Image Composition

COS 526
Princeton University

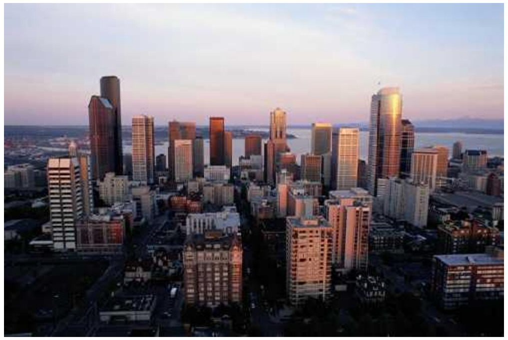
Modeled after lecture by Alexei Efros.
Slides by Efros, Durand, Freeman, Hays, Fergus, Lazebnik, Agarwala, Shamir, and Perez.

Image Composition

Jurassic Park

Image Blending

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



Joe Marquette / AP file

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

Composite by
David Dewey

Slide credit: A. Efros

Image Composition

Laplacian pyramid blending

Poisson composition

Graphcut seams

Texture synthesis

Image Composition

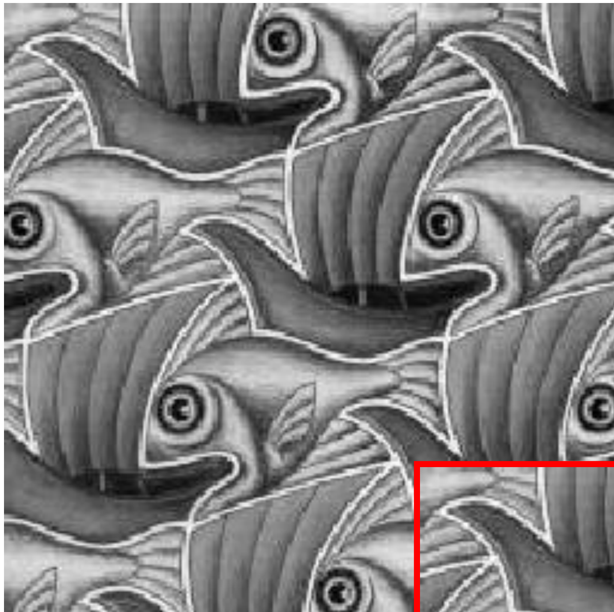
Laplacian pyramid blending

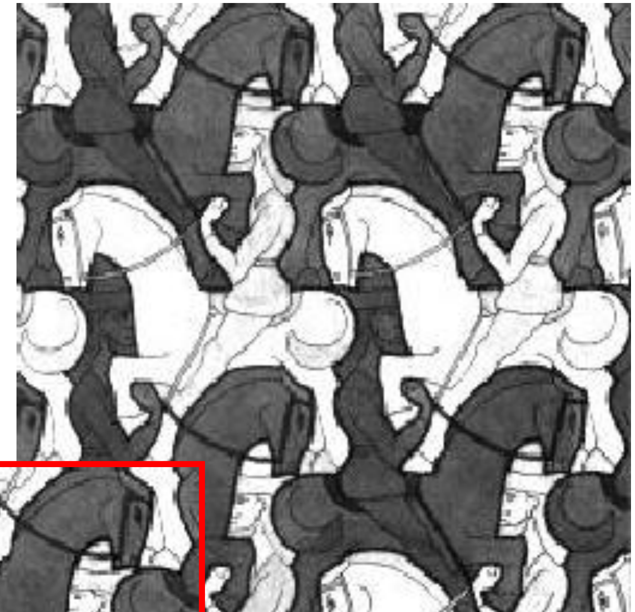
Poisson composition

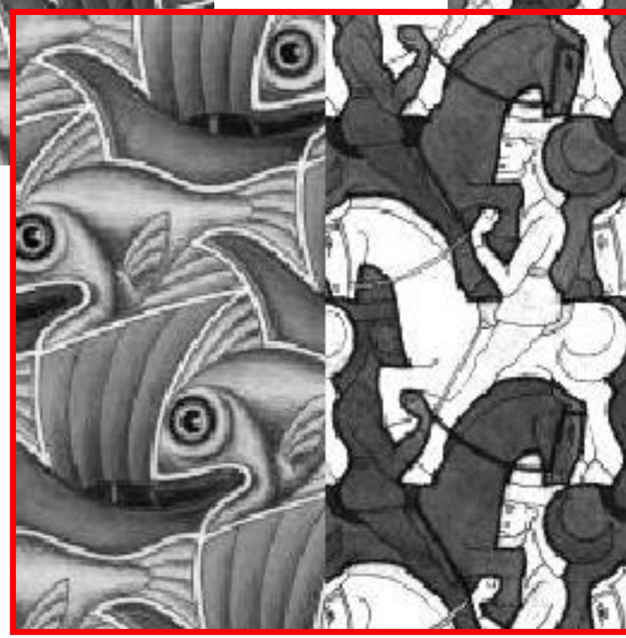
Graphcut seams

Texture synthesis

Image Blending



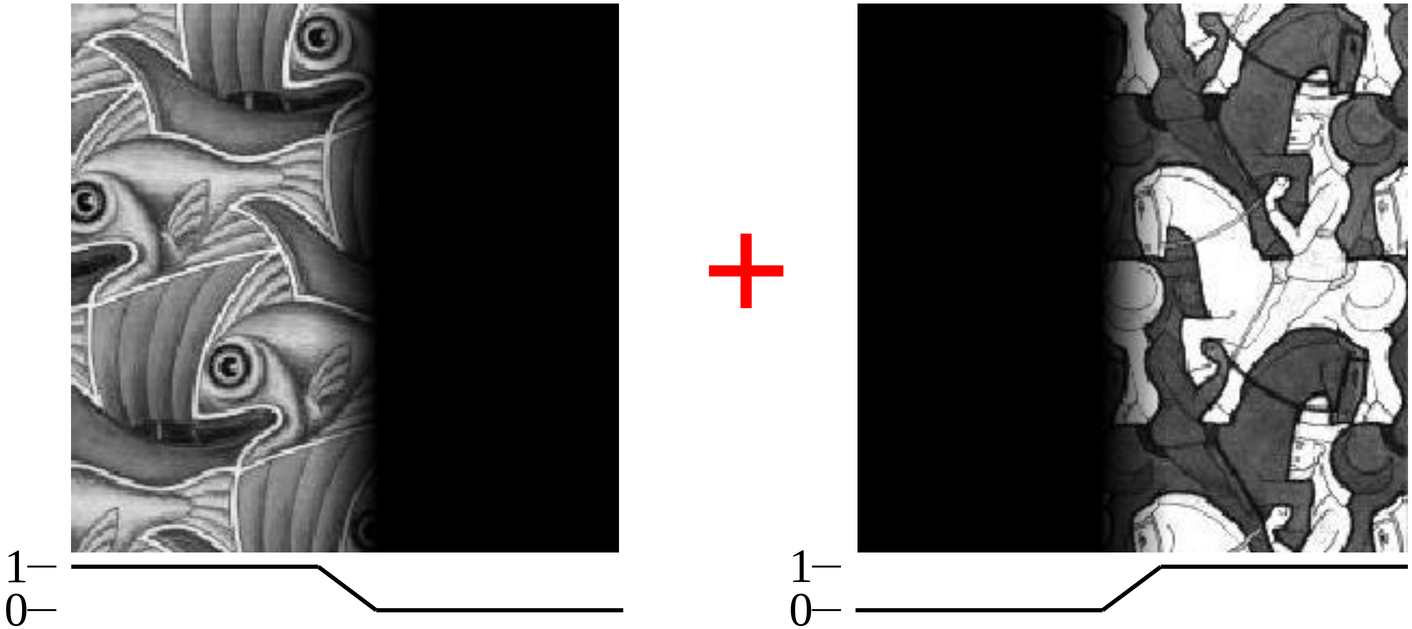


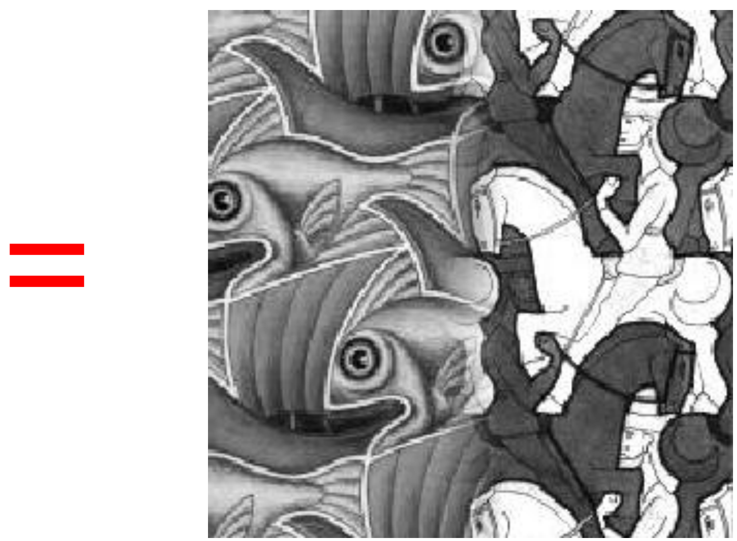


Without Blending

Slide credit: A. Efros

Alpha Blending / Feathering

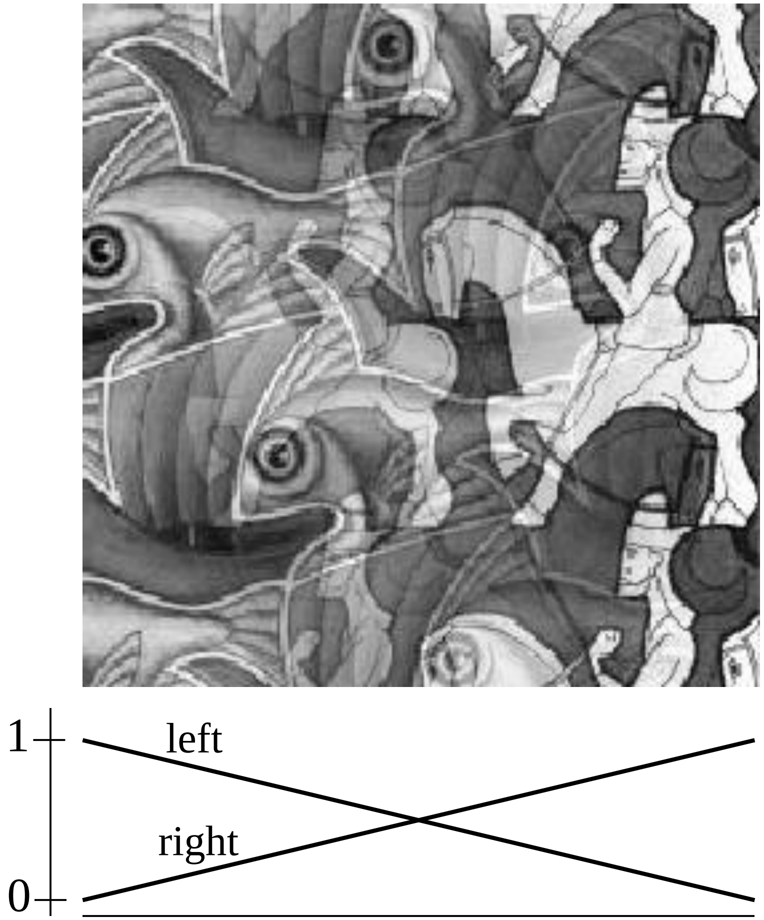


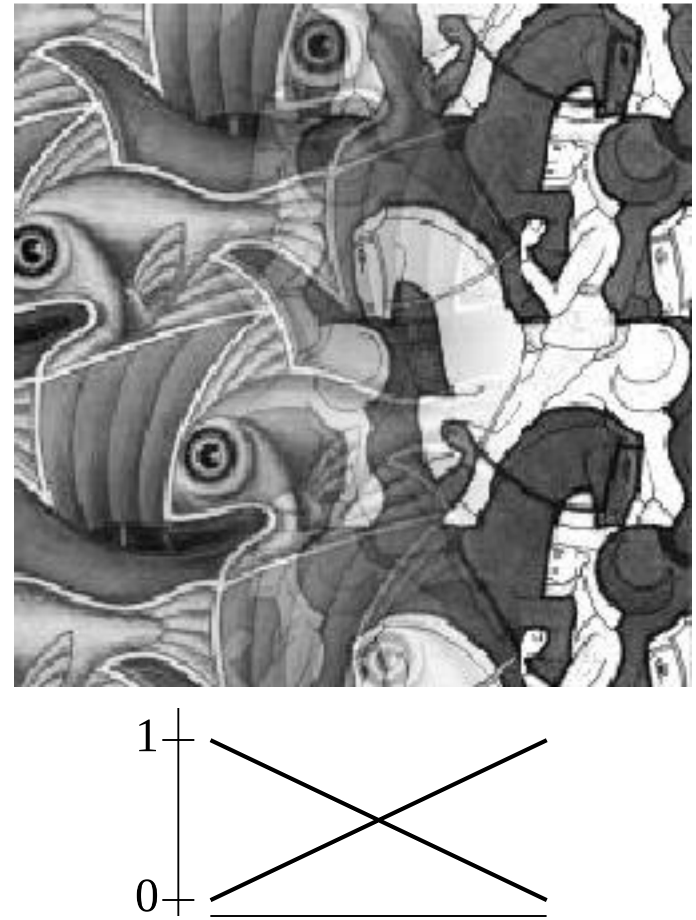


$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

Slide credit: A. Efros

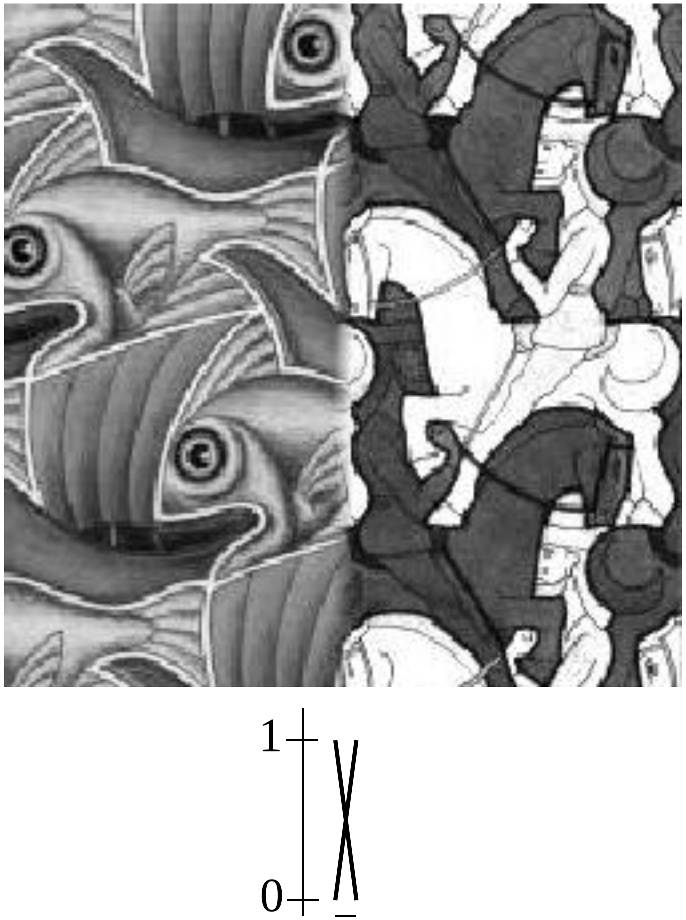
Affect of Window Size

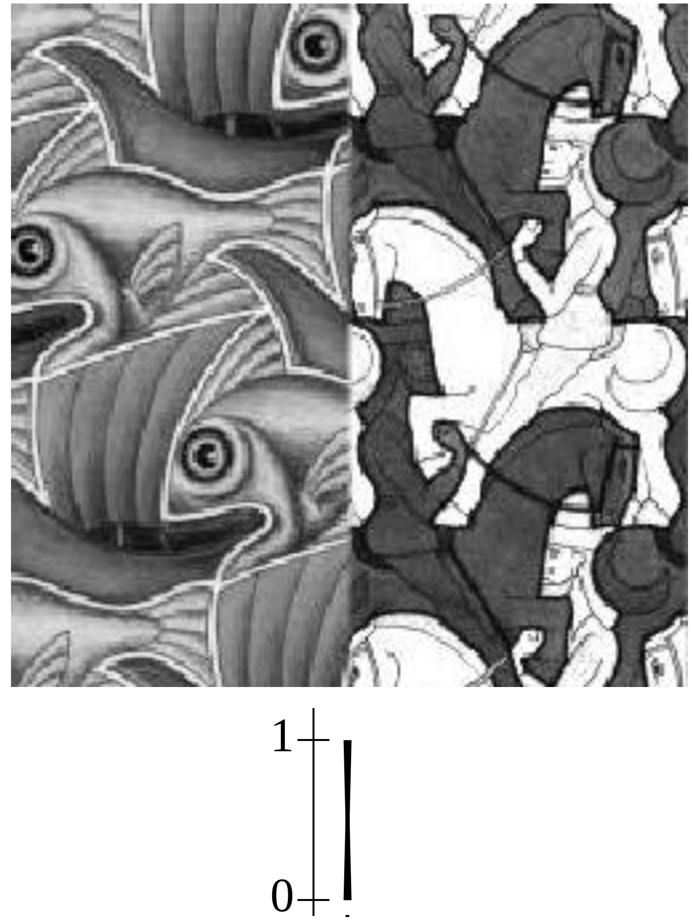




Slide credit: A. Efros

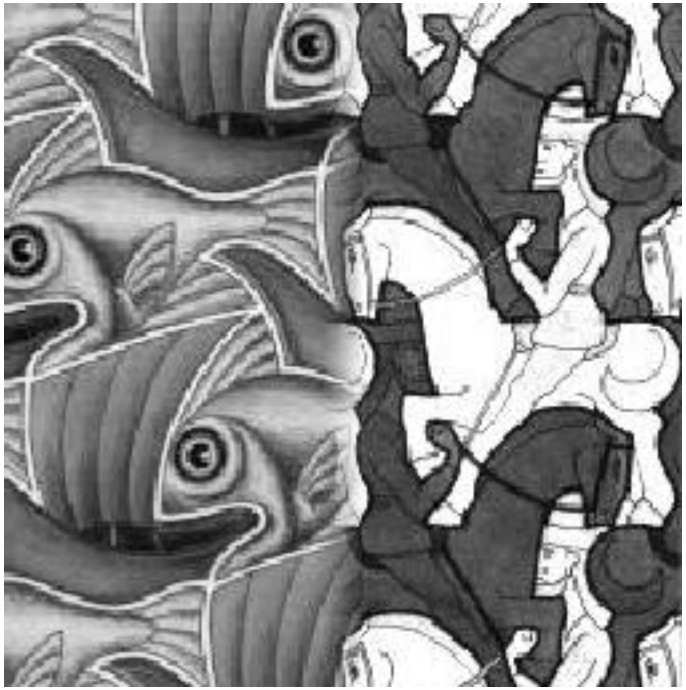
Affect of Window Size

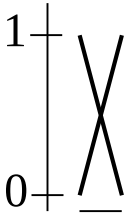




Slide credit: A. Efros

Good Window Size





"Optimal" Window: smooth but not ghosted

Slide credit: A. Efros

What is the Optimal Window?

To avoid seams

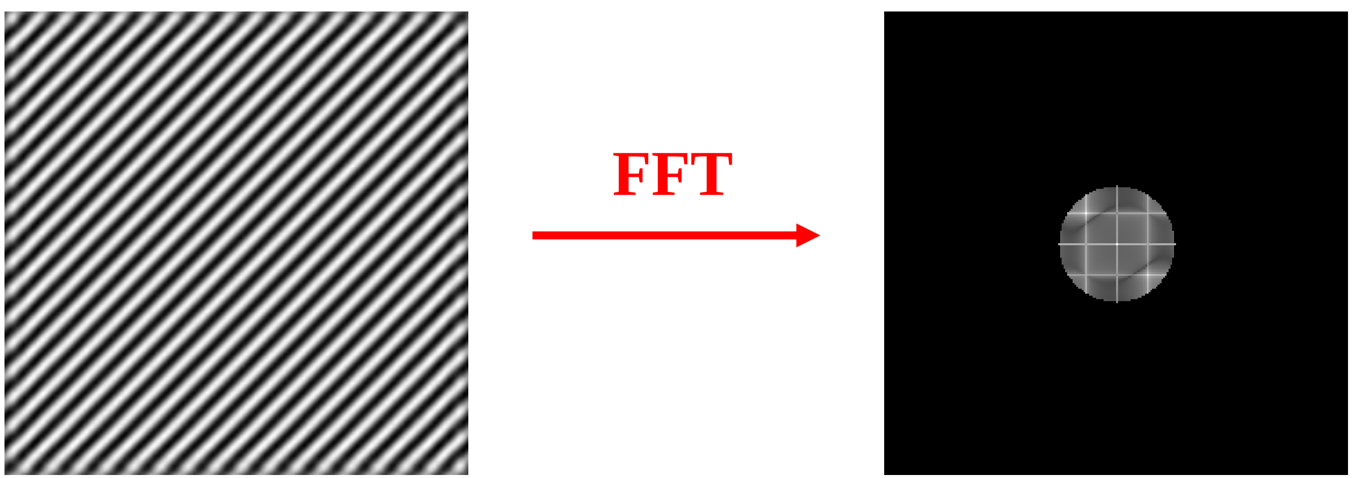
- window = size of largest prominent feature

To avoid ghosting

- window <= 2*size of smallest prominent feature

Natural to cast this in the *Fourier domain*

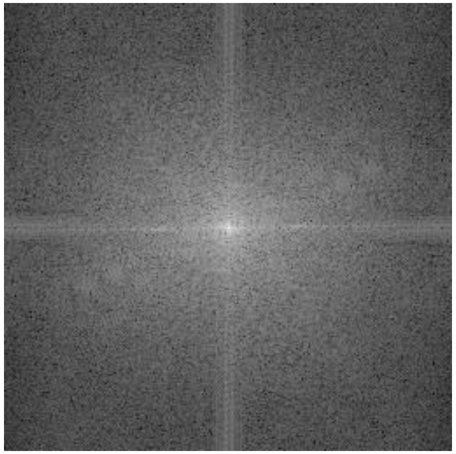
- largest frequency <= 2*size of smallest frequency
- image frequency content should occupy one “octave” (power of two)



Slide credit: A. Efros

What if the Frequency Spread is Wide

FFT



Idea (Burt and Adelson)

- Different window sizes for different frequencies

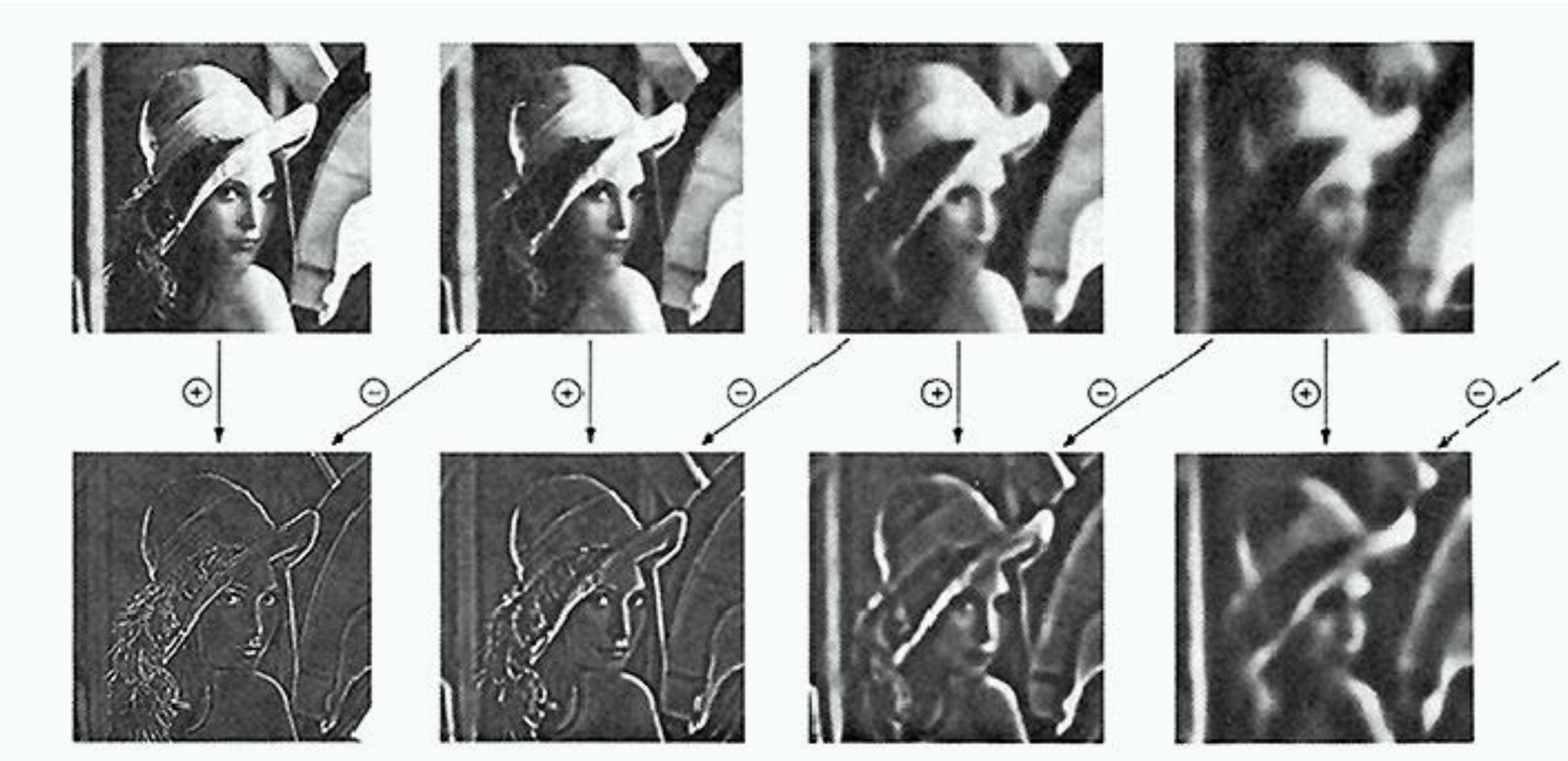
Method

- Decompose image into octaves (frequency bands)
- Feather each octave with appropriate window size
- Sum feathered octave images to reconstruct blended image

Slide credit: A. Efros

Laplacian Pyramid

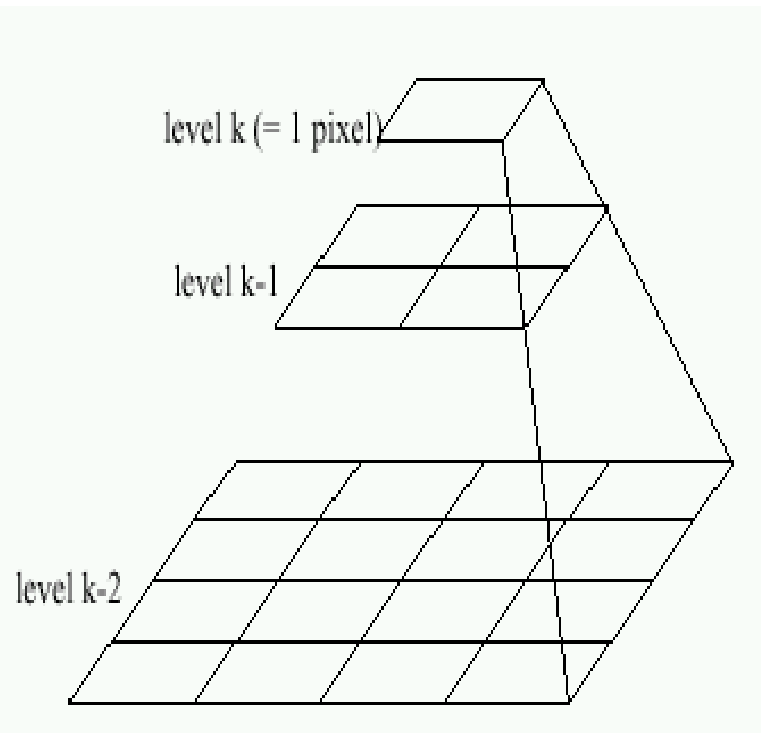
Lowpass Images

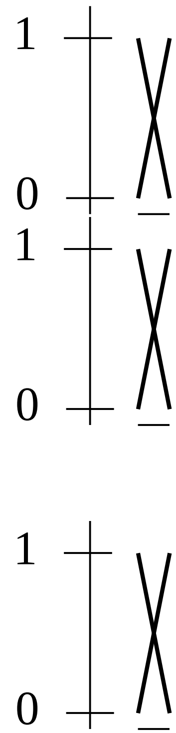


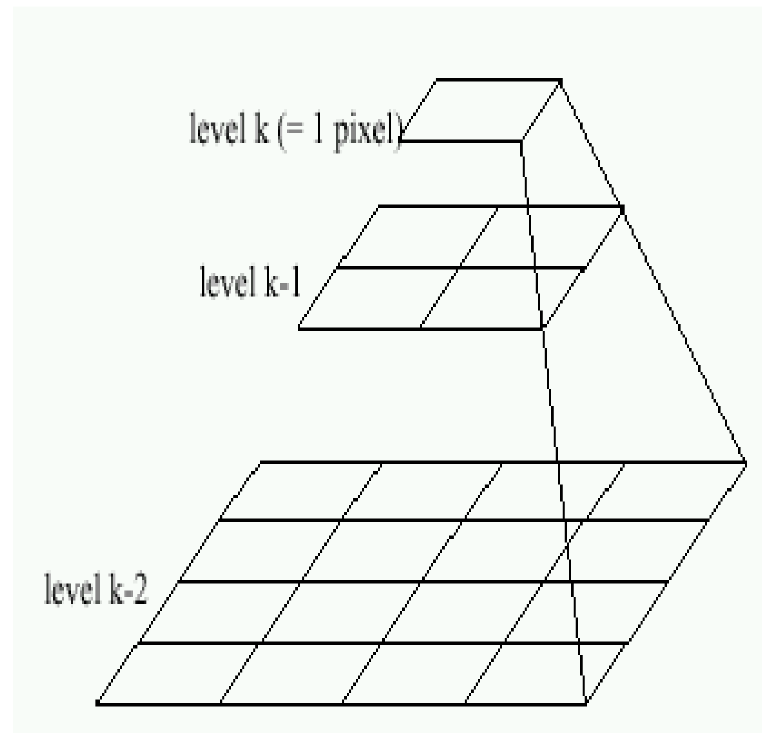
Bandpass Images

Slide credit: A. Efros

Laplacian Pyramid Blending







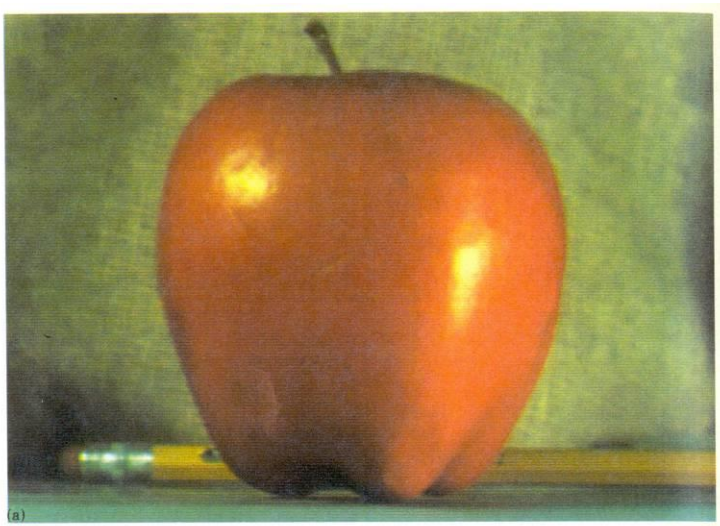
Left pyramid

blend

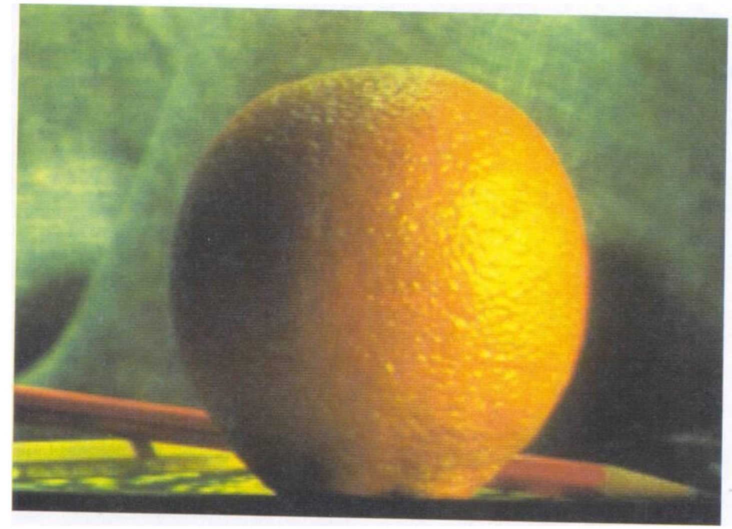
Right pyramid

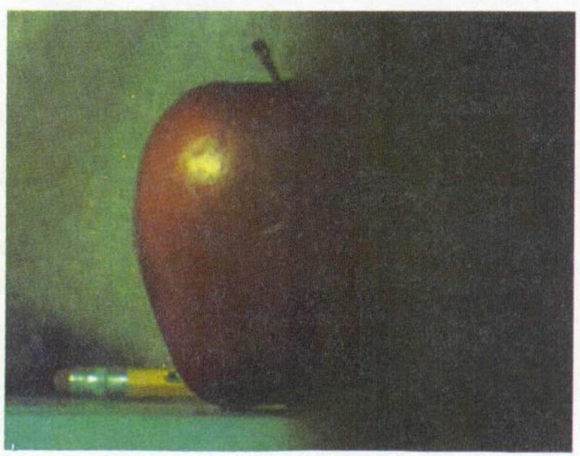
Slide credit: A. Efros

Laplacian Pyramid Blending

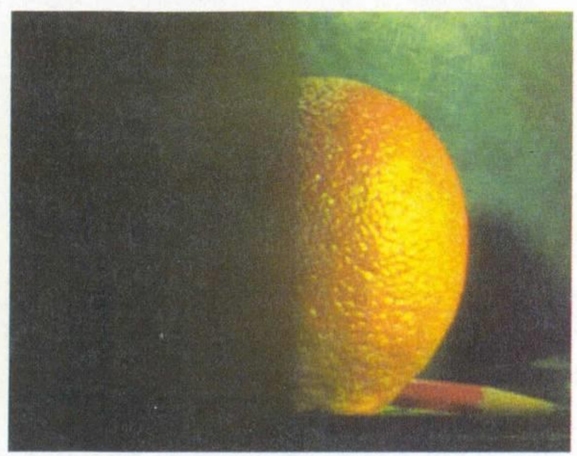


(a)

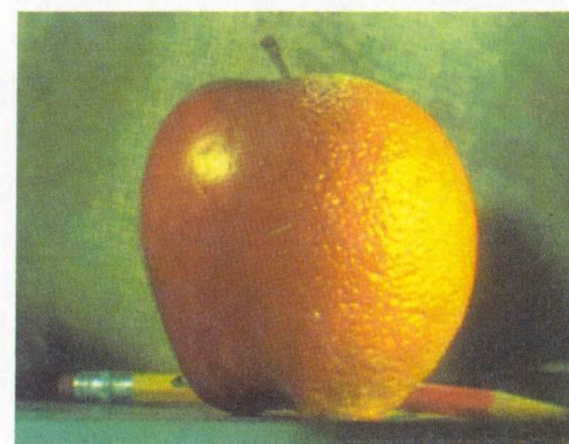




(d)

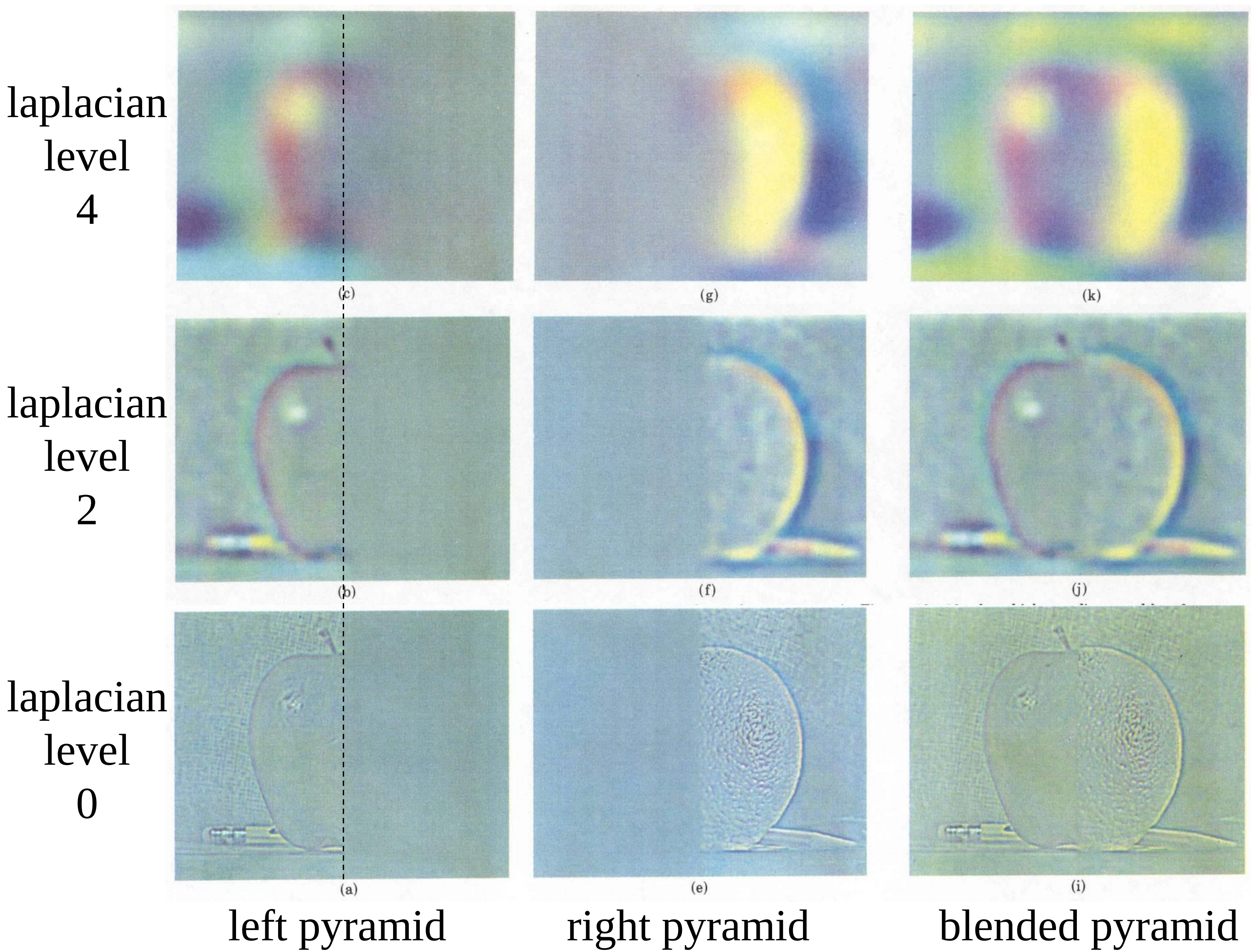


(h)



(l)

Slide credit: A. Efros

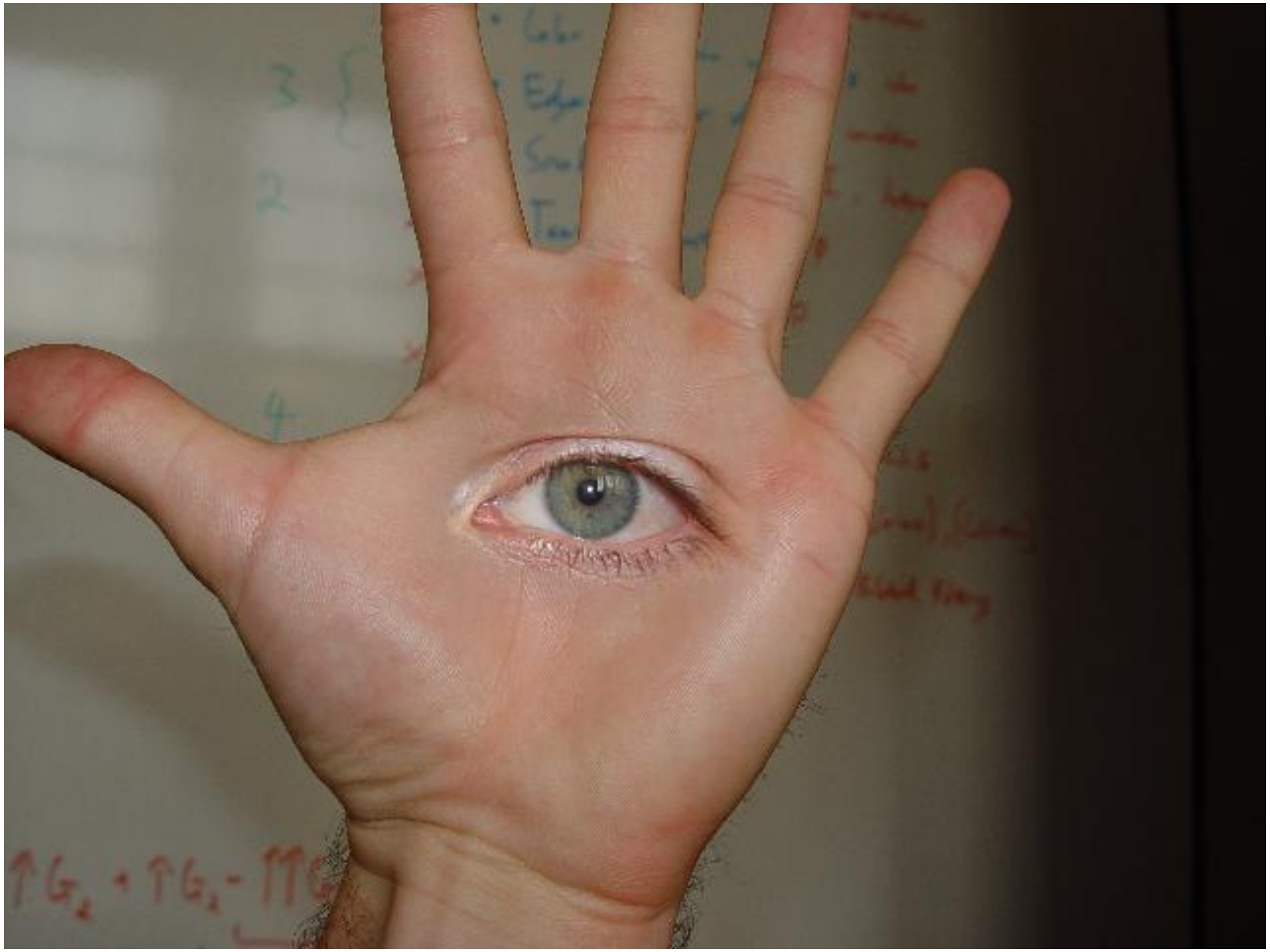


laplacian level 4
(c)
(g)
(k)
laplacian level 2
(b)
(f)
(j)
laplacian level 0
(a)
(e)
(i)
left pyramid
right pyramid
blended pyramid

Laplacian Pyramid Blending

Slide credit: A. Efros

Laplacian Pyramid Blending

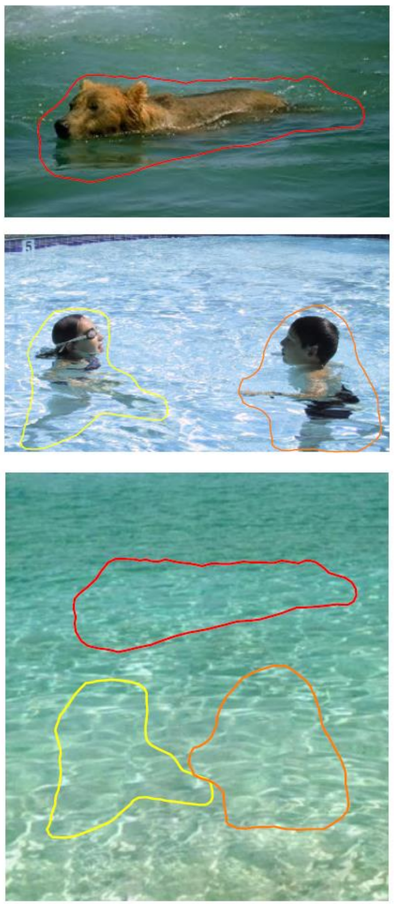


© david dmartin (Boston College)

Slide credit: A. Efros

Problem with Blending

What if colors/intensities are different?



sources/destinations

cloning

Slide credit: F. Durand

Image Composition

Laplacian pyramid blending

Poisson composition

Graphcut seams

Texture synthesis

Gradient domain image editing

Motivation:

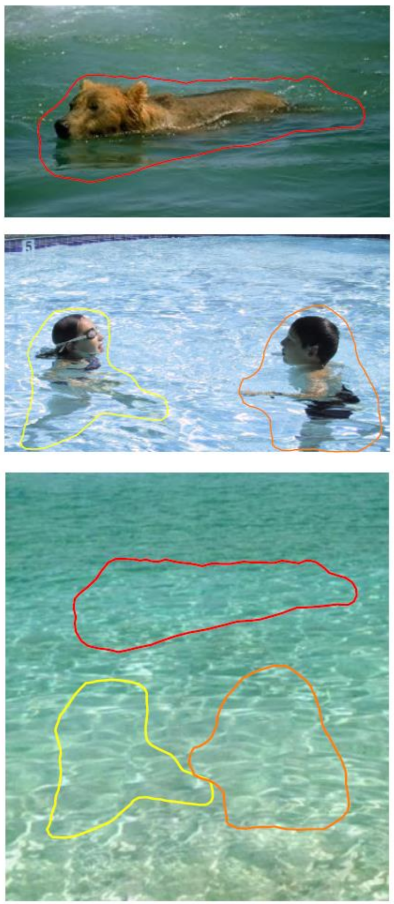
- Human visual system is very sensitive to gradient
- Gradient encode edges and local contrast quite well

Approach:

- Edit in the gradient domain
- Reconstruct image from gradient

Slide credit: F. Durand

Gradient domain image editing



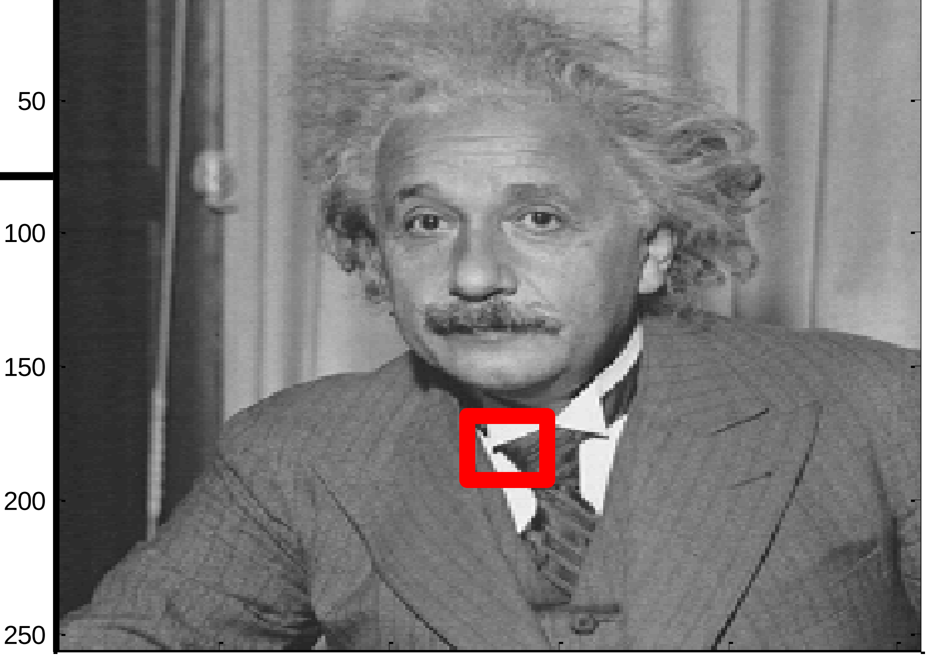
sources/destinations

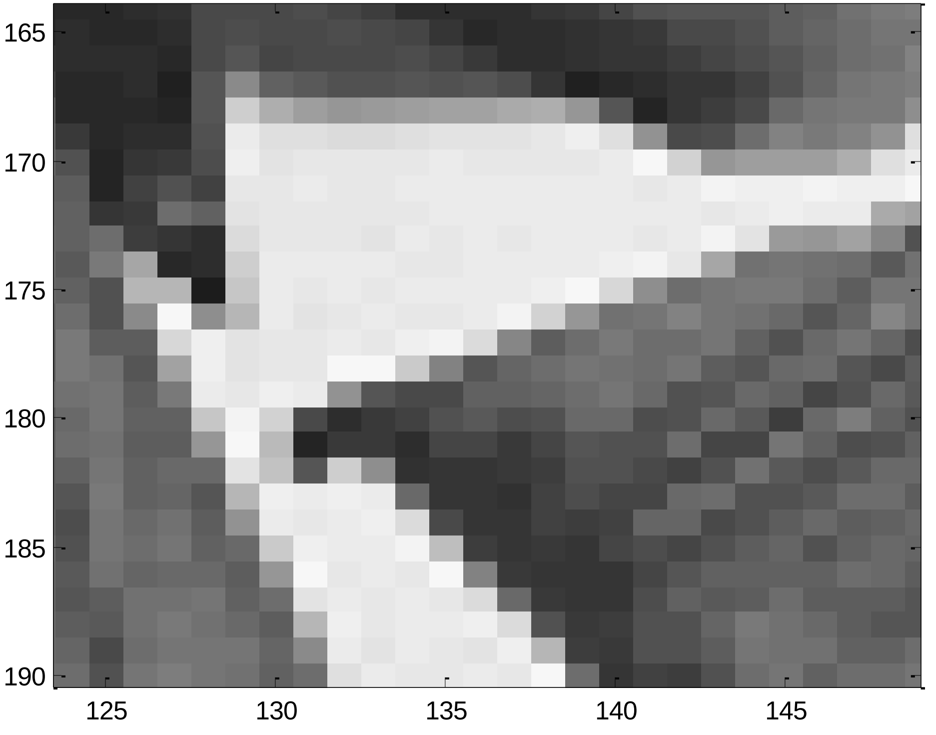
cloning

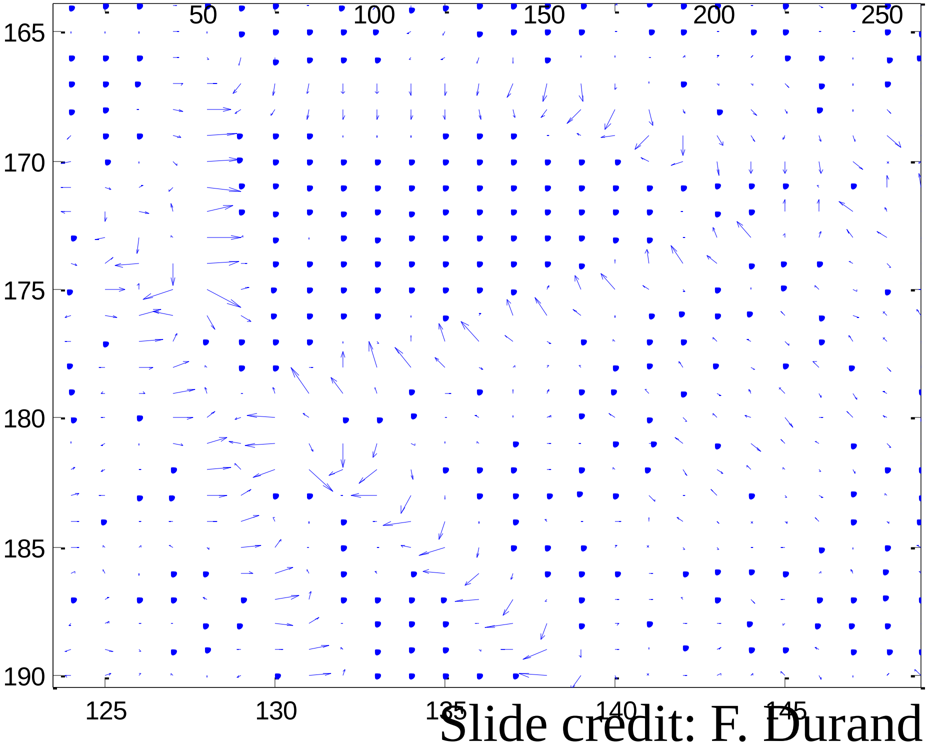
seamless cloning

Slide credit: F. Durand

Gradient domain







Slide credit: F. Durand

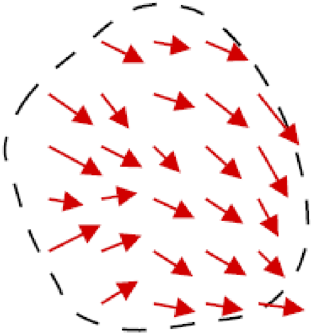
Seamless Poisson cloning

Given vector field *v* (pasted gradient), find the value of *f* in unknown region that optimizes:

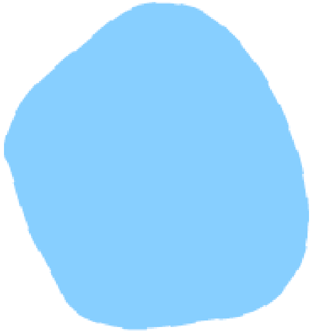
$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

Pasted gradient

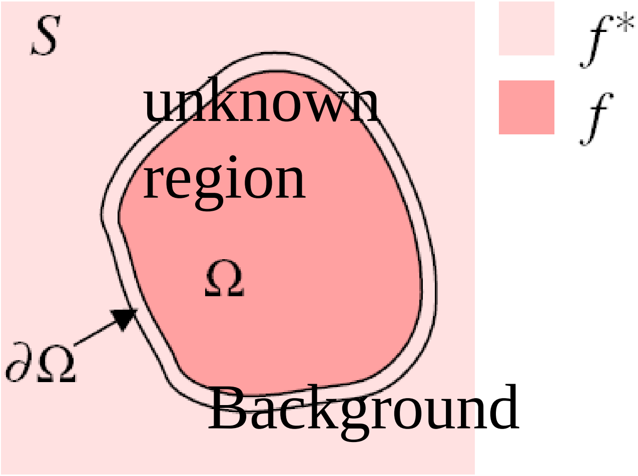
Mask



$\mathbf{v}$



g



Slide credit: F. Durand

Discrete Poisson solver

Minimize variational problem $\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega},$

Discretized gradient

$$\min_{f|_\Omega} \sum_{\langle p,q\rangle \cap \Omega \neq \emptyset} (f_p - f_q - v_{pq})^2, \text{ with } f_p = f_p^*, \text{for all } p \in \partial\Omega$$

(all pairs that are in Ω)

Discretized v: g(p)-g(q)

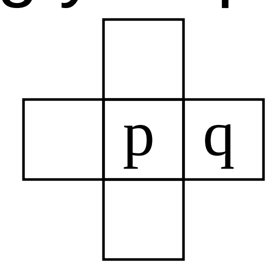
Boundary condition

Rearrange and call N_p the neighbors of p

$$\text{for all } p \in \Omega, \quad \boxed{|N_p| f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} v_{pq}}$$

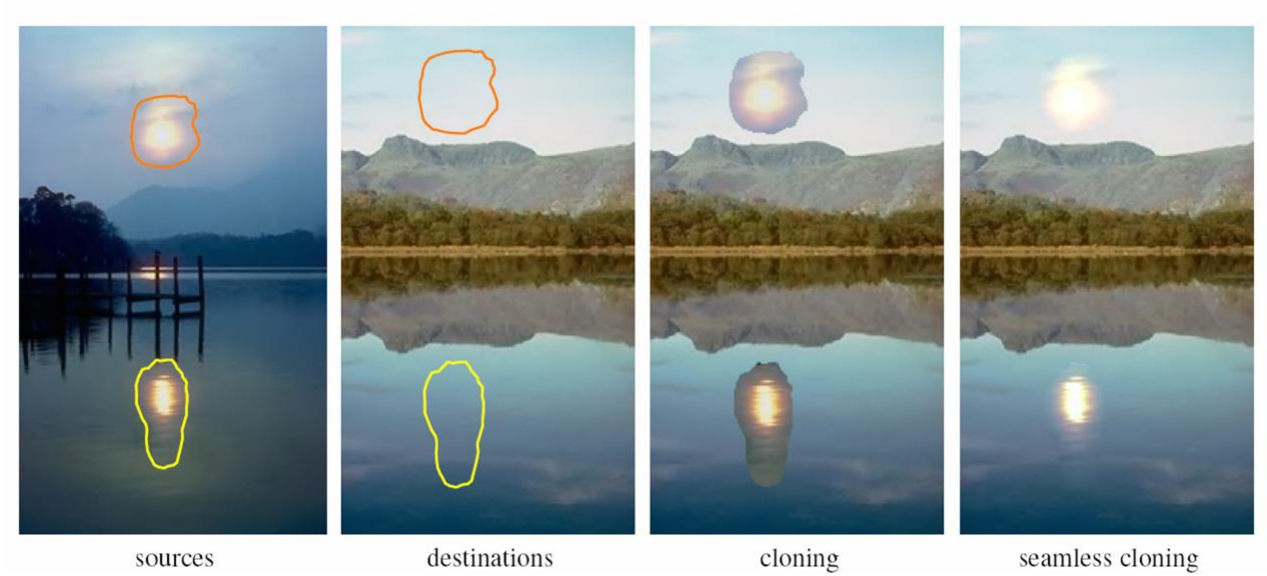
Only for boundary pixels

Big yet sparse linear system

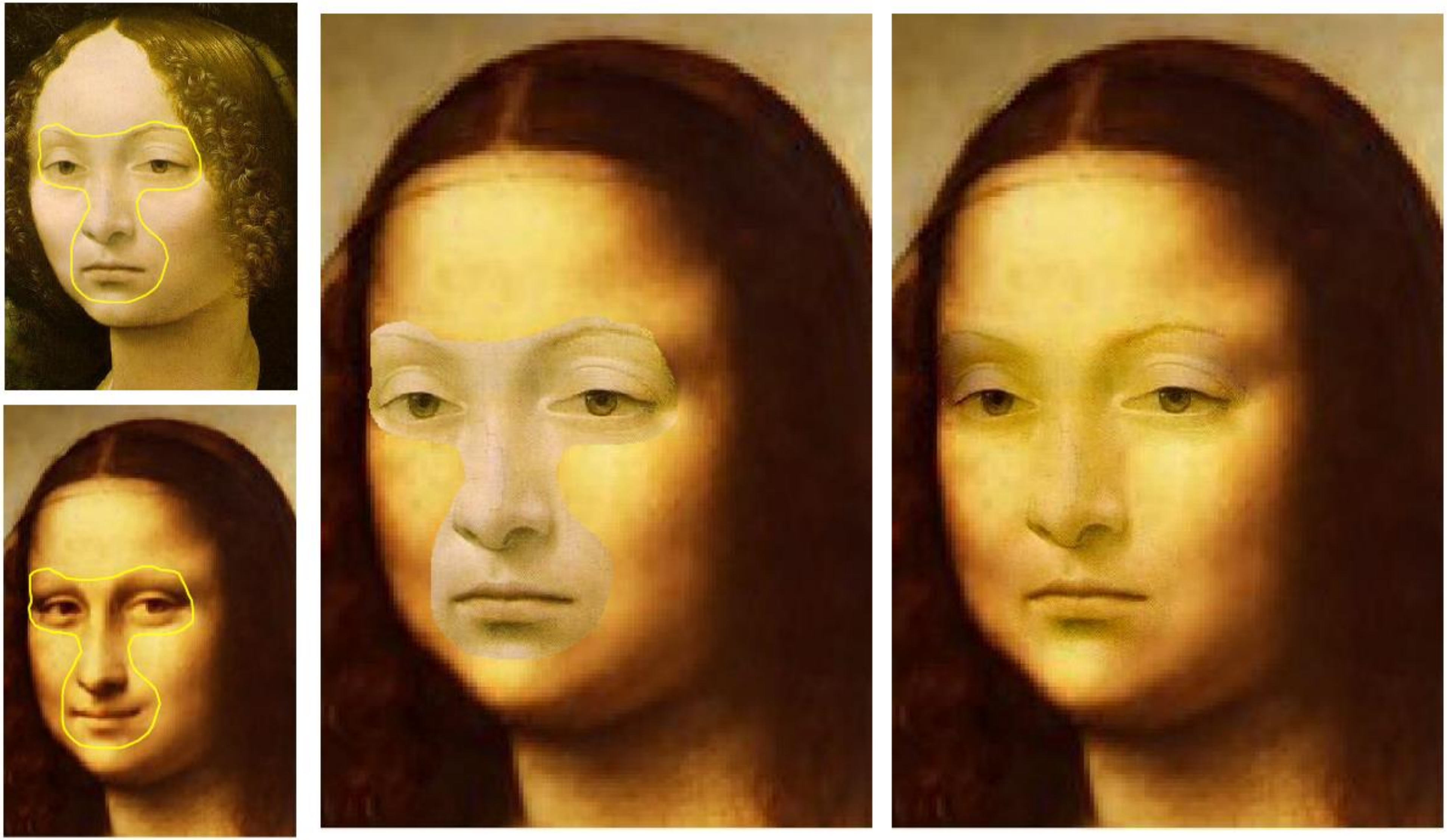


Slide credit: F. Durand

Image Composition Results



Perez et al. SIGGRAPH 03



source/destination cloning seamless cloning

Perez et al. SIGGRAPH 03

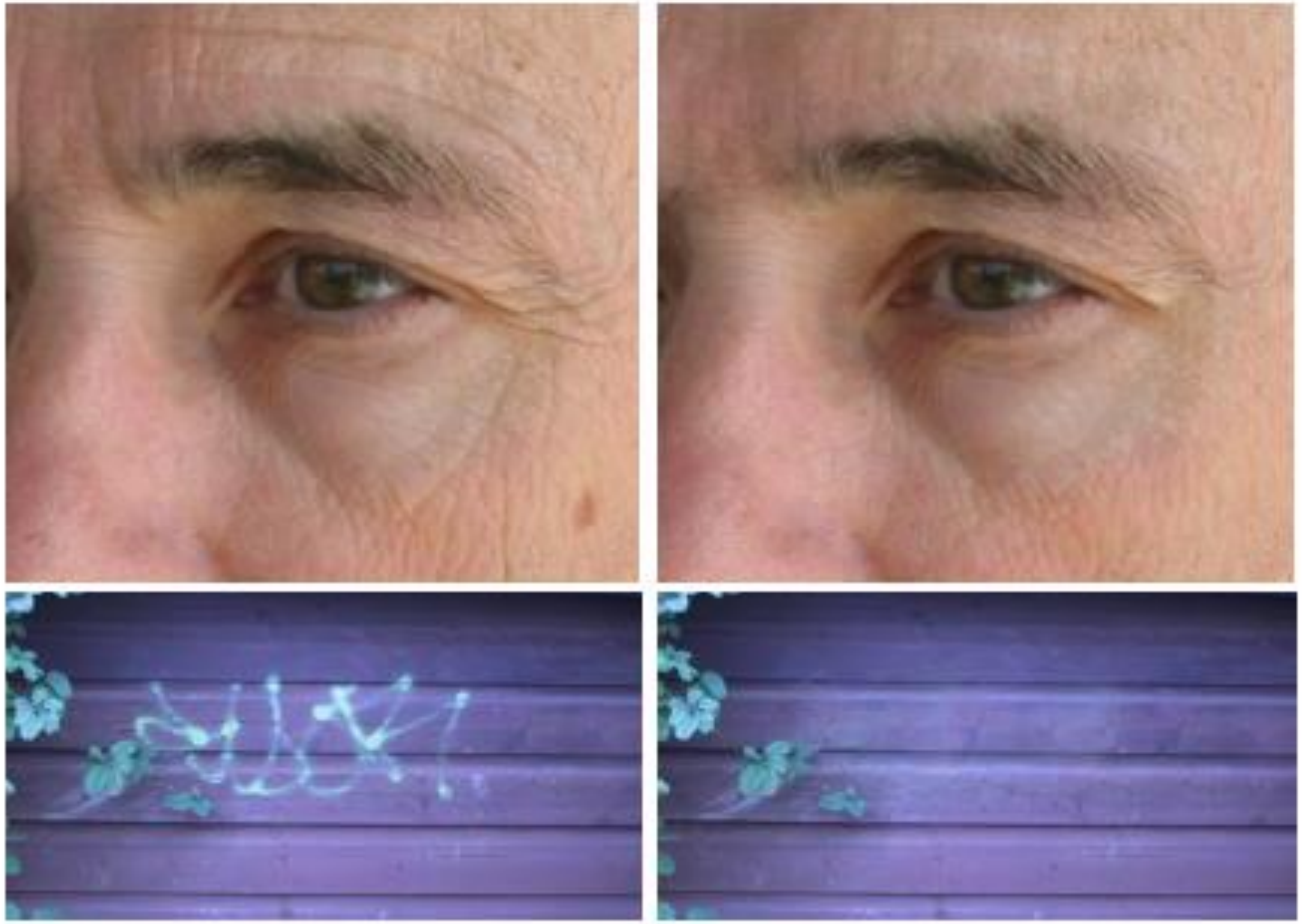


Figure 2: **Concealment**. By importing seamlessly a piece of the background, complete objects, parts of objects, and undesirable artifacts can easily be hidden. In both examples, multiple strokes (not shown) were used.

Perez et al. SIGGRAPH 03

Problem with composition

Misaligned (moving) objects become ghosts

Slide credit: A. Efros

Image Composition

Laplacian pyramid blending

Poisson cloning

Graph cut seams

Texture synthesis

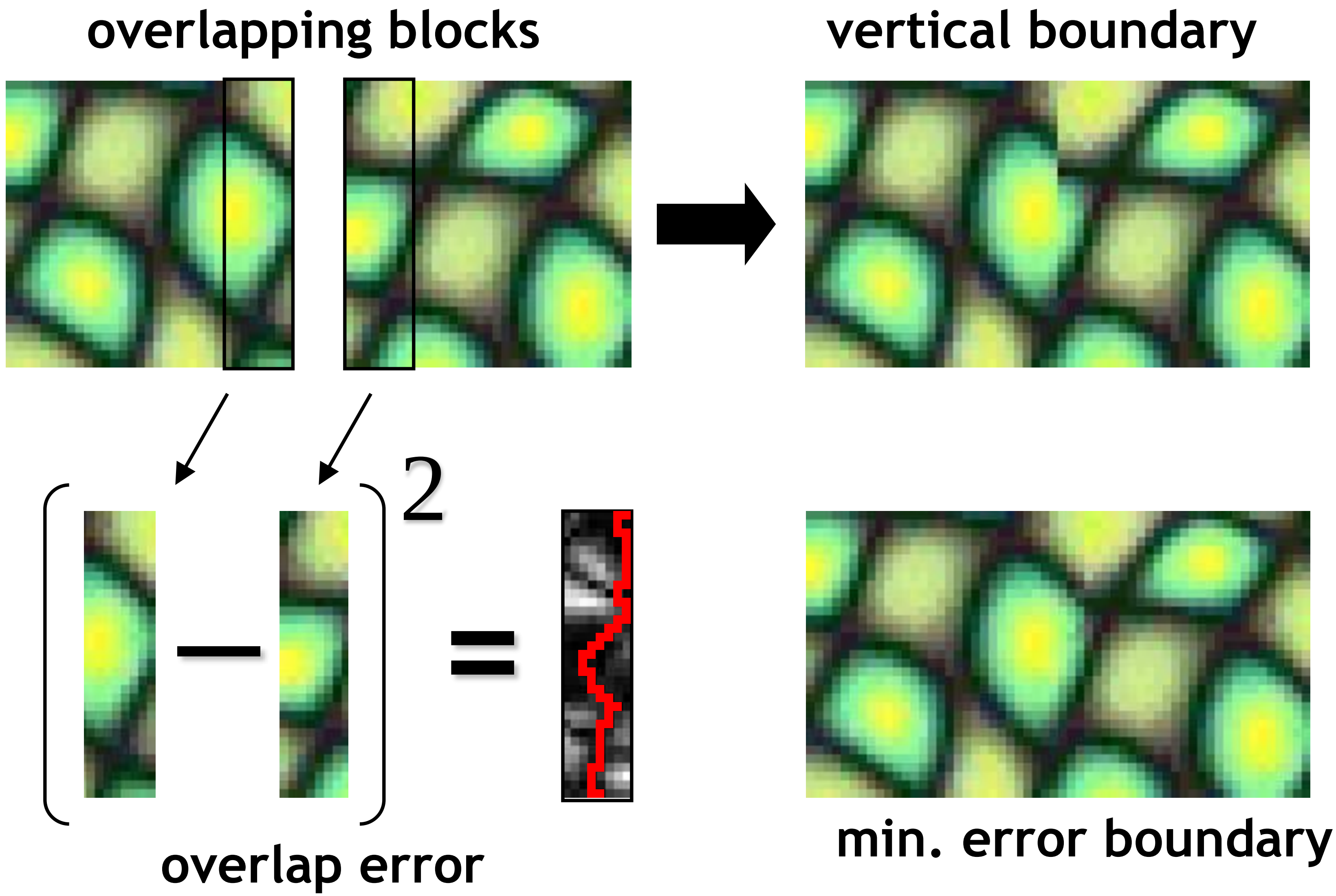
Graph Cuts

General idea

- Single source image per segment (avoids blurring)
- Careful cut placement, plus optional blending (avoids seams)

Slide credit: A. Efros

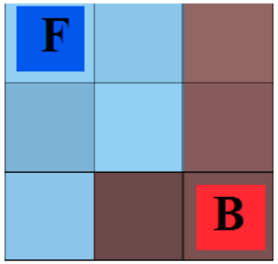
Graph Cuts in Image Composition

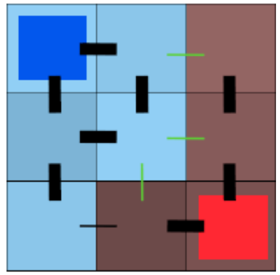


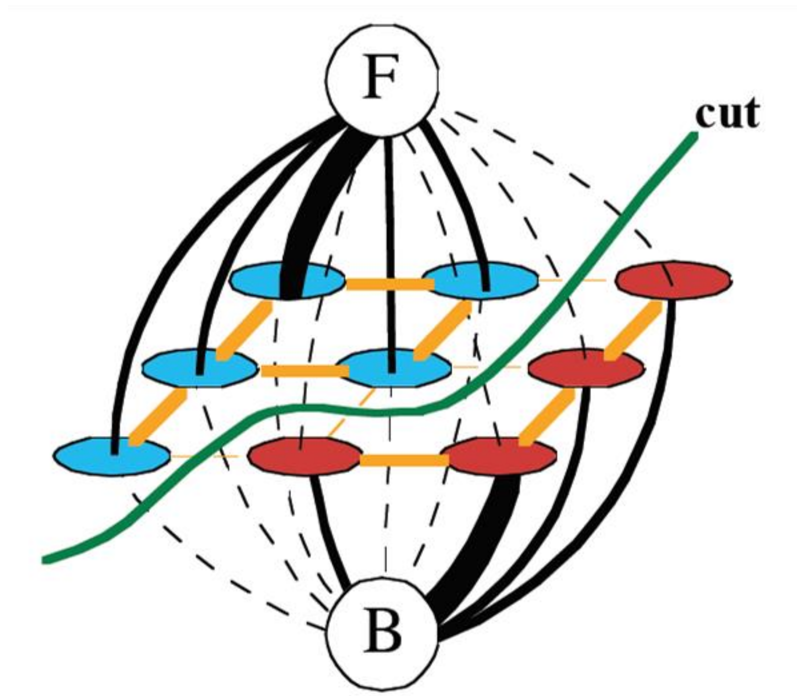
Slide credit: A. Efros

Graph Cut Algorithm

Boykov&Jolly, ICCV'01







F	F	B
F	F	B
F	B	B

Minimum cost cut can be computed in polynomial time

(max-flow/min-cut algorithms)

Graph Cuts in Image Segmentation

Lazy Snapping [Li 2004]

Interactive segmentation using graphcuts

Graph cuts in Image Retargeting

Seam Carving

Graph cuts in Image Retargeting

Seam Carving

Shamir

Problems with Graph Cuts

Image Composition

Laplacian pyramid blending

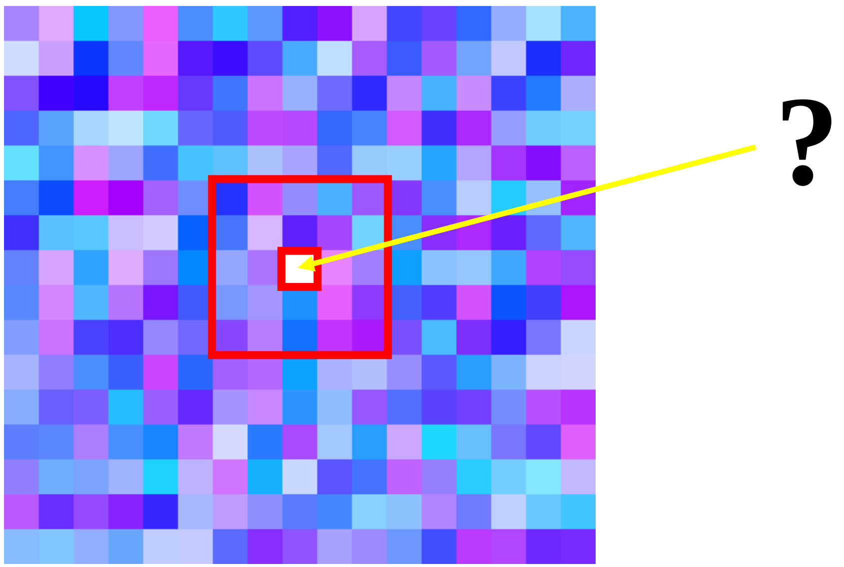
Poisson composition

Graphcut seams

Texture synthesis

Nonparametric Texture Synthesis

- Assume patches in output image should locally match patches in input image
- *Markov property*:
  p(pixel | rest of image) = p(pixel | neighborhood)
- Use patches from set of input images to model p(pixel | neighborhood)



Motivation from Language

Shannon (1948) proposed a way to generate English-looking text using *N-grams:*

Assume a Markov model

Large corpus gives probability distribution for each letter, given N–1 previous letters

Starting from a seed, repeatedly sample conditional probabilities to generate new letters

Can also use whole words instead of letters

Mark V. Shaney (Bell Labs)

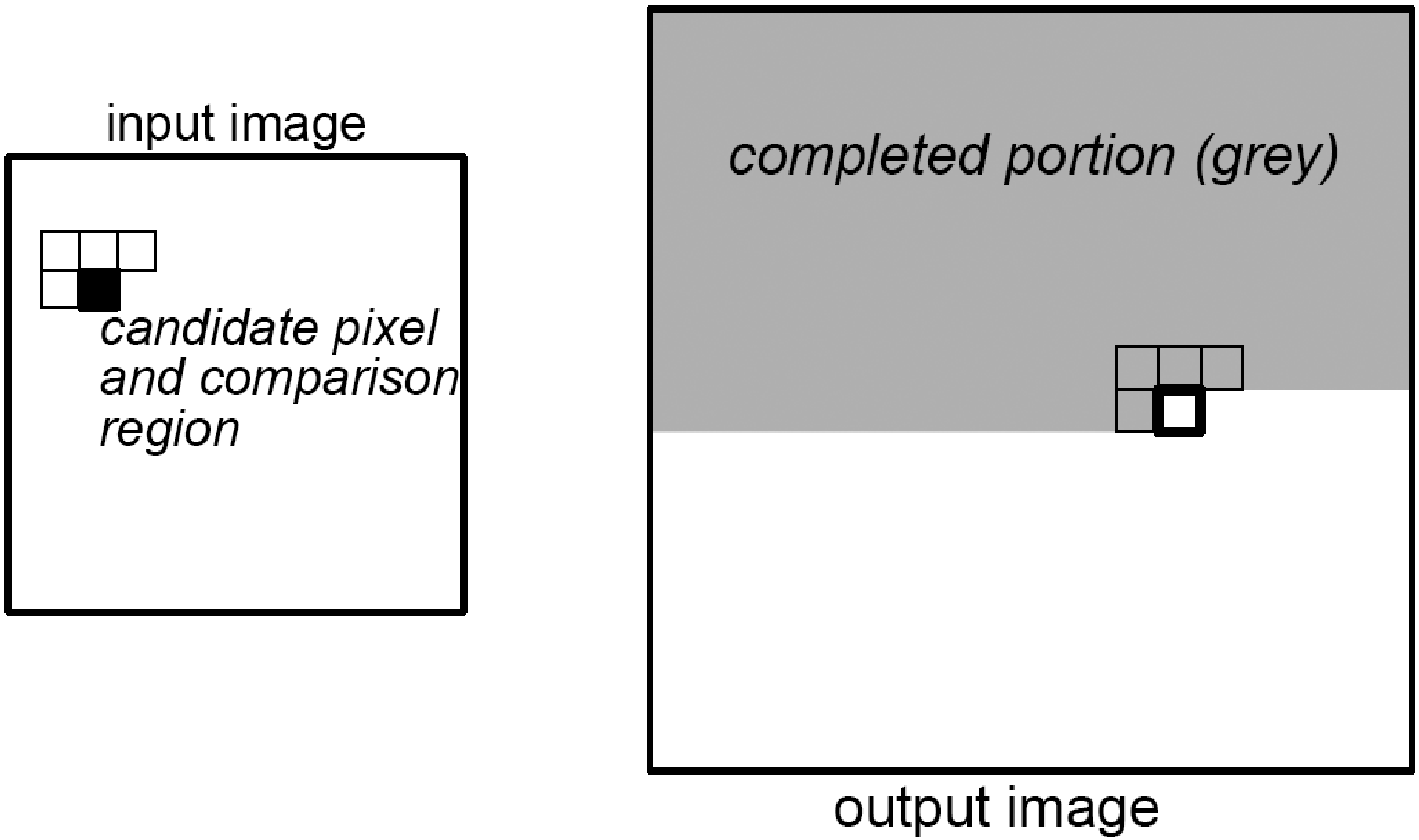
Results (using alt.singles corpus):

- *"As I've commented before, really relating to someone involves standing next to impossible."*
- *"One morning I shot an elephant in my arms and kissed him."*
- *"I spent an interesting evening recently with a grain of salt."*

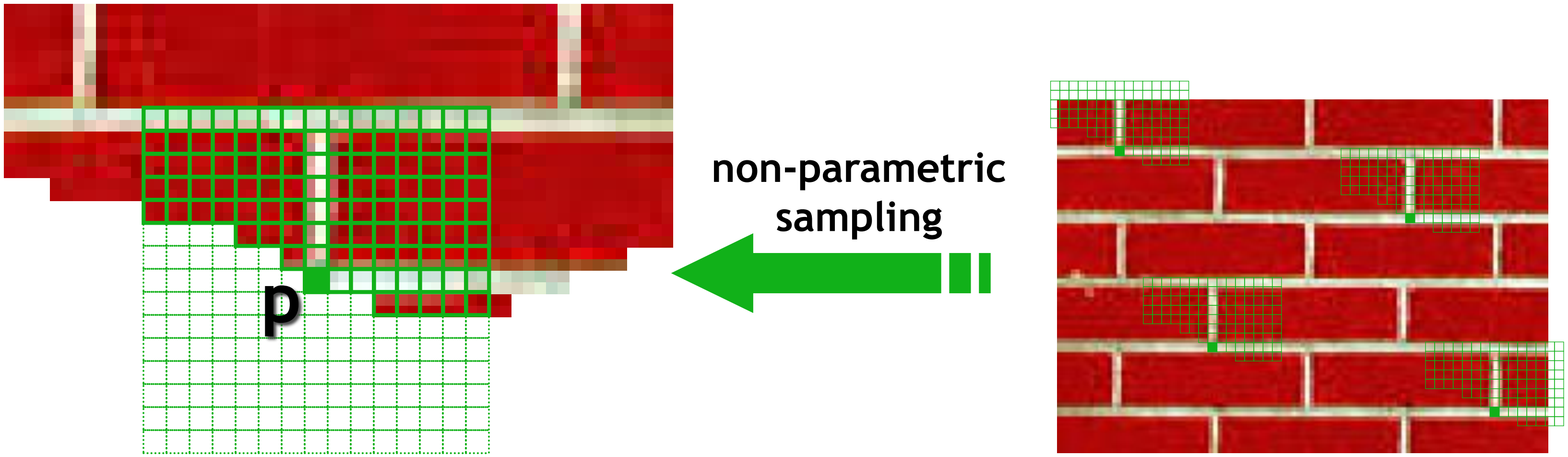
Notice how well local structure is preserved!

- Now let's try this in 2D...

Efros & Leung Algorithm



Efros & Leung Algorithm



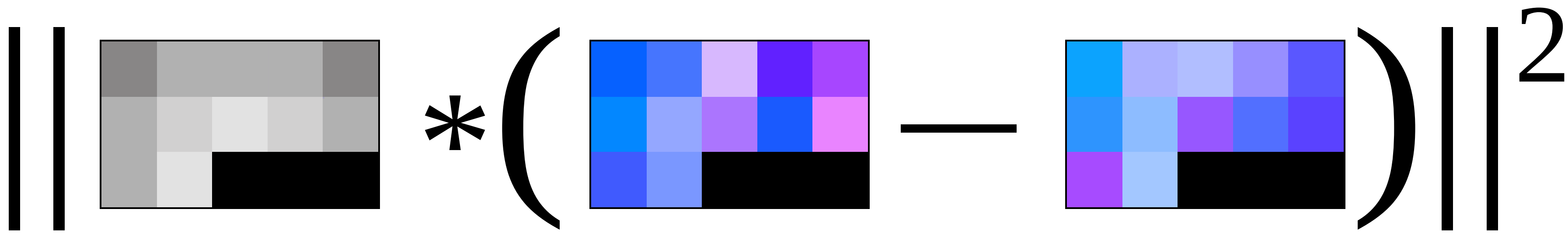
Synthesizing a pixel

Input image

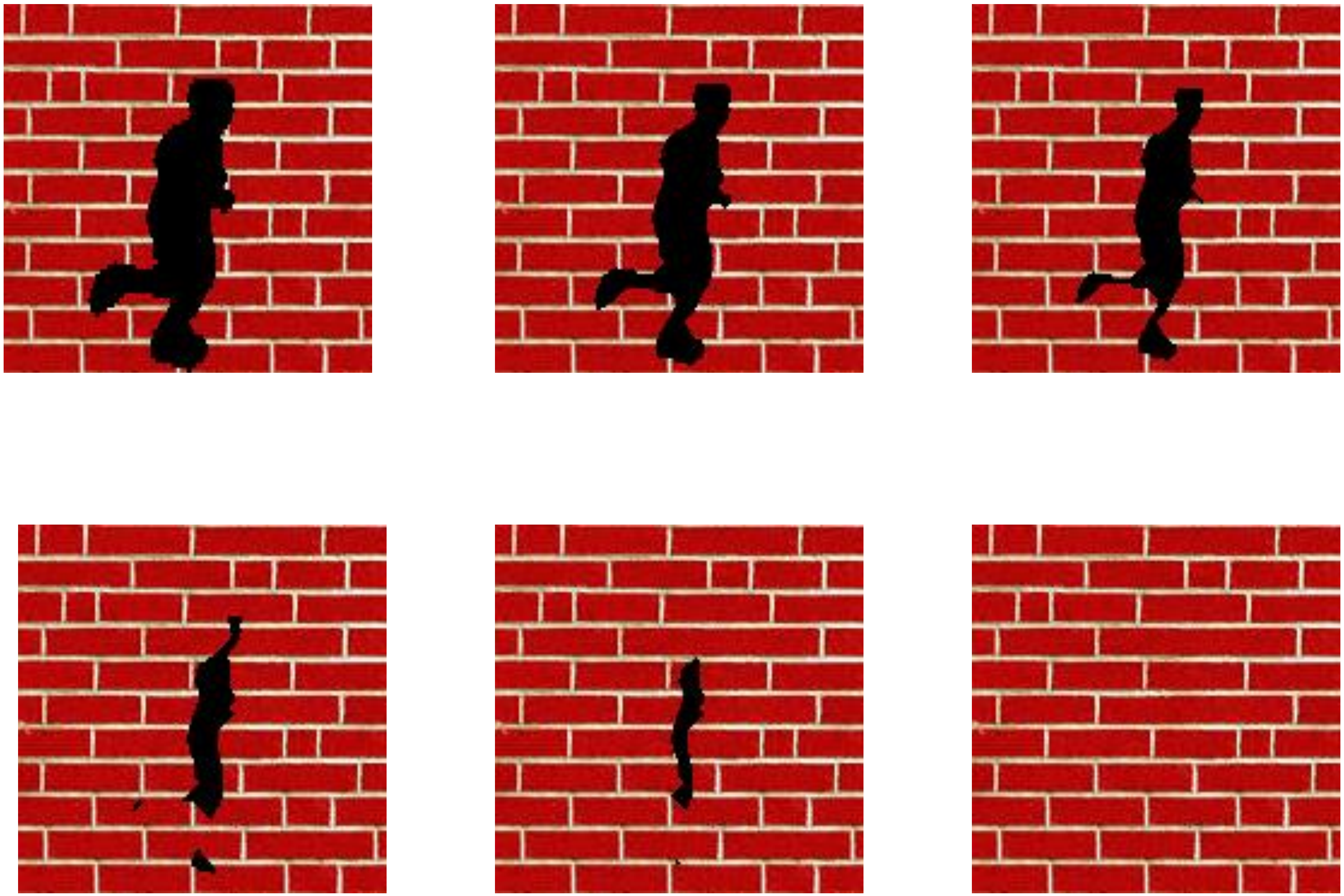
Finding matches

E.g., sum of squared differences (SSD)

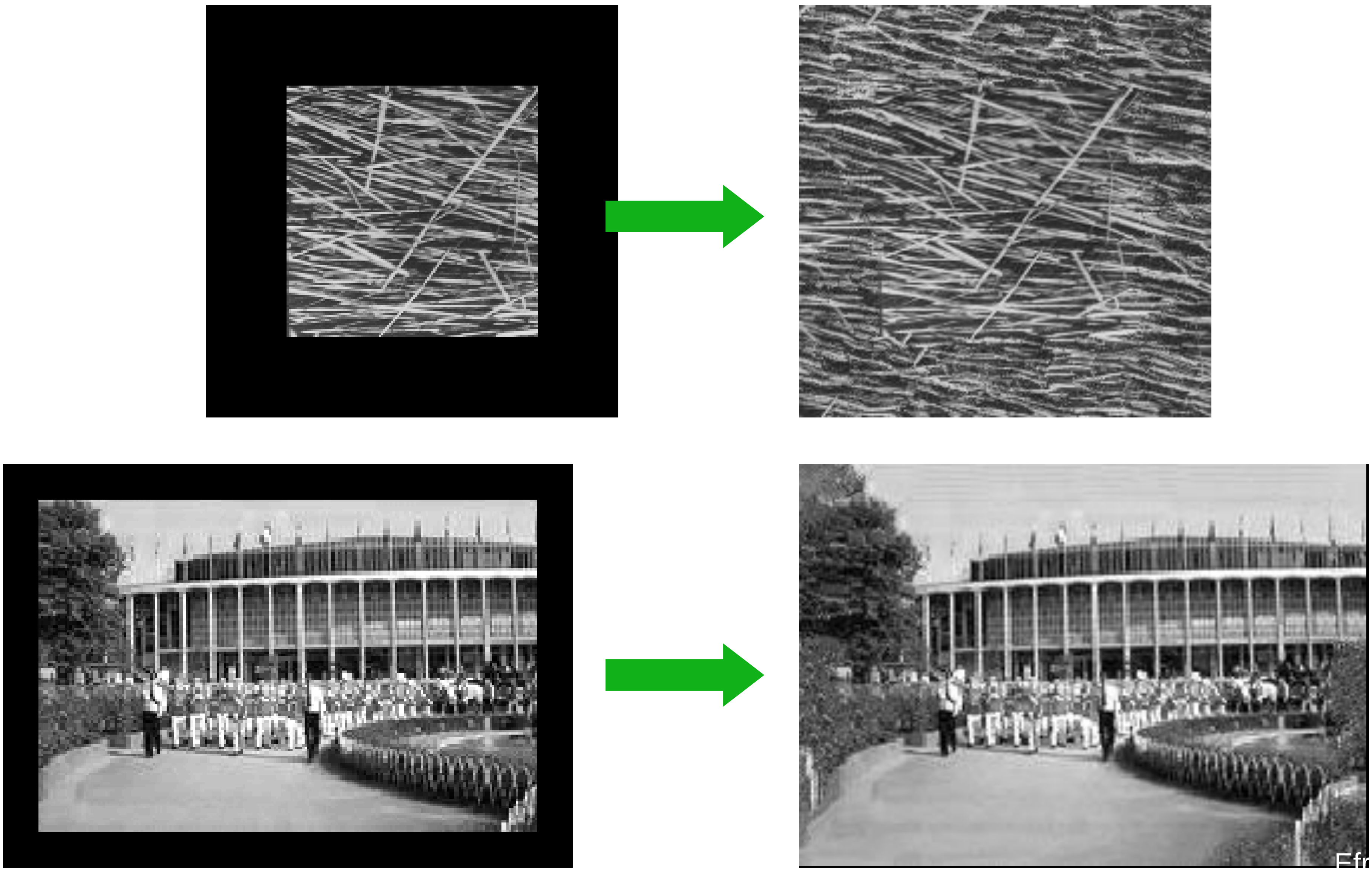
- *Gaussian-weighted* to make sure closer neighbors are in better agreement



Hole Filling



Extrapolation



Practical texture synthesis

Fast similarity search

Coherence

Multiresolution

Patches

Quilting

Similarity Search

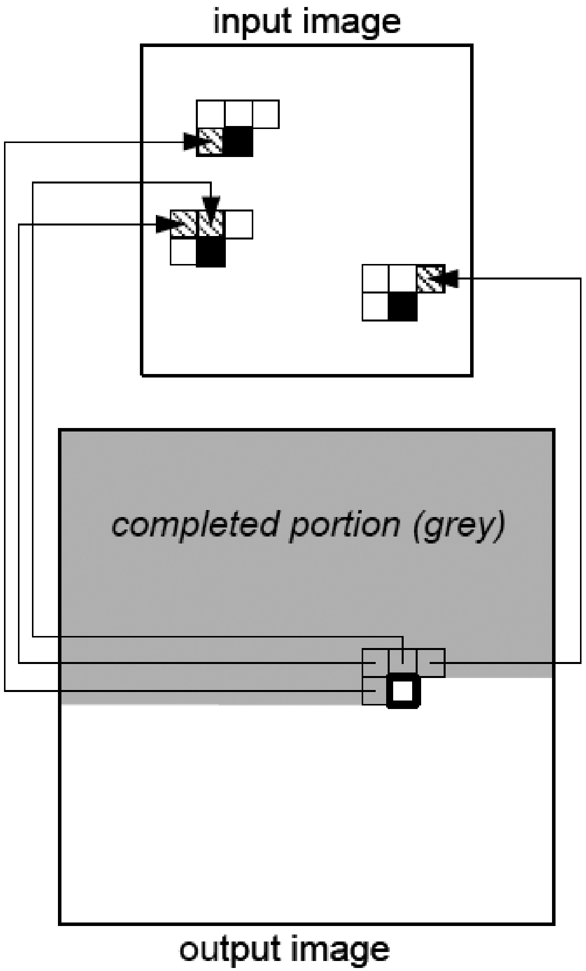
Perform fast approximate nearest neighbor search using spatial data structure

- *tree-structured vector quantization (TSVQ)*
- *kd-tree (optionally with PCA)*

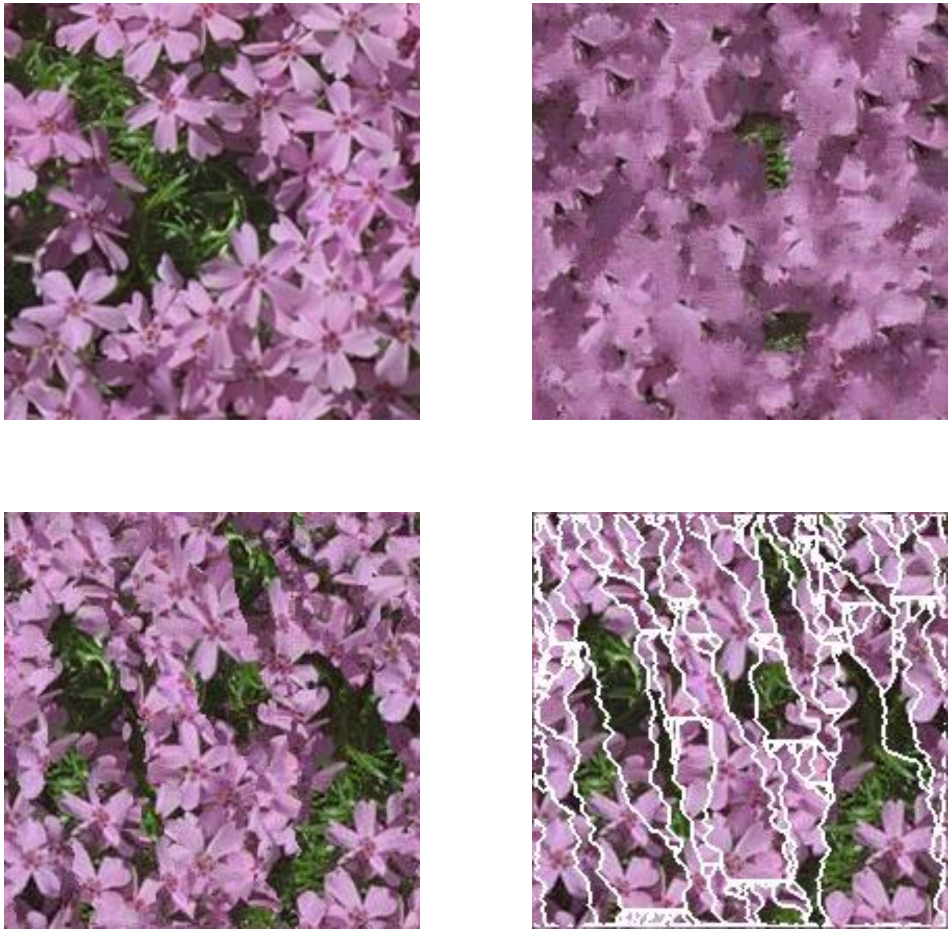
Perform fast approximate nearest neighbor search using randomized algorithm

- *Patch-Match [Barnes09]*

Coherence



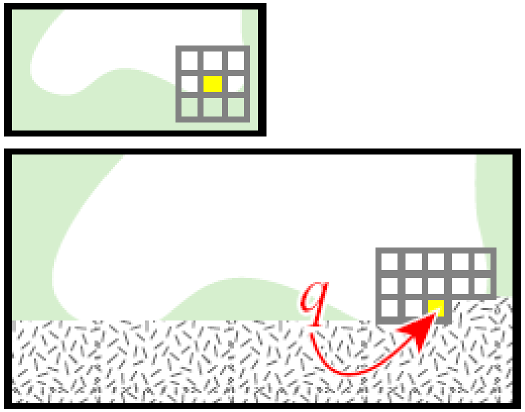
Coherence



Multiresolution

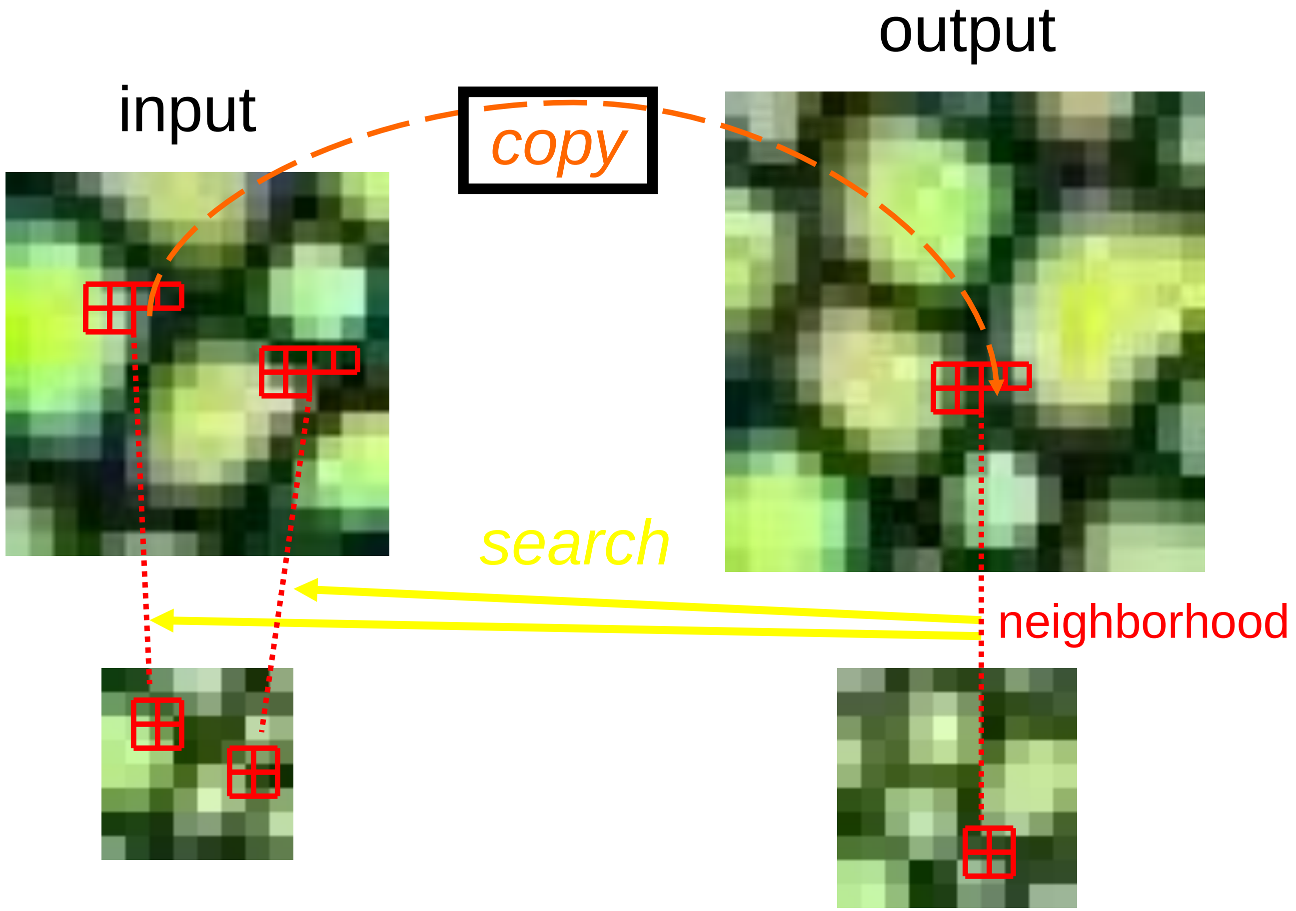
For textures with large-scale structures, use a *Gaussian pyramid* to reduce required neighborhood size

1. Synthesize at low-resolution
2. Repeat for higher-res levels: “neighborhood” consists of generated pixels at this level and all neighboring pixels at lower level



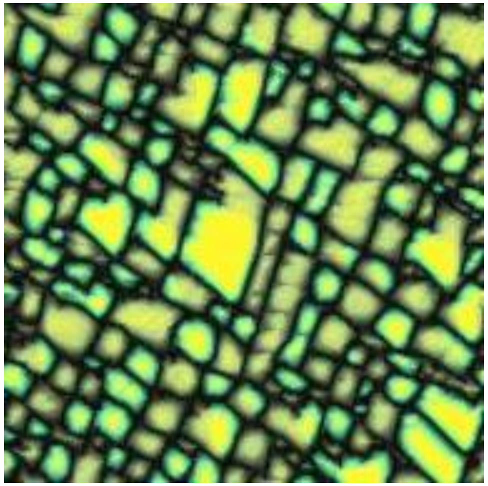
Multiresolution

Example:

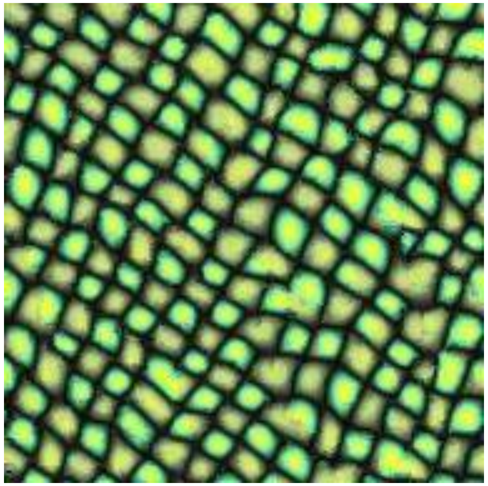


Multiresolution

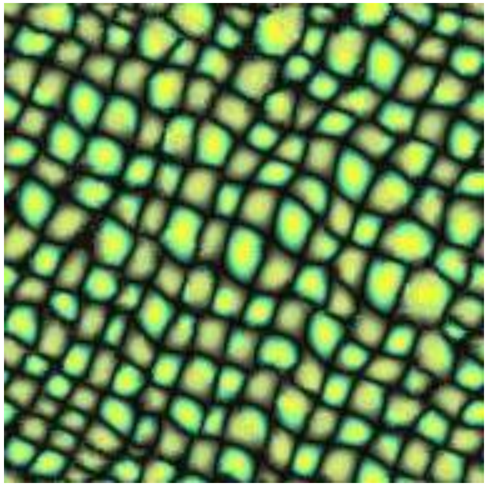
Results



1 level
5×5



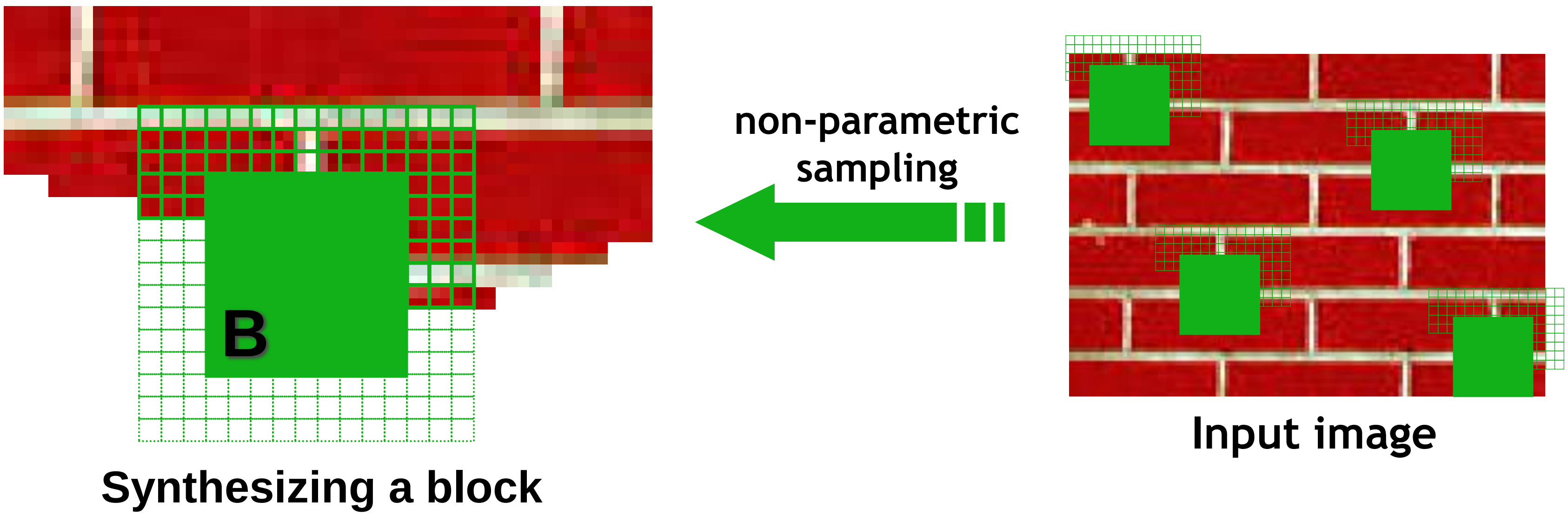
1 level
11×11



3 levels
5×5

Patch-Based Synthesis

Copy patches of pixels rather than pixels

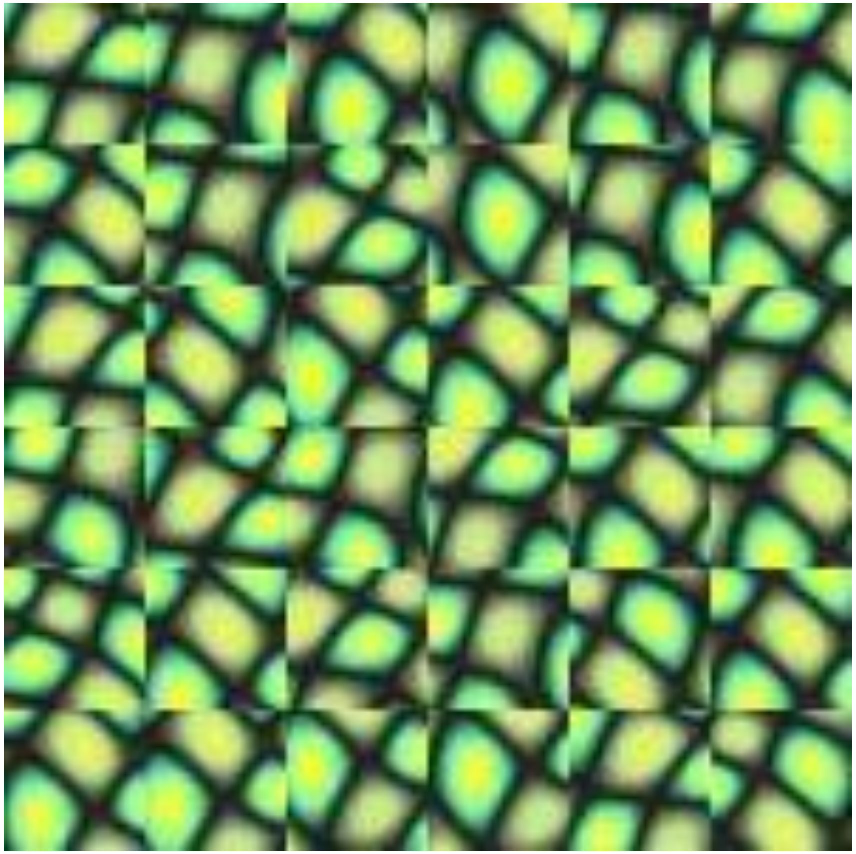


<u>Observation:</u> neighbor pixels are highly correlated

- Exactly the same as Efros & Leung but P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once

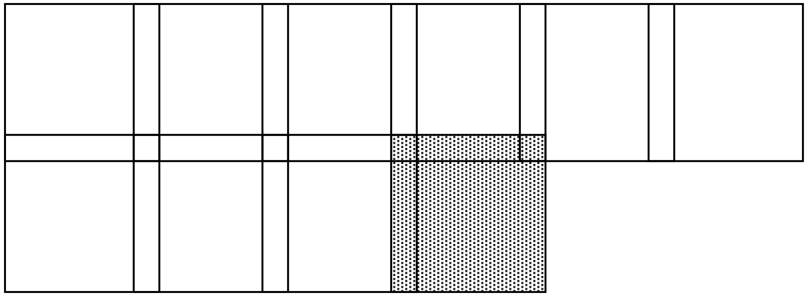
Image Quilting [Efros & Freeman]

Regularly arranged patches



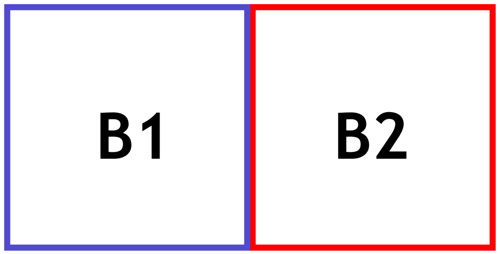
Efros & Freeman Algorithm

- Decompose image into tiles (patches)
- Synthesize tiles in raster order

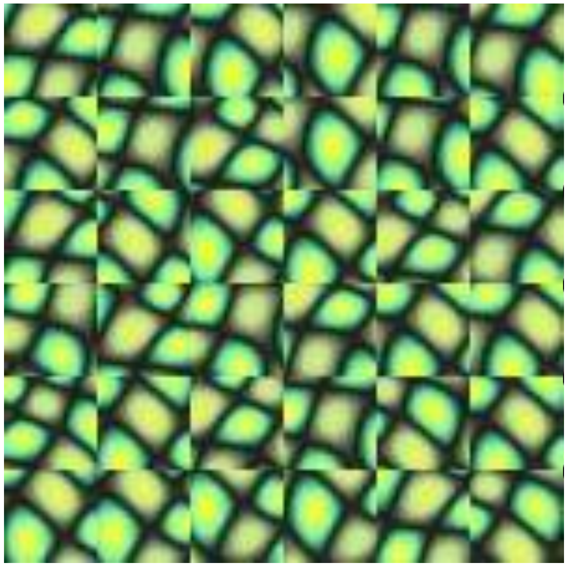


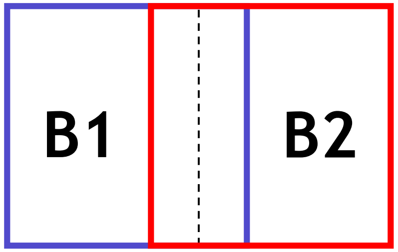
- Search input texture for tile that satisfies overlap constraints (above and left)
- Paste new tile into resulting texture
  - Adjust overlap areas with graph cut, or other image composition method

Efros & Freeman Example

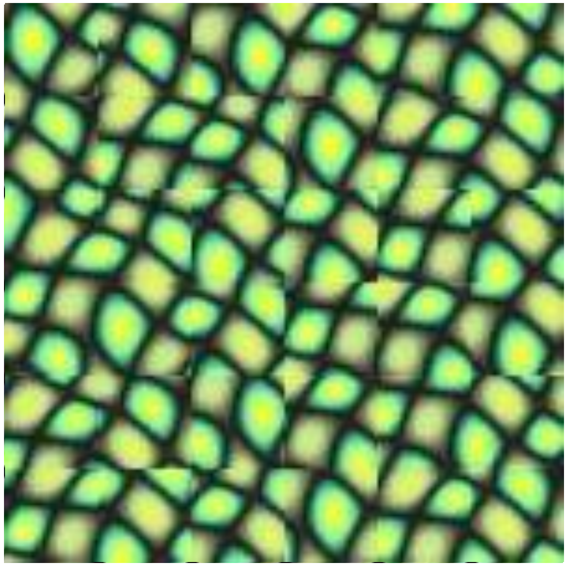


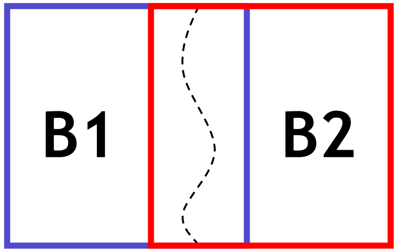
Random placement of blocks



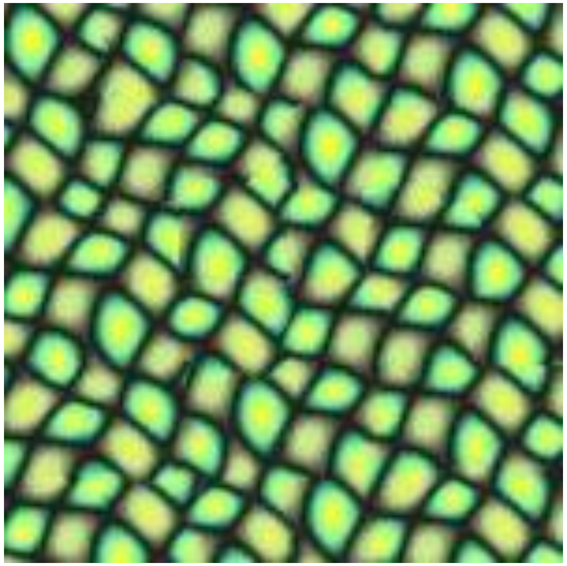


Neighboring blocks constrained by overlap



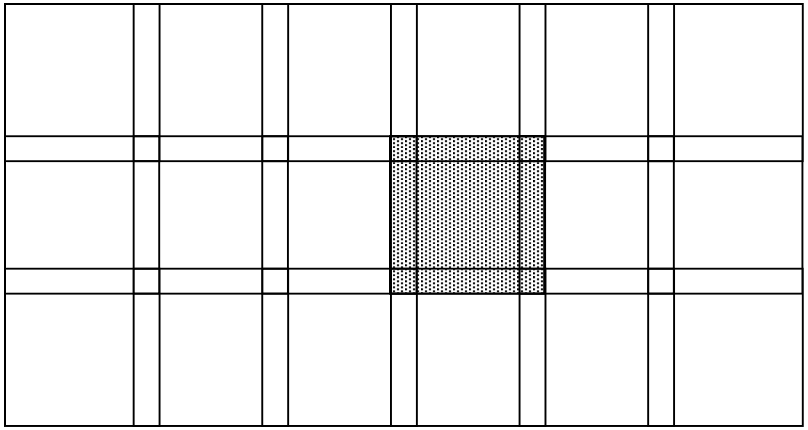


Minimal error boundary cut



Wexler Algorithm

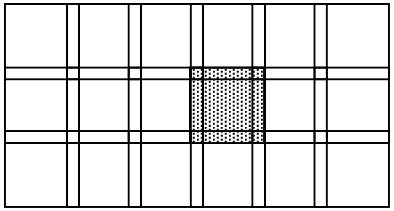
- Initialize output image
- Decompose image into tiles (patches)



- Iteratively pick tile
  - Search for highest scoring tile from source images (e.g., best matches colors)
  - Adjust overlap areas with graph cut, or other image composition method

Image Melding [Darabi12]

- Initialize output image
- Decompose image into tiles (patches)



- Iteratively
  - Replace all tiles based on matches of color and gradient after gain normalization
  - Reconstruct image colors and gradients by voting with overlapping tiles
  - Solve for pixel colors with Poisson method

Image Melding [Darabi12]

Image Melding: Combining Inconsistent Images using Patch-based Synthesis

Soheil Darabi[1], Connelly Barnes[2], Eli Shechtman[2], Dan B Goldman[2], Pradeep Sen[1]

[1]UNM Advanced Graphics Lab, [2]Adobe Systems

Our Next Assignment ….