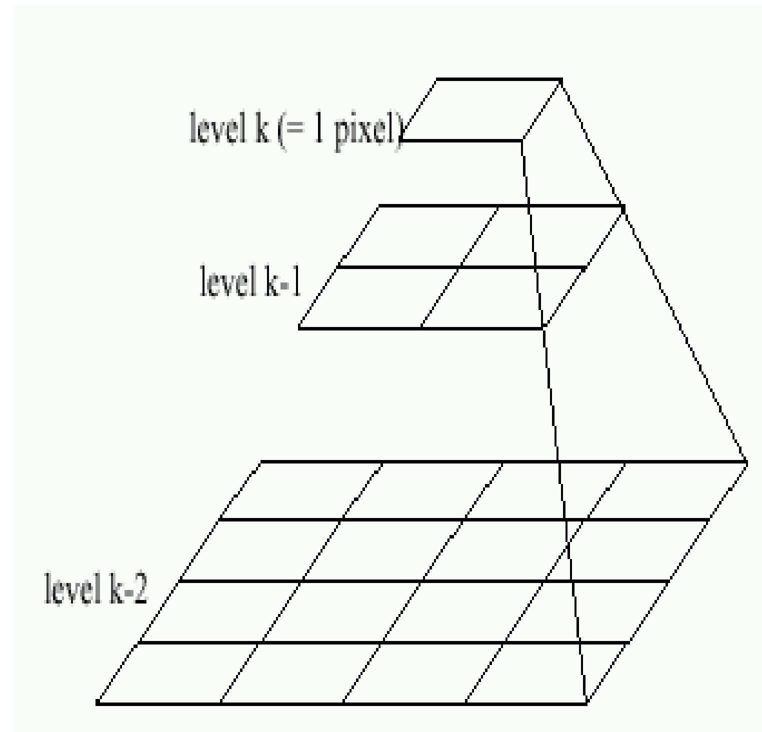


Bandpass Images

Pyramid Blending





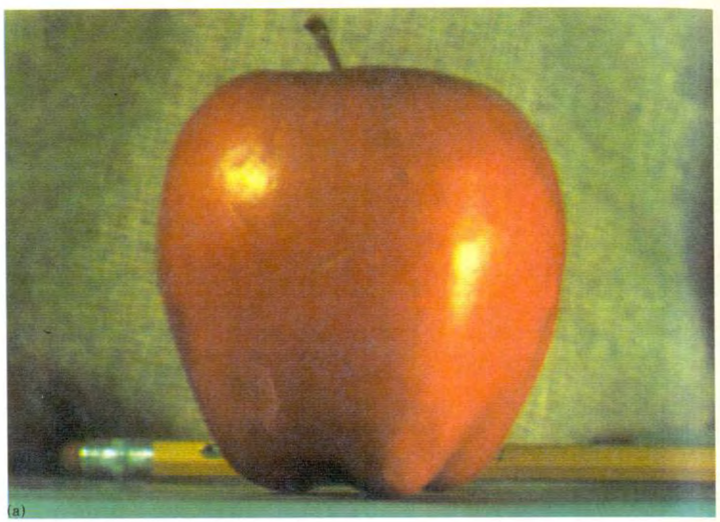


Left pyramid

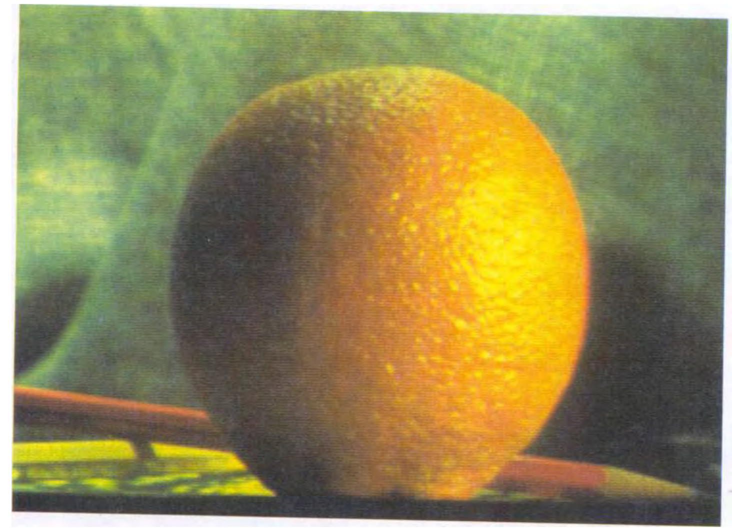
blend

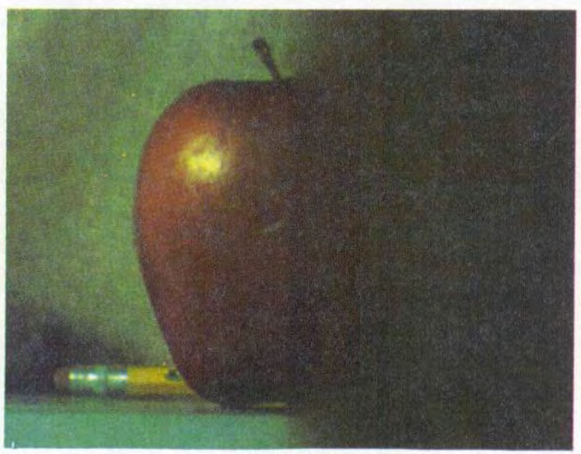
Right pyramid

Pyramid Blending

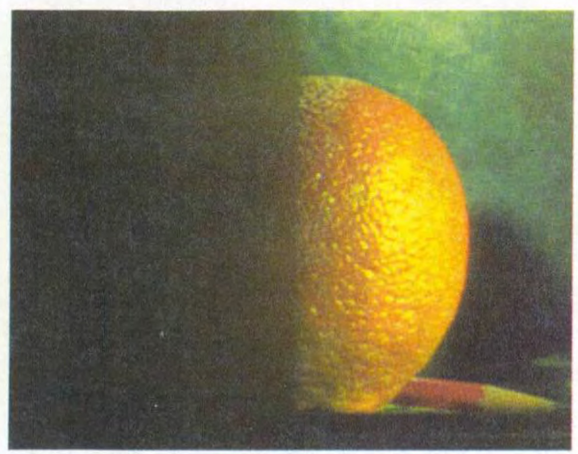


(a)

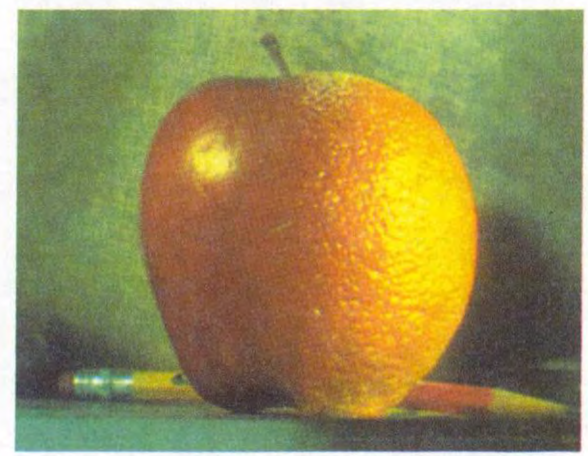




(d)



(h)

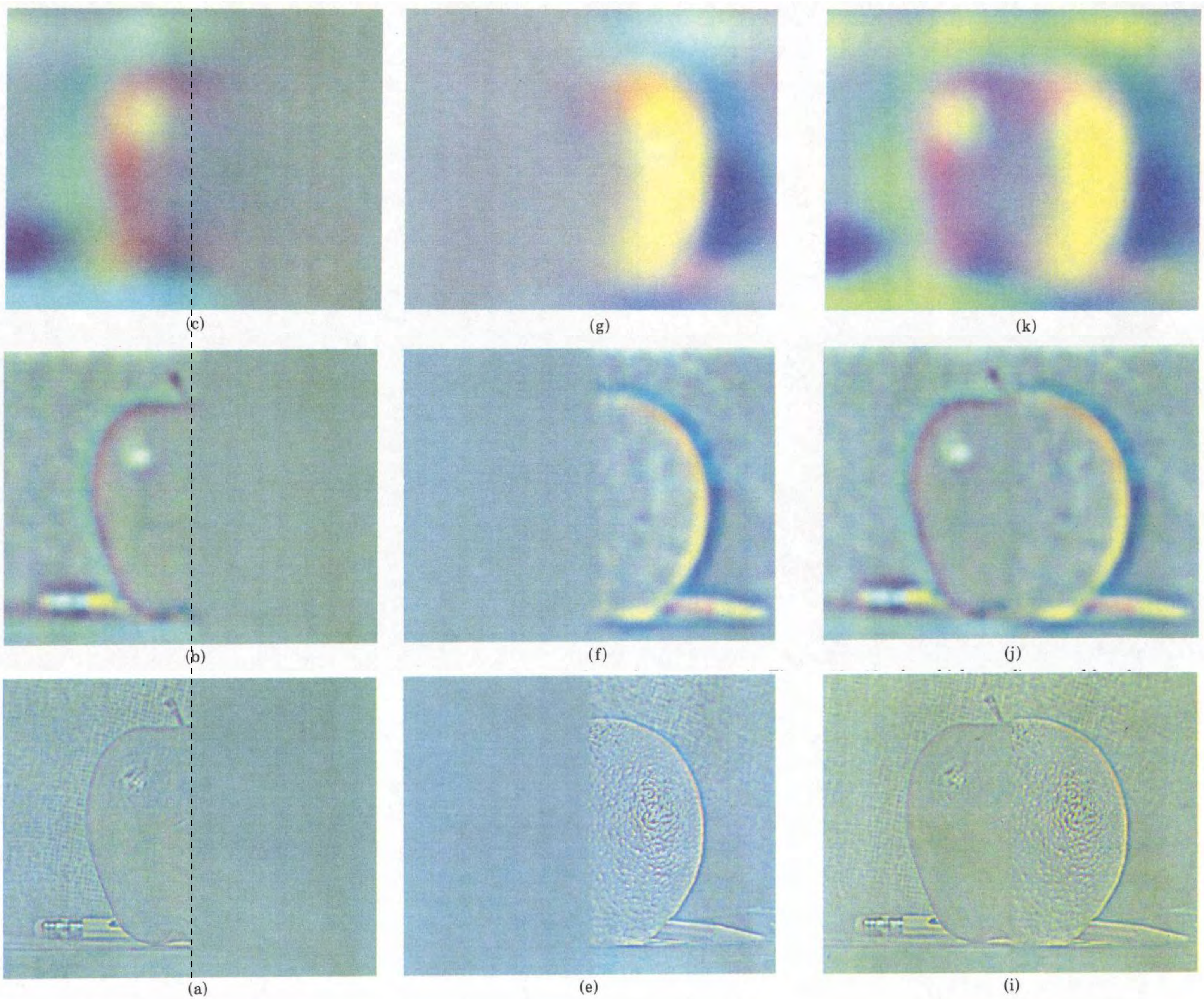


(l)

laplacian level 4

laplacian level 2

laplacian level 0



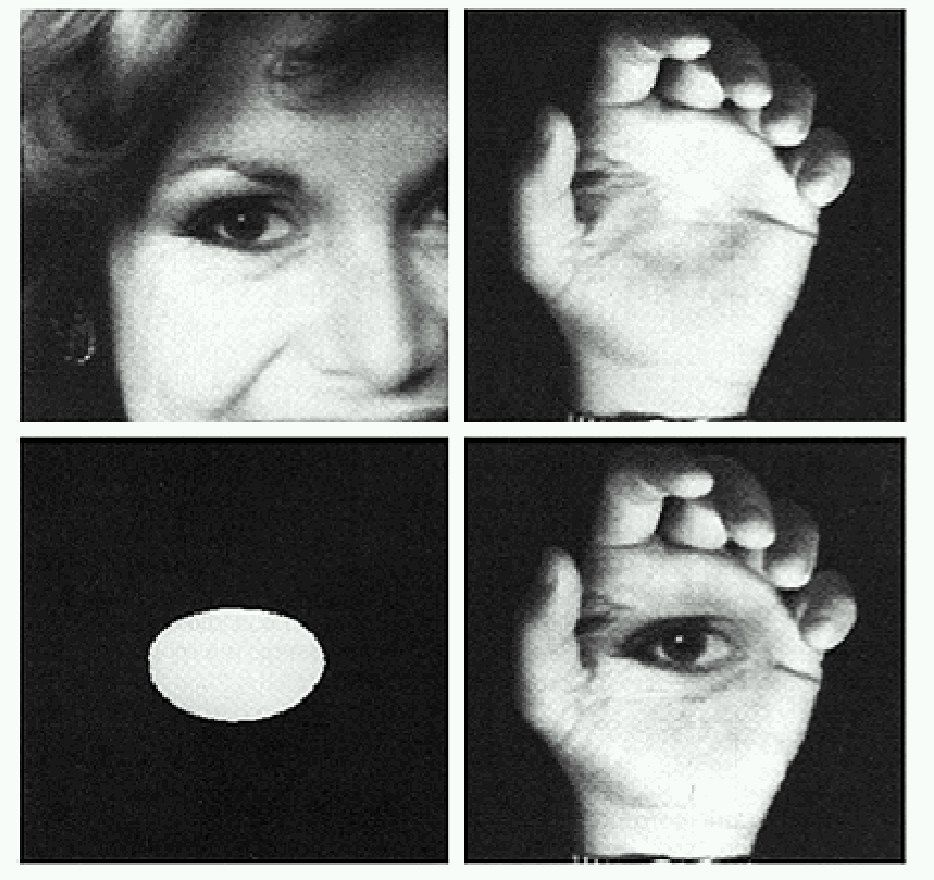
left pyramid right pyramid blended pyramid

Laplacian Pyramid: Blending

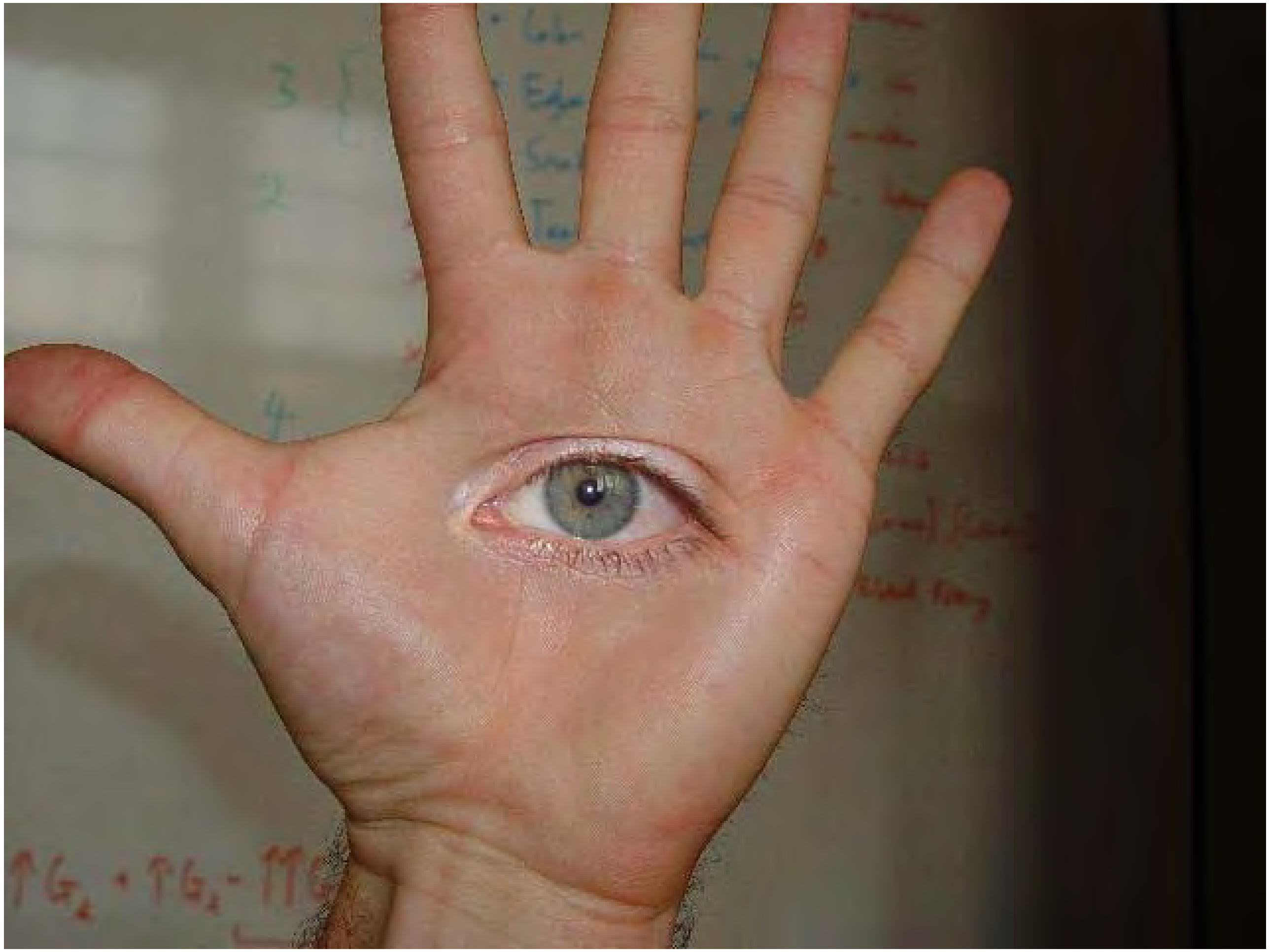
General Approach:

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



© david dmartin (Boston College)

Results from this class (fall 2005)

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 alpha

2-band Blending

Low frequency (λ > 2 pixels)

High frequency (λ < 2 pixels)

Linear Blending

2-band Blending

Don't blend, CUT!

Moving objects become ghosts

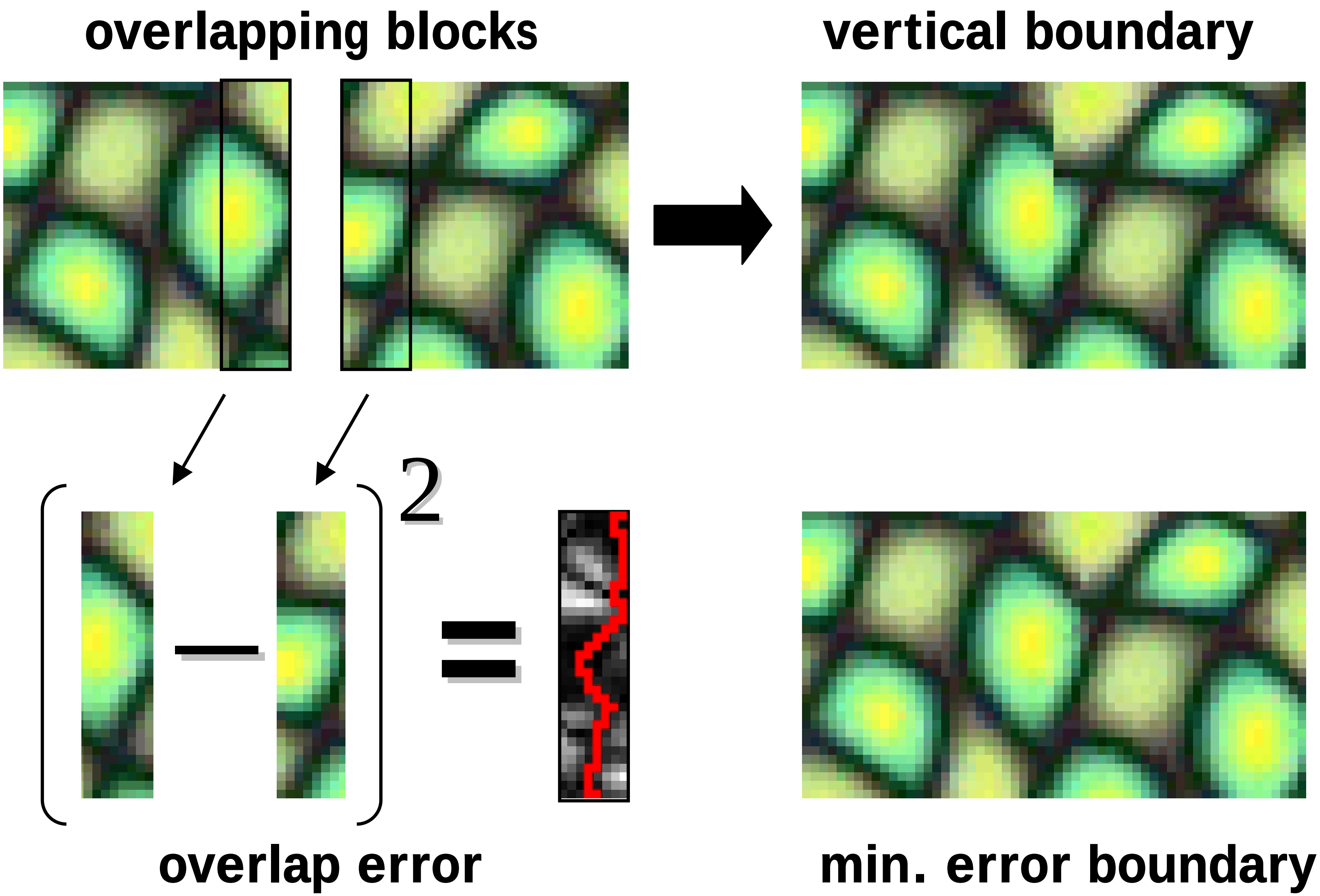
So far we only tried to blend between two images. What about finding an optimal seam?

Davis, 1998

Segment the mosaic

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

Minimal error boundary



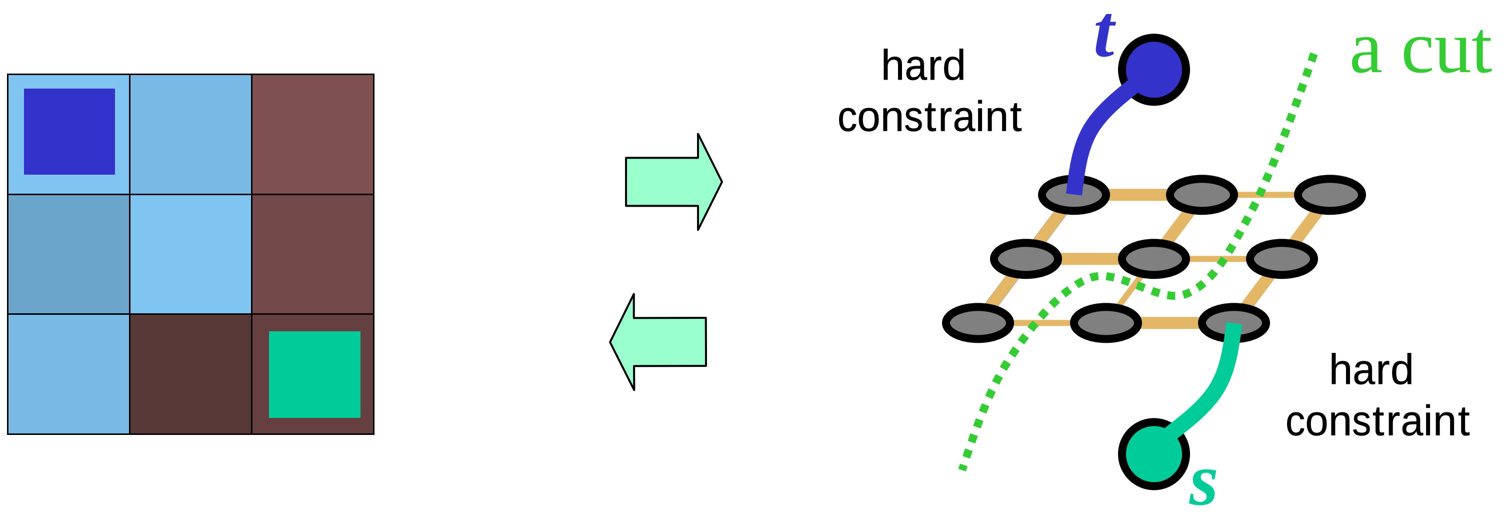
Graphcuts

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


(a) Girl (4/2/12) (b) Ballet (4/7/14) (c) Boy (6/2/13)
(c) Grandpa (4/2/11) (d) Twins (4/4/12)

Interactive segmentation using graphcuts

Gradient Domain

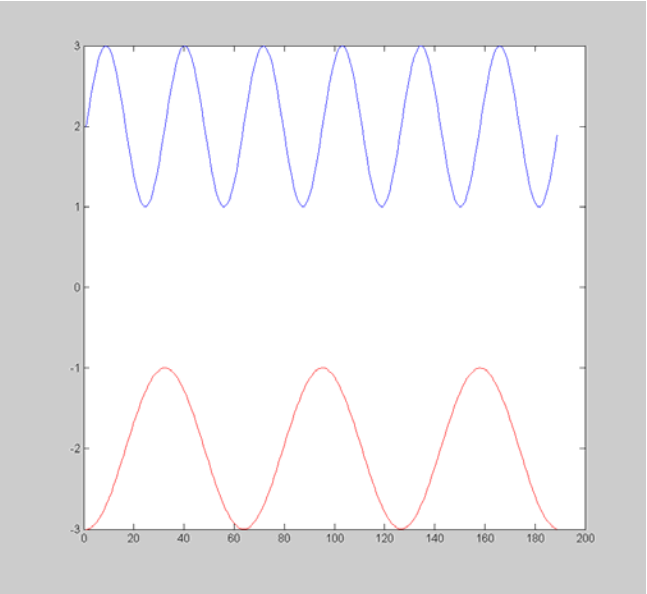
In Pyramid Blending, we decomposed our image into 2^{nd} derivatives (Laplacian) and a low-res image

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)

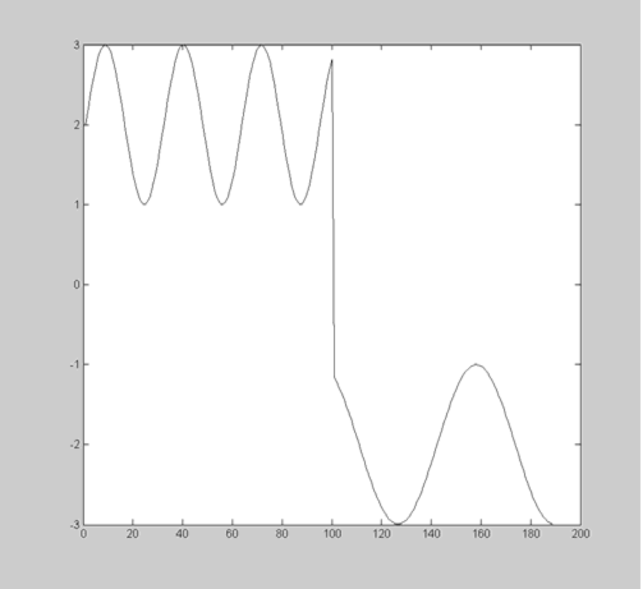
Two signals

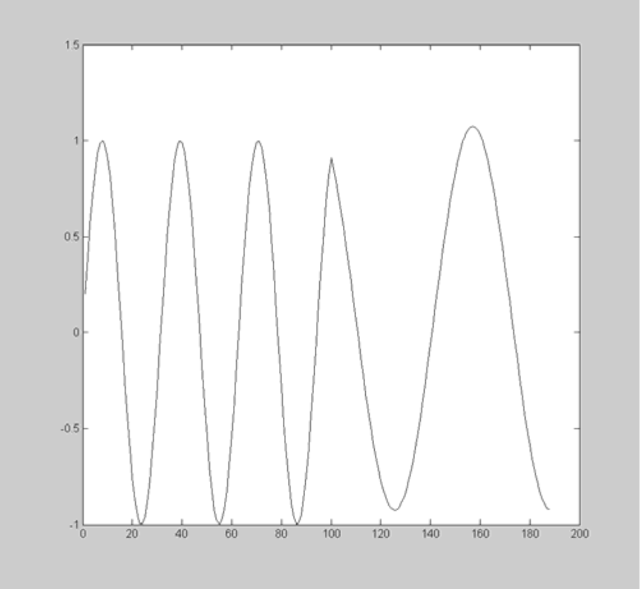


bright

dark

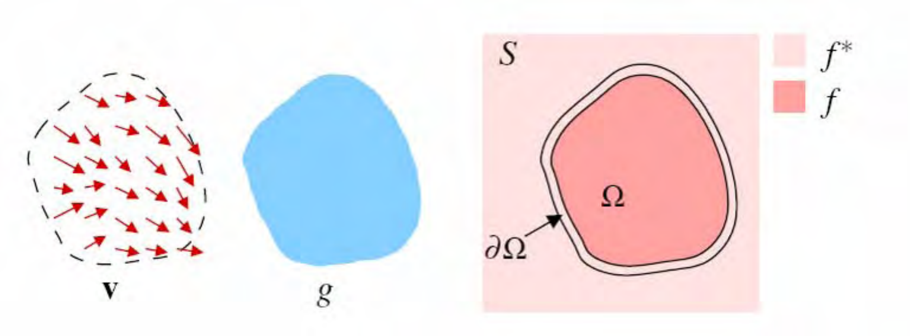
Regular blending





Blending derivatives

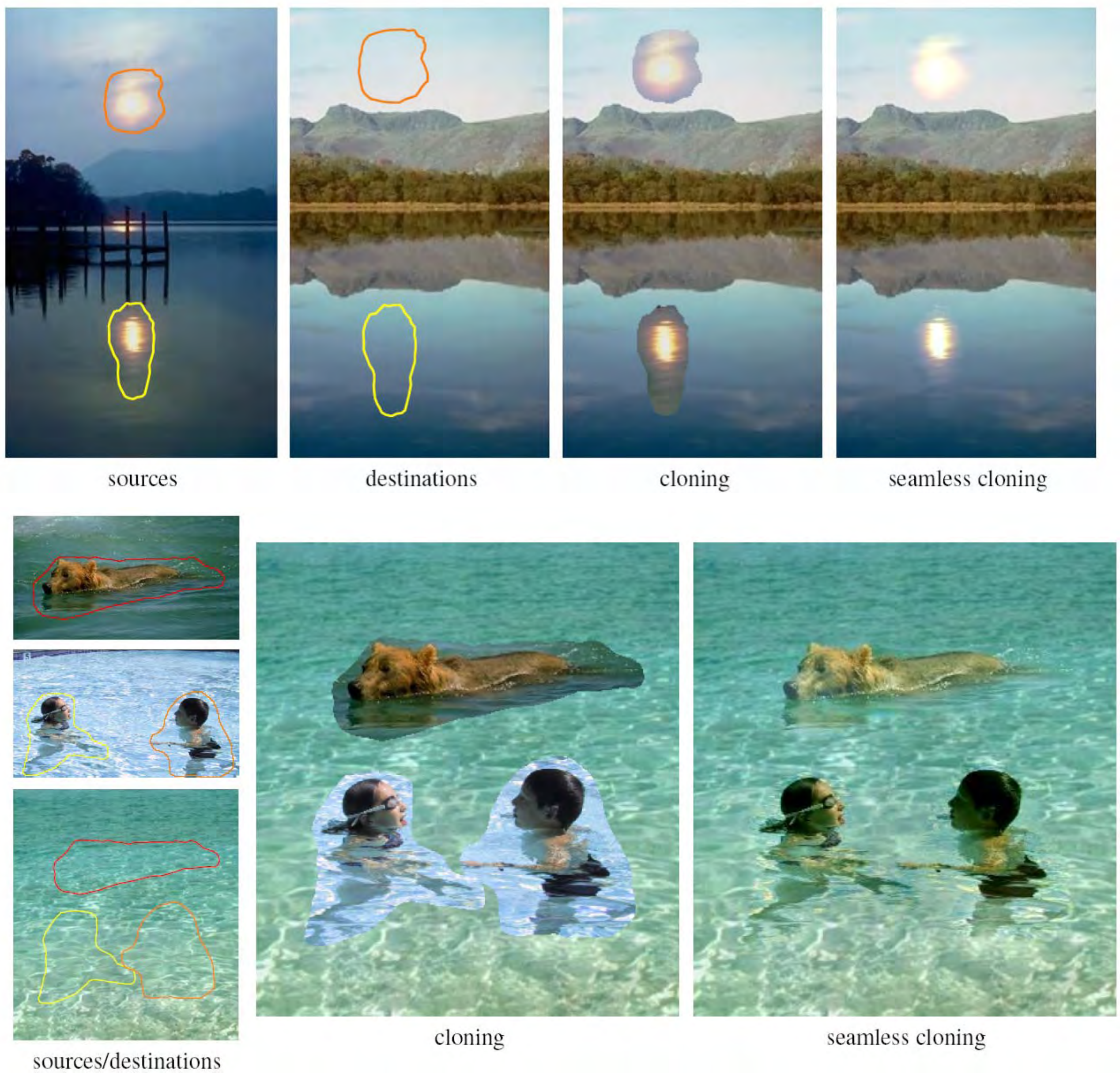
Gradient Domain Blending (2D)



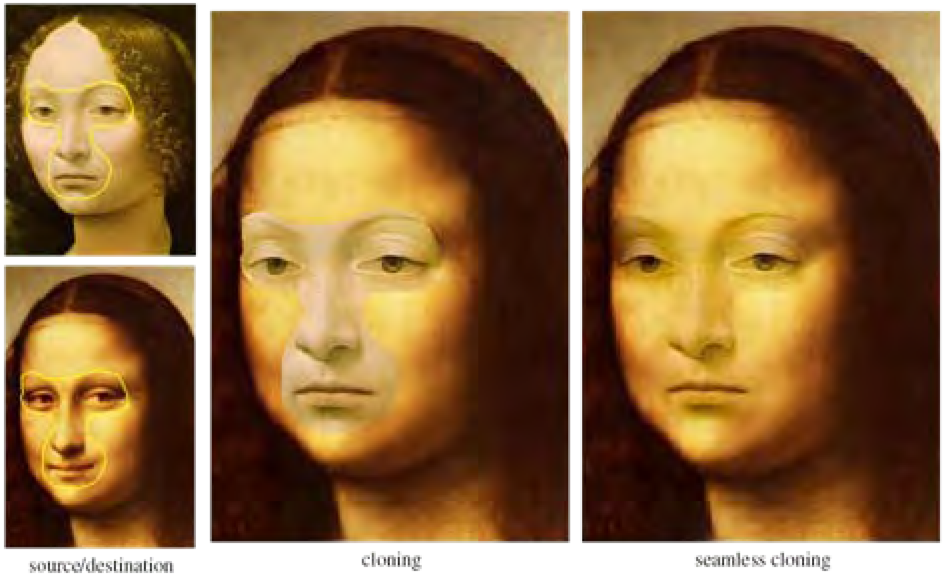
Trickier in 2D:

- 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.

Perez et al., 2003



Perez et al, 2003

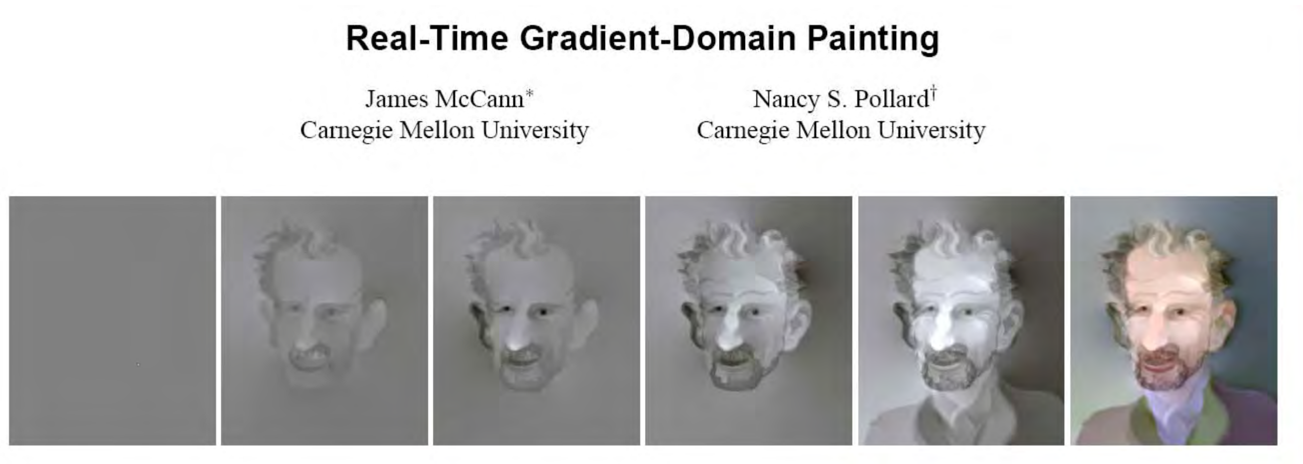


editing

Limitations:

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

Painting in Gradient Domain! (McCann)


Real-Time Gradient-Domain Painting

James McCann*
Carnegie Mellon University

Nancy S. Pollard†
Carnegie Mellon University

Code available!

See Jim’s talk this Friday:

James McCann

Real-Time Gradient-Domain Painting, 12:00 p.m., 5409 Wean Hall

Putting it all together

Compositing images

- Have a clever blending function
  - Feathering
  - Center-weighted
  - 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