
Blending and Compositing

15-463: Rendering and Image
Processing
Alexei Efros

Today

Image Compositing
Alpha Blending
Feathering
Pyramid Blending
Gradient Blending
Seam Finding

Reading:

Szeliski Tutorial, Section 6

For specific algorithms:

- Burt & Adelson
- Ask me for further references

Blending the mosaic



An example of image compositing:
the art (and sometime science) of
combining images together...

Image Compositing



Compositing Procedure

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

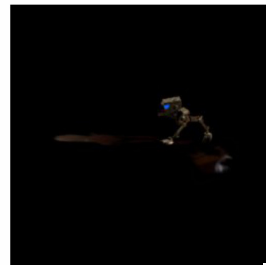


2. Blend them into the composite (in the right order)

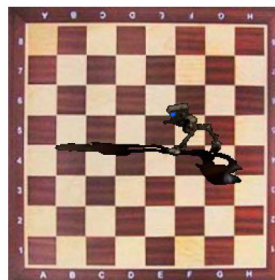


Composite by
David Dewey

Just replacing pixels rarely works

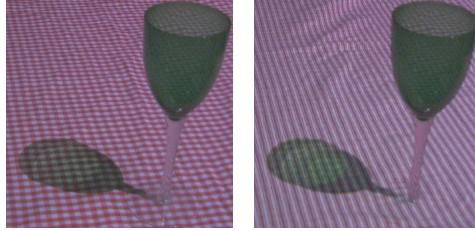


Binary
mask



Problems: boundaries & transparency (shadows)

Two Problems:



Semi-transparent objects



Pixels too large

Solution: alpha channel

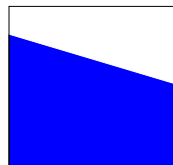
Add one more channel:

- Image(R,G,B,alpha)

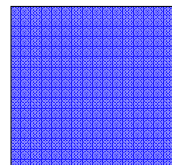
Encodes transparency (or pixel coverage):

- Alpha = 1: opaque object (complete coverage)
- Alpha = 0: transparent object (no coverage)
- $0 < \text{Alpha} < 1$: semi-transparent (partial coverage)

Example: alpha = 0.3



Partial coverage



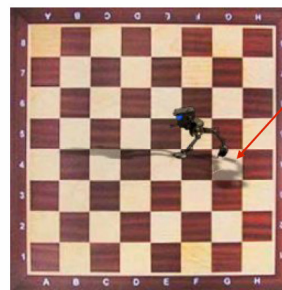
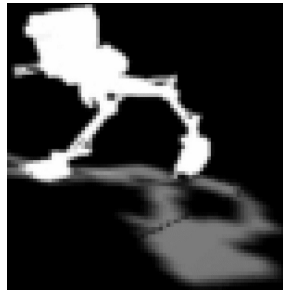
or semi-transparency

Alpha Blending



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

alpha
mask



shadow

Multiple Alpha Blending

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

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

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

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

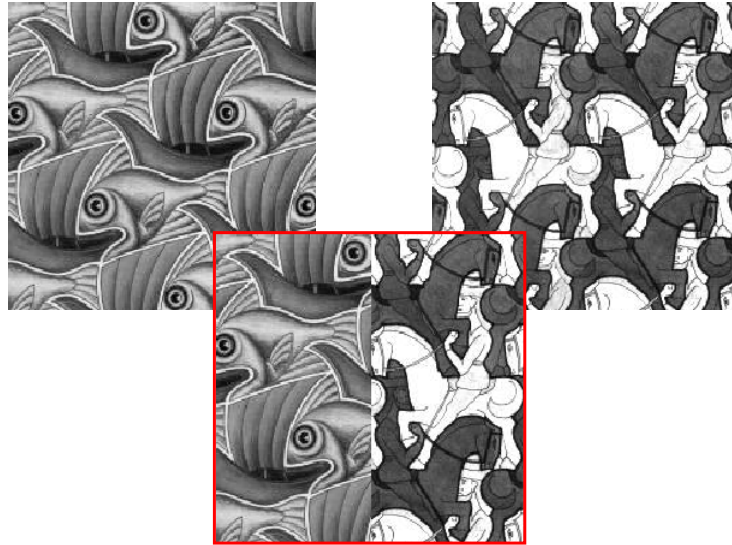
Note: sometimes alpha is
premultiplied: $\text{im}(\alpha R, \alpha G, \alpha B, \alpha)$:

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

(same for alpha!)

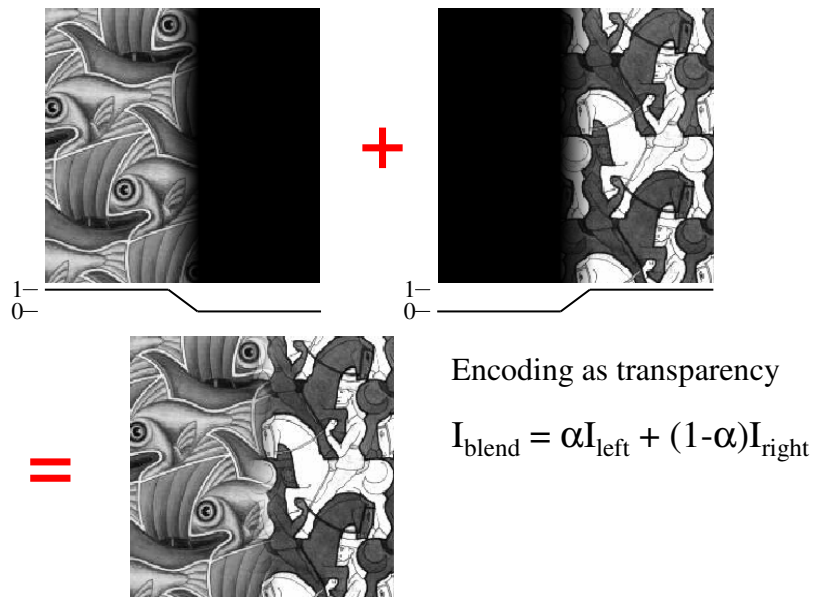


Alpha Hacking...



No physical interpretation, but it smooths the seams

Feathering

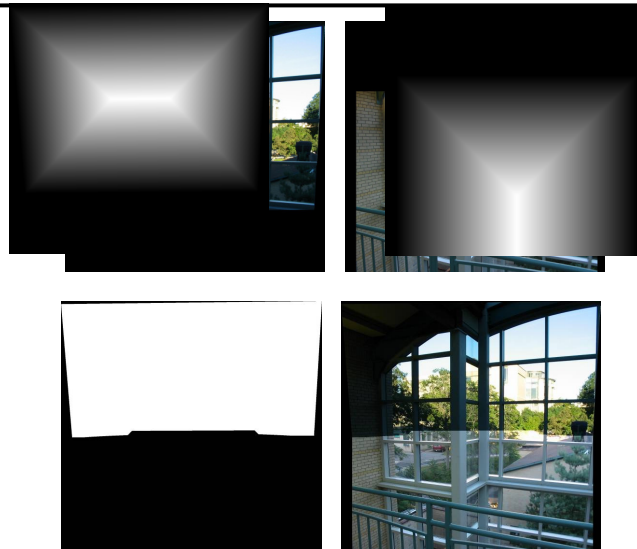


Setting alpha: simple averaging



Alpha = .5 in overlap region

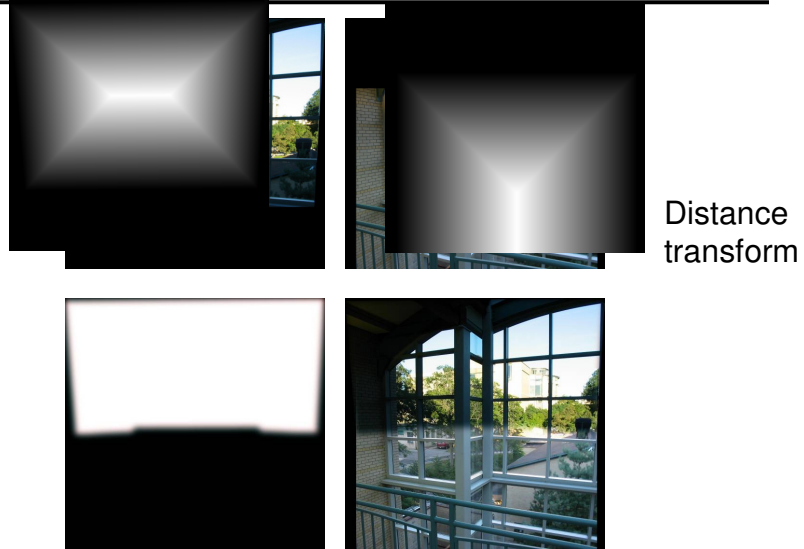
Setting alpha: center seam



Distance
transform

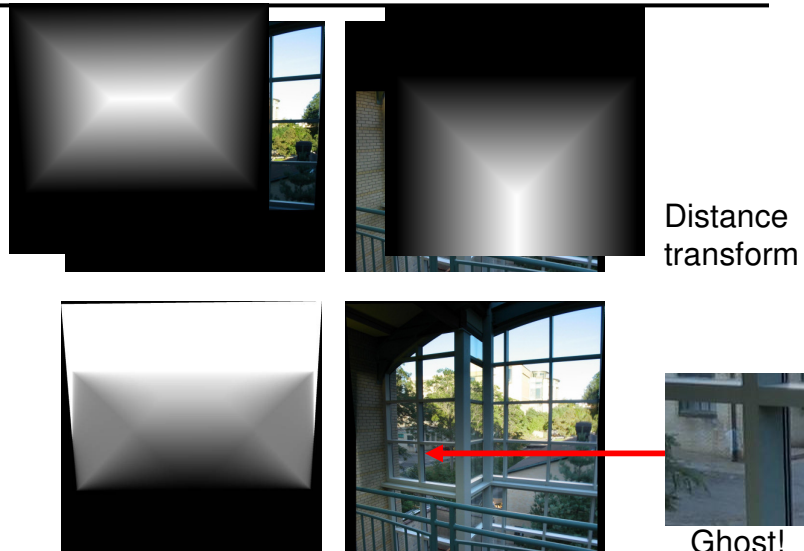
Alpha = logical(dtrans1 > dtrans2)

Setting alpha: blurred seam



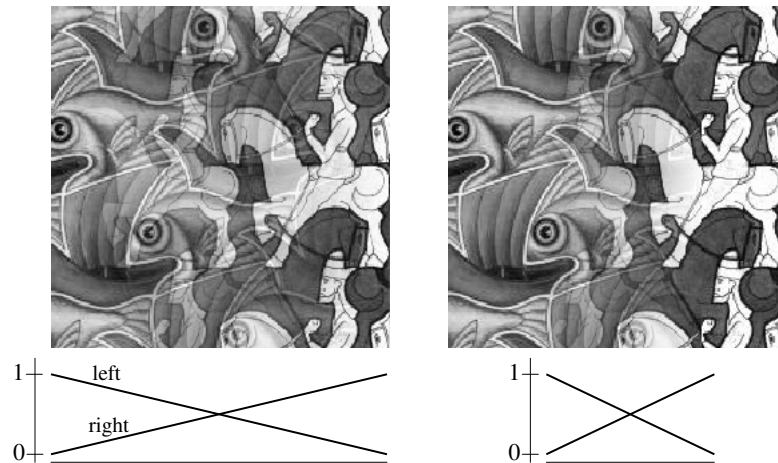
Alpha = blurred

Setting alpha: center weighting

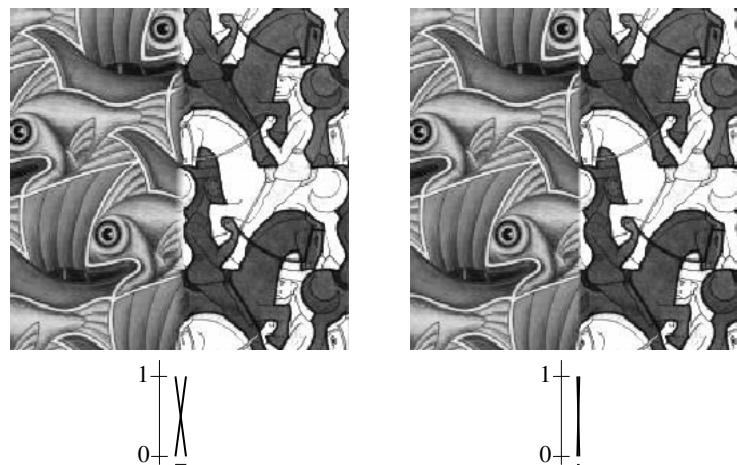


$\text{Alpha} = \text{dtrans1} / (\text{dtrans1} + \text{dtrans2})$

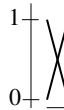
Affect of Window Size



Affect of Window Size



Good Window Size



“Optimal” Window: smooth but not ghosted

What is the Optimal Window?

To avoid seams

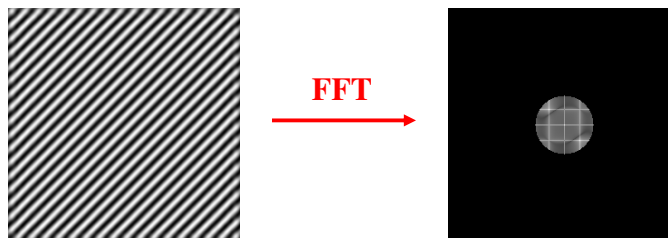
- window = size of largest prominent feature

To avoid ghosting

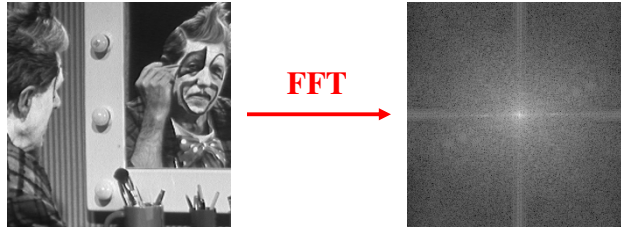
- window $\leq 2 \times$ size of smallest prominent feature

Natural to cast this in the *Fourier domain*

- largest frequency $\leq 2 \times$ size of smallest frequency
- image frequency content should occupy one “octave” (power of two)



What if the Frequency Spread is Wide



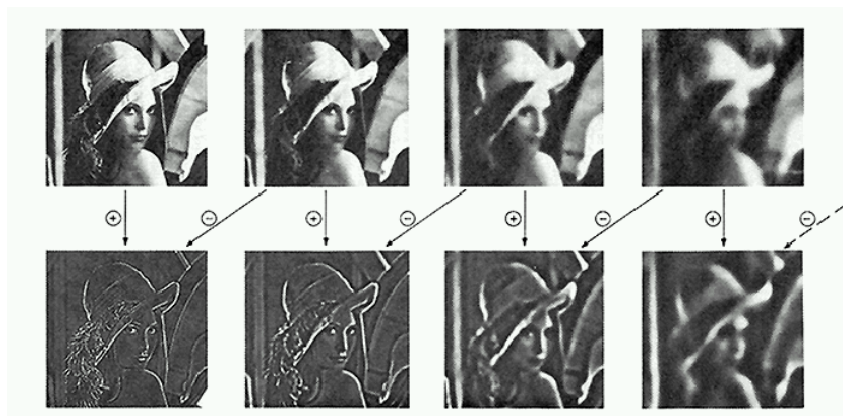
Idea (Burt and Adelson)

- Compute $F_{\text{left}} = \text{FFT}(I_{\text{left}})$, $F_{\text{right}} = \text{FFT}(I_{\text{right}})$
- Decompose Fourier image into octaves (bands)
 - $F_{\text{left}} = F_{\text{left}}^1 + F_{\text{left}}^2 + \dots$
- 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*

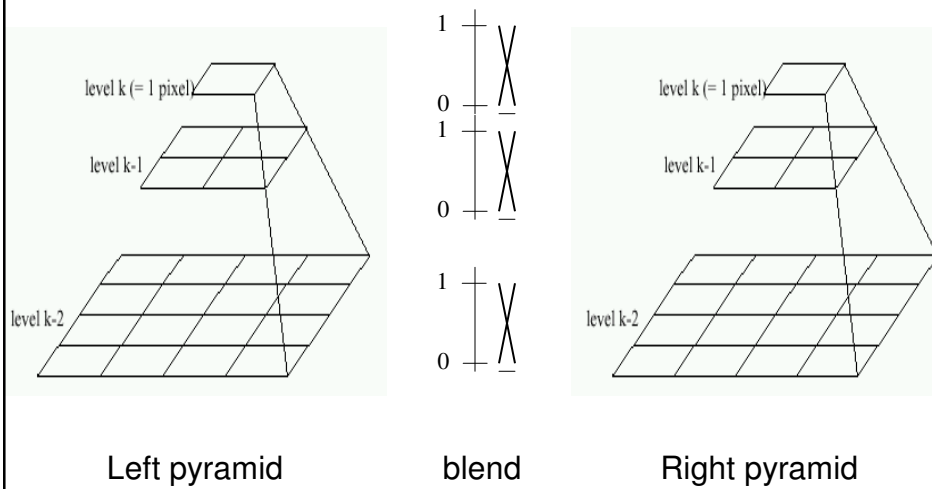
Octaves in the Spatial Domain

Lowpass Images

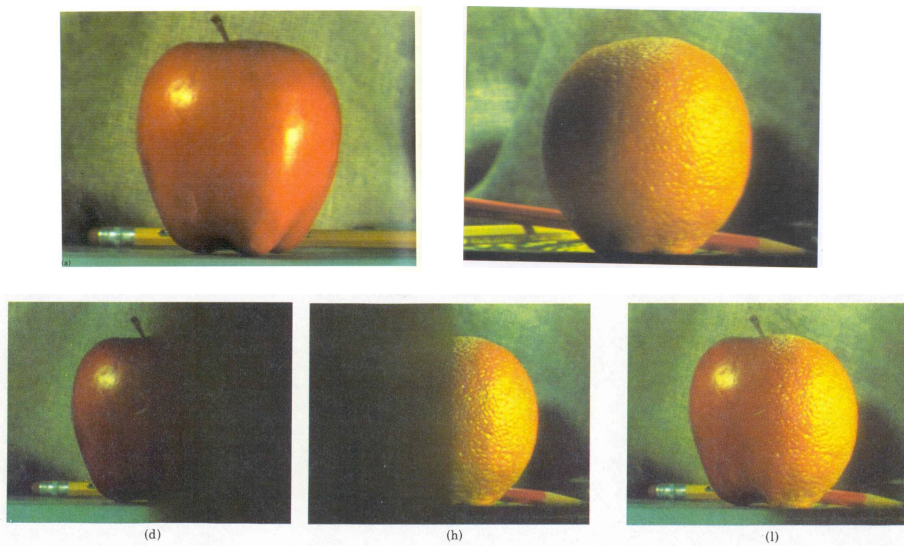


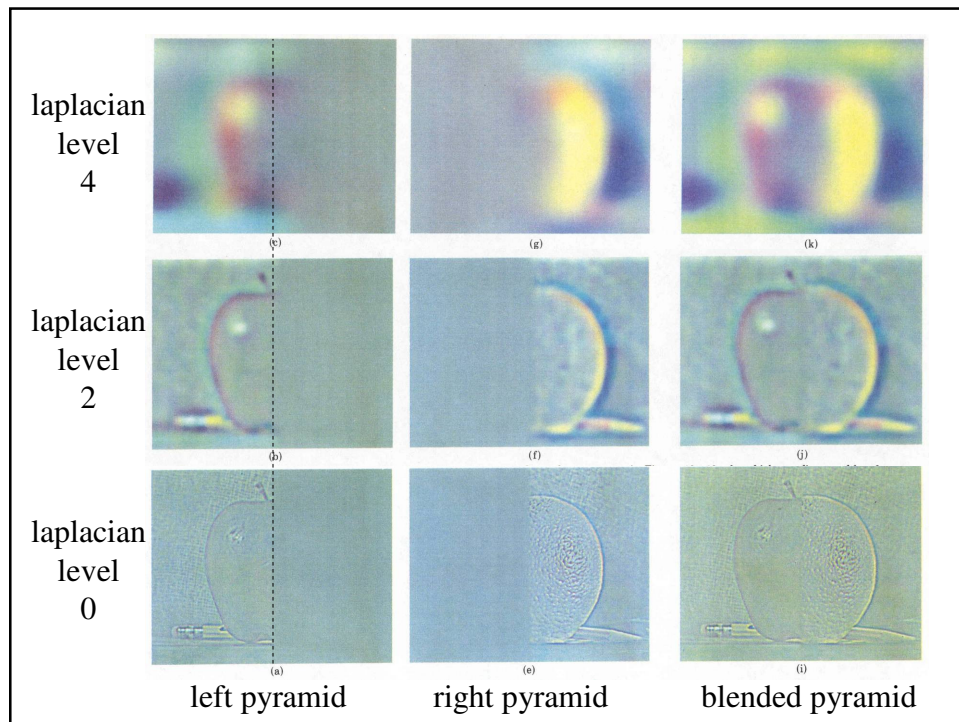
Bandpass Images

Pyramid Blending



Pyramid Blending





Laplacian Pyramid: Blending

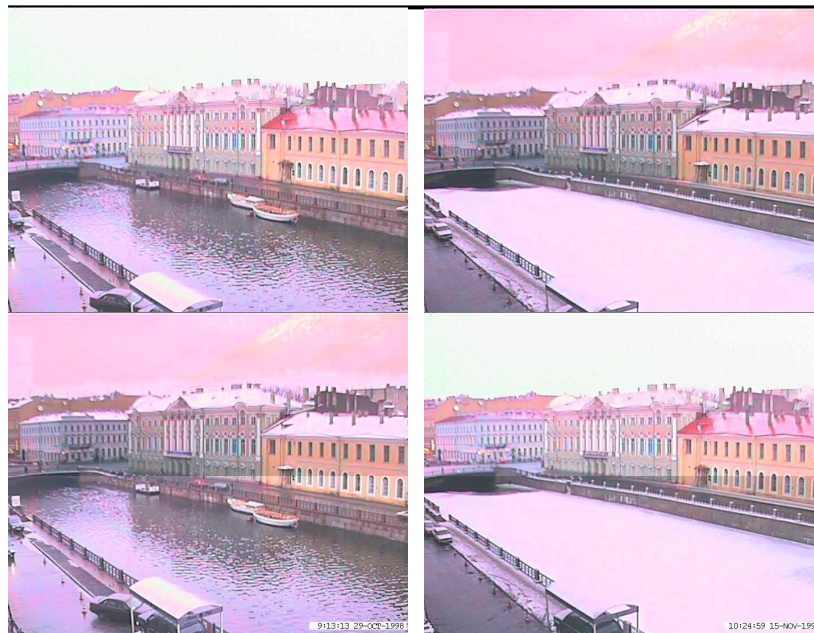
General Approach:

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



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 ($\lambda > 2$ pixels)



High frequency ($\lambda < 2$ pixels)

Linear Blending



2-band Blending



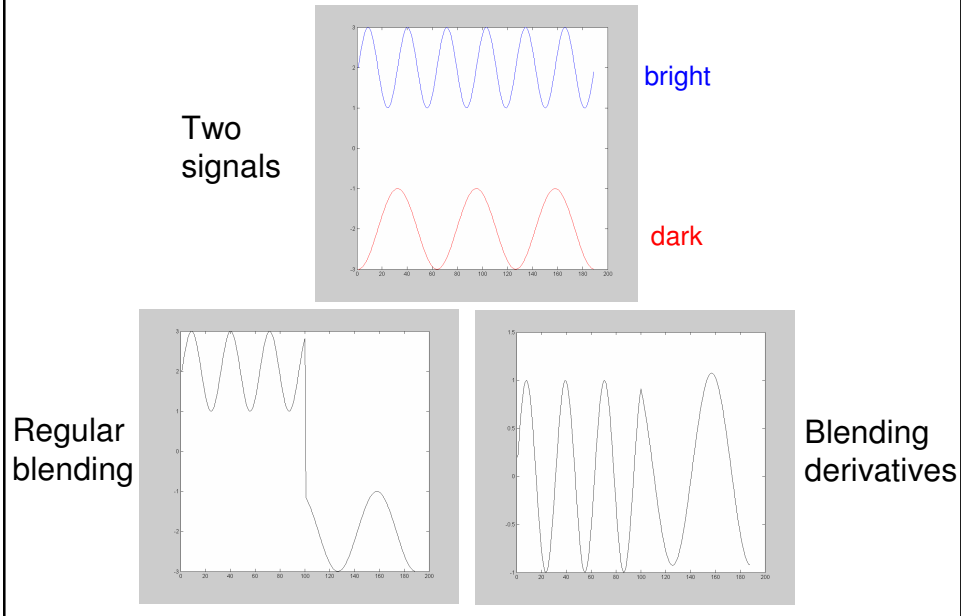
Gradient Domain

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

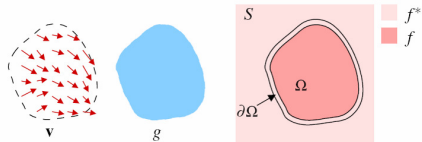
Let us now look at 1st derivatives (gradients):

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

Gradient Domain blending (1D)



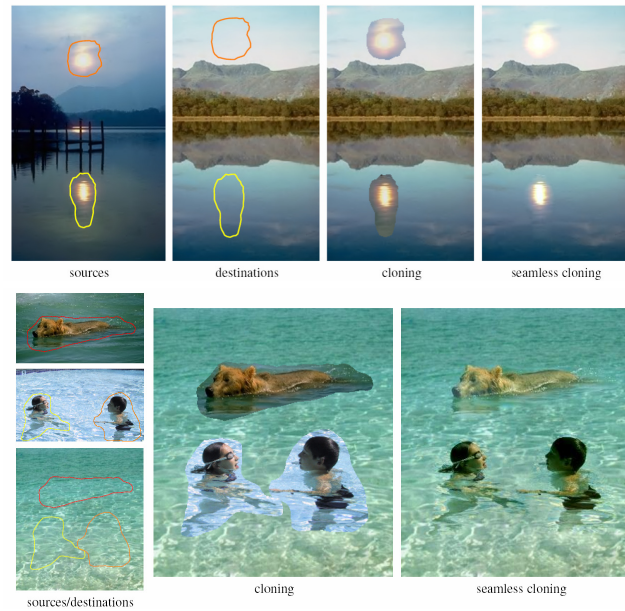
Gradient Domain Blending (2D)



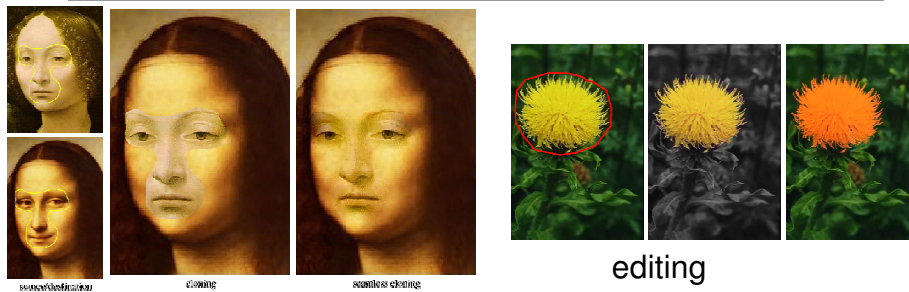
Trickier in 2D:

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

Perez et al., 2003



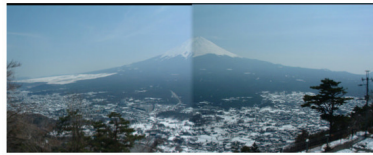
Perez et al., 2003



Limitations:

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

Mosaic results: Levin et al, 2004



Pyramid blending



Feathering



Pyramid blending on the gradients



GIST1

Don't blend, CUT!



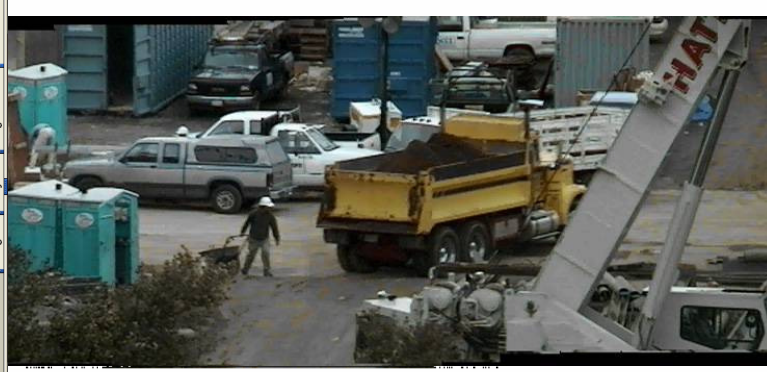
Moving objects become ghosts

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

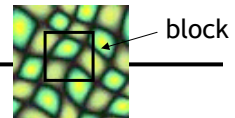
Davis, 1998

Segment the mosaic

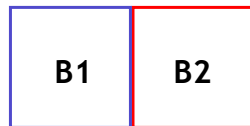
- Single source image per segment
- Avoid artifacts along boundaries
 - Dijkstra's algorithm



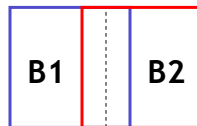
Efros & Freeman, 2001



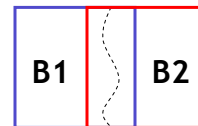
Input texture



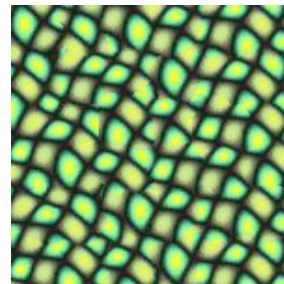
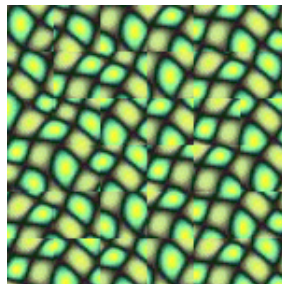
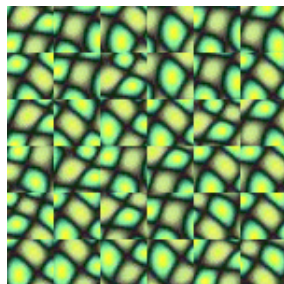
Random placement
of blocks



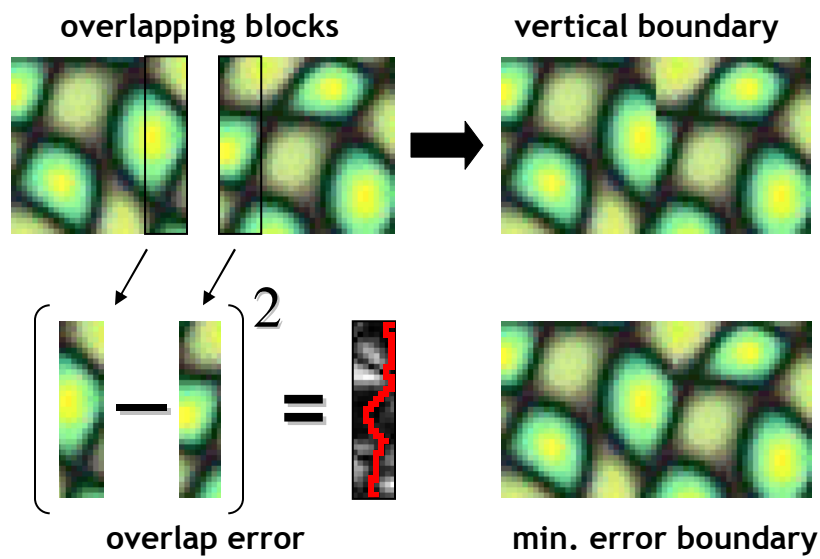
Neighboring blocks
constrained by overlap



Minimal error
boundary cut



Minimal error boundary



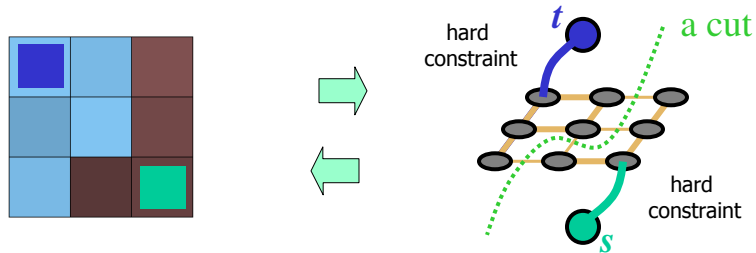
Graphcuts

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)



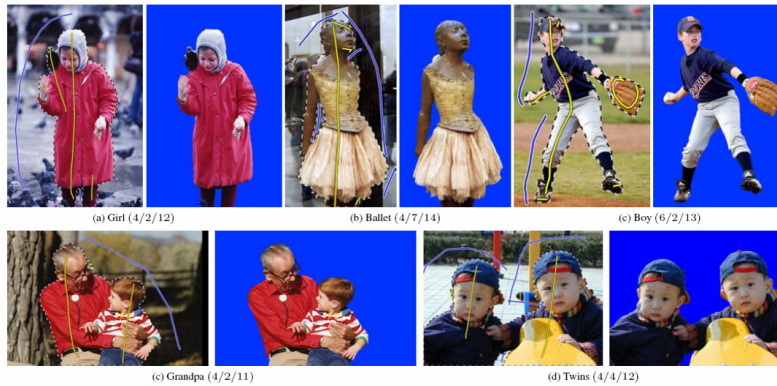
Minimum cost cut can be computed in polynomial time
(max-flow/min-cut algorithms)

Kwatra et al, 2003



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

Lazy Snapping (today's speaker)



Interactive segmentation using graphcuts

Putting it all together

Compositing images/mosaics

- 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

