# Blending and Compositing

Computational Photography

Derek Hoiem, University of Illinois

# Project 1: issues

- Basic tips
  - Display/save Laplacian images using mat2gray or imagesc
  - Downsampling: use sigma ~= 2  for factor of 2

- Emailing projects: code only (no images)

# Color shift

Result from Sam Ricker

More red (increase a channel)

L*a*b* space

Less yellow (decrease positive b channel values)

# Project 1 Results

- Incomplete list of great project pages
  - Donald Cha: http://web.engr.illinois.edu/~ddcha3/cs498dwh/proj1/
  - Sam Ricker: http://web.engr.illinois.edu/~sricker2/cs498dwh/proj1/
  - David Turner: http://web.engr.illinois.edu/~dkturne2/cs498dwh/proj1/
  - Elizabeth Weeks: http://web.engr.illinois.edu/~eweeks2/cs498dwh/proj1/
  - Jeremy Goodsitt: http://web.engr.illinois.edu/~goodsit2/cs498dwh/proj1/

# Hybrid results

Pooja Bag

Donald Cha

Hao Gao

# Last class: finding boundaries

- Intelligent scissors
  - Good boundary has a low-cost path from seed to cursor
  - Low cost = edge, high gradient, right orientation

- GrabCut
  - Good region is similar to foreground color model and dissimilar from background color
  - Good boundaries have a high gradient
  - Optimize over both

# Take-home questions

1. What would be the result in “Intelligent Scissors” if all of the edge costs were set to 1?

2. Typically, “GrabCut” will not work well on objects with thin structures. How could you change the boundary costs to better segment such objects?

# Last Class: cutting out objects

# This Class

How do I put an object from one image into another?

# Image Compositing


Some slides from Efros/Seitz

# News Composites

http://www.guardian.co.uk/world/2010/sep/16/mubarak-doctored-red-carpet-picture

Original

"Enhanced" Version

# News Composites

Original

“Enhanced” Version

Walski, LA Times, 2003

# Three methods

1. Cut and paste

2. Laplacian pyramid blending

3. Poisson blending

# Method 1: Cut and Paste

# Method 1: Cut and Paste

Method:

- Segment using intelligent scissors
- Paste foreground pixels onto target region

# Method 1: Cut and Paste

Problems:

- Small segmentation errors noticeable
- Pixels are too blocky
- Won’t work for semi-transparent materials

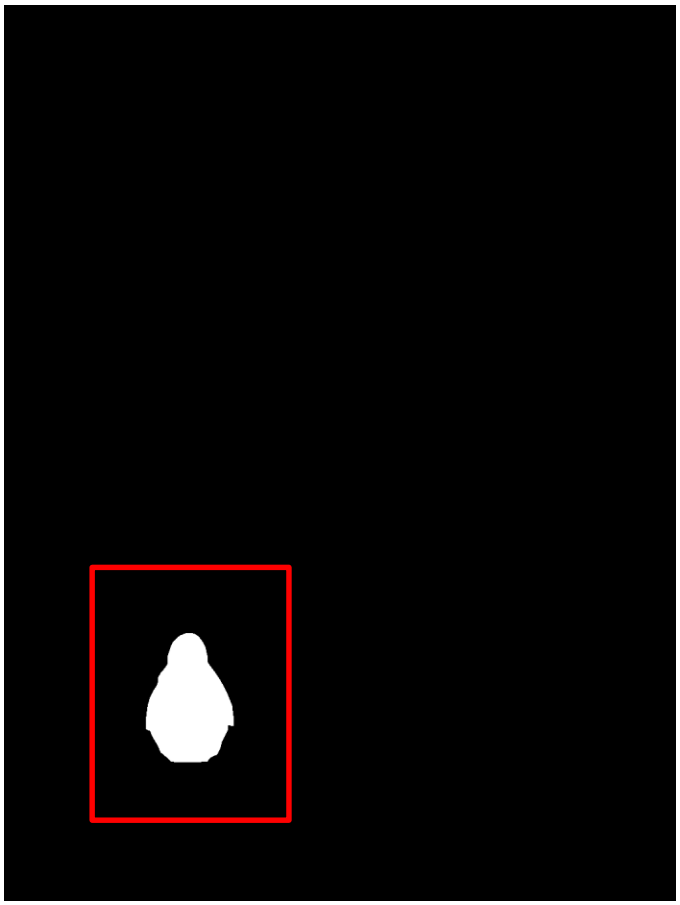
# Feathering

Near object boundary pixel values come partly from foreground and partly from background

# Method 1: Cut and Paste (with feathering)

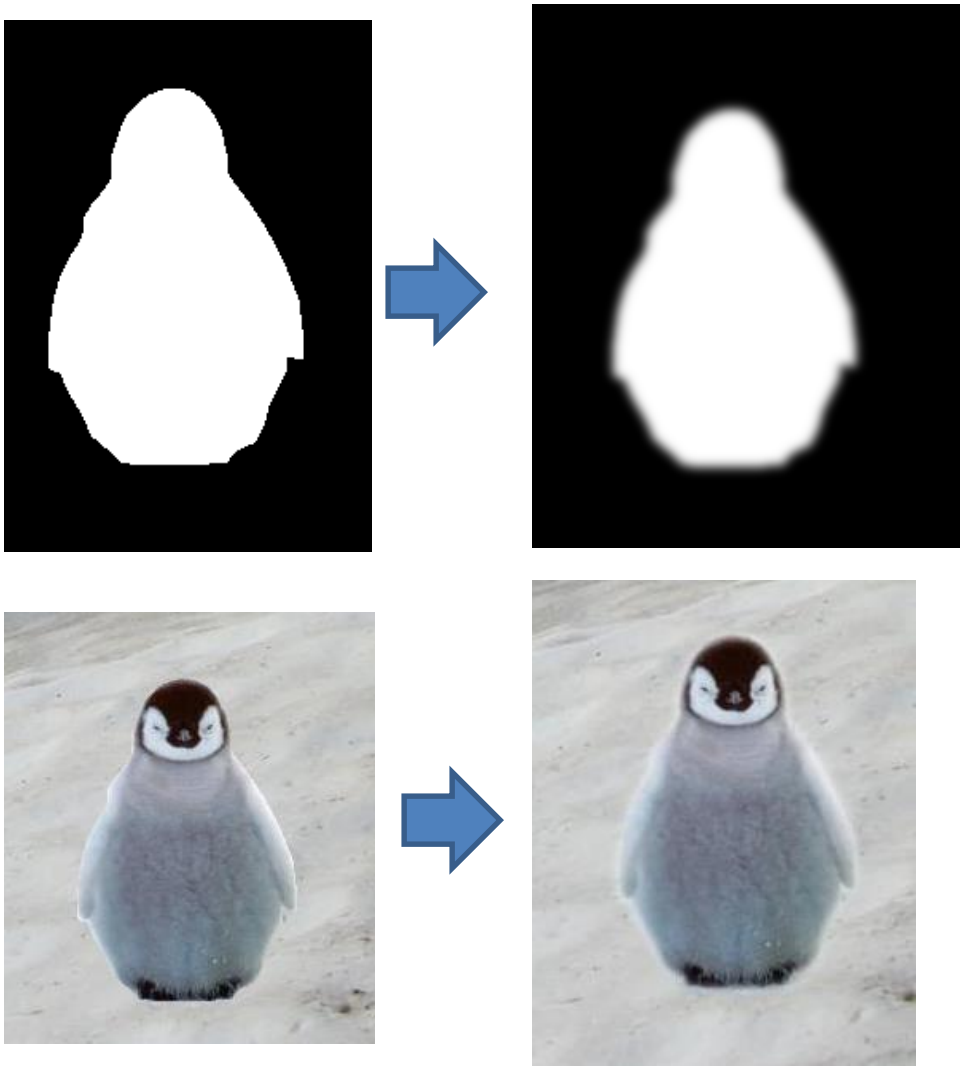
# Alpha compositing

+



Output = foreground*mask + background*(1-mask)

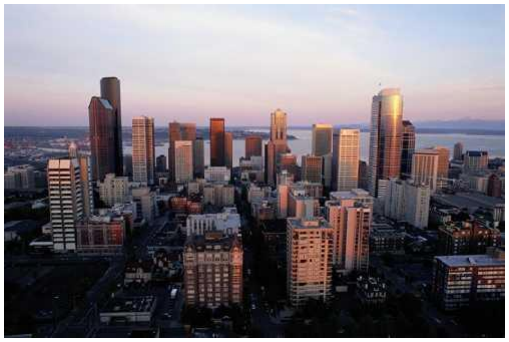
# Alpha compositing with feathering



Output = foreground*mask + background*(1-mask)

# Another example (without feathering)

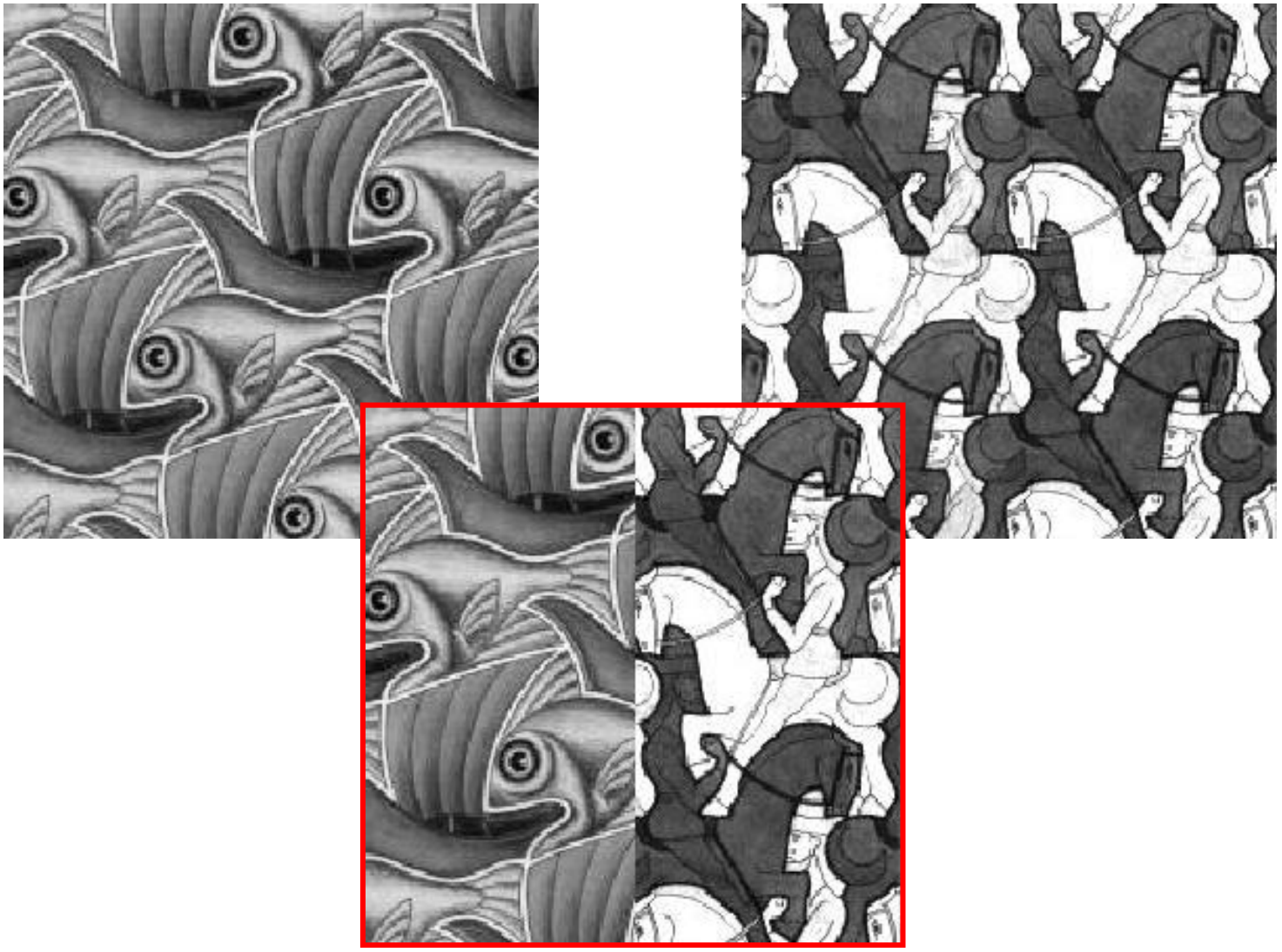
Mattes



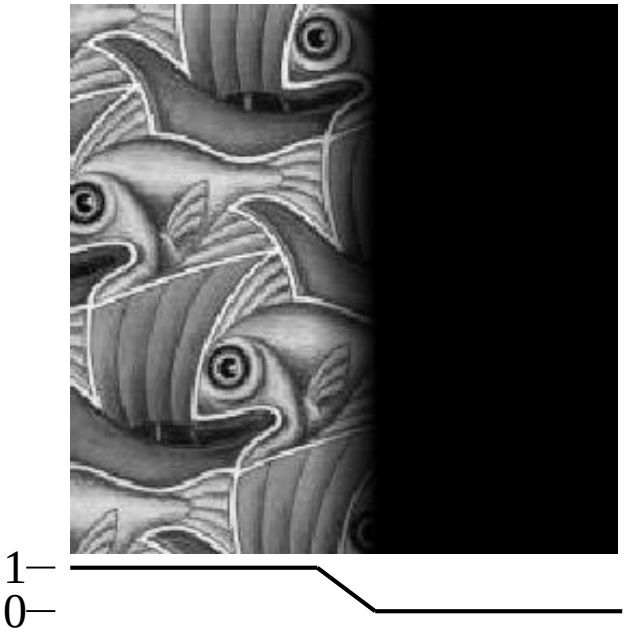

Composite

Composite by
David Dewey

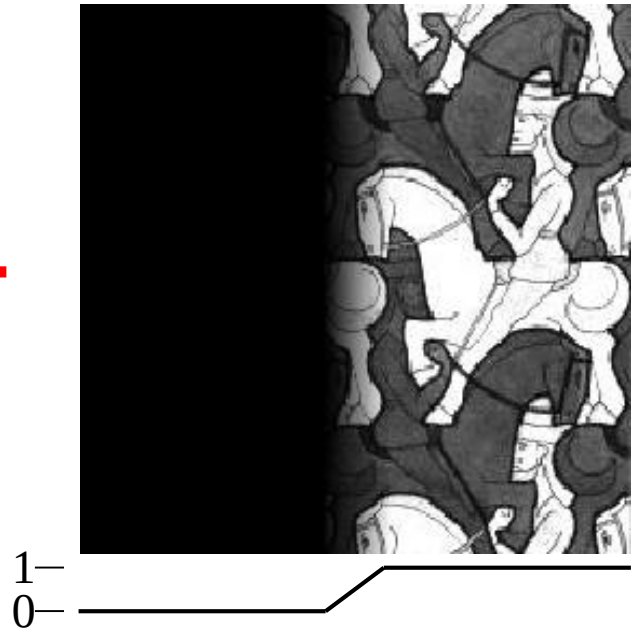
# Proper blending is key



# Alpha Blending / Feathering



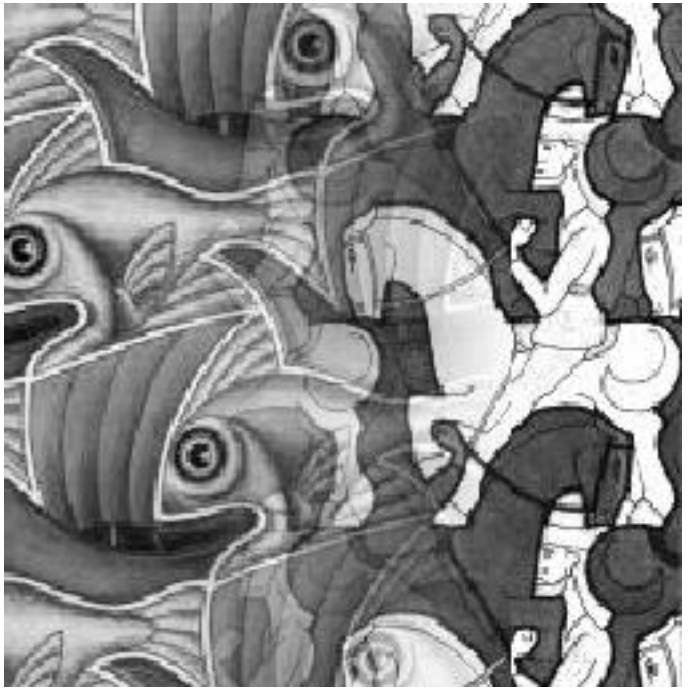

+

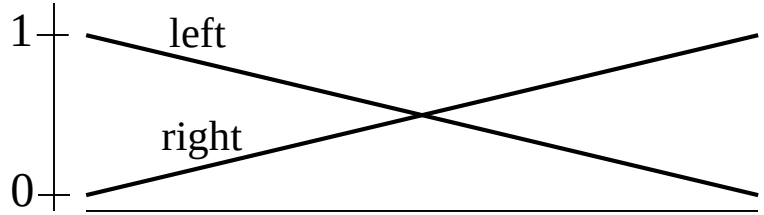


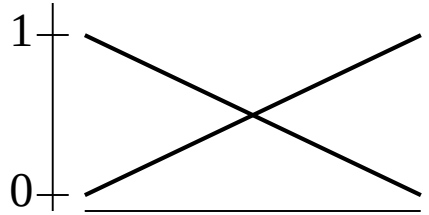

=

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

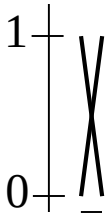
# Affect of Window Size

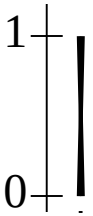




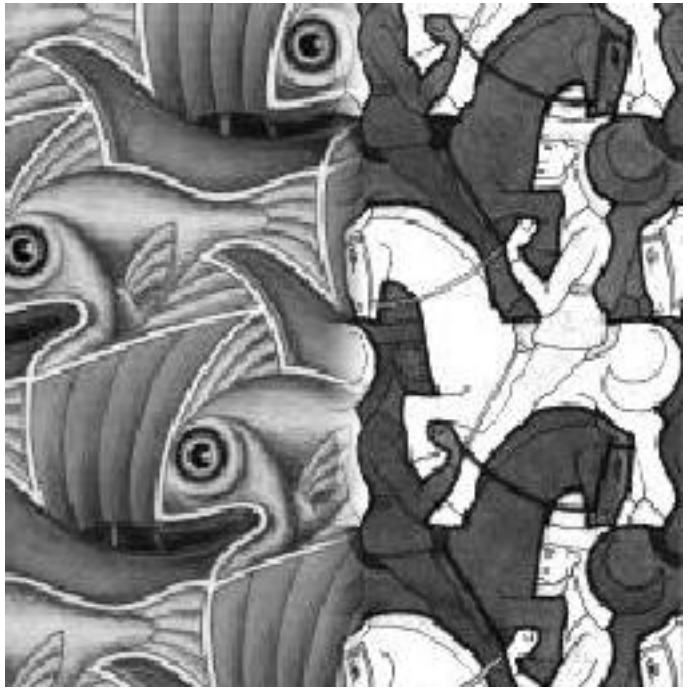



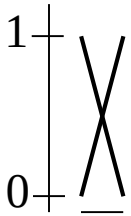
# Affect of Window Size



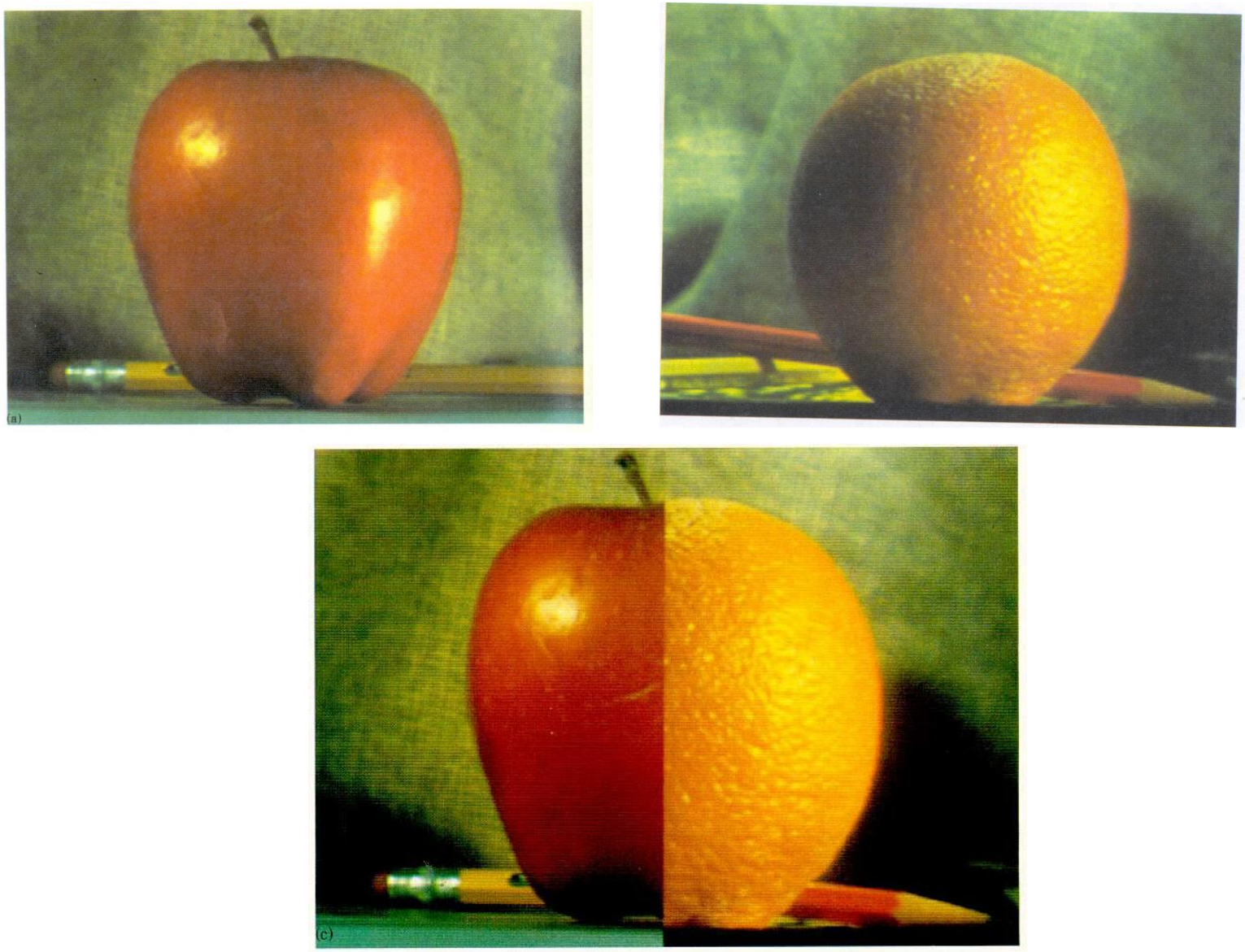
# Good Window Size





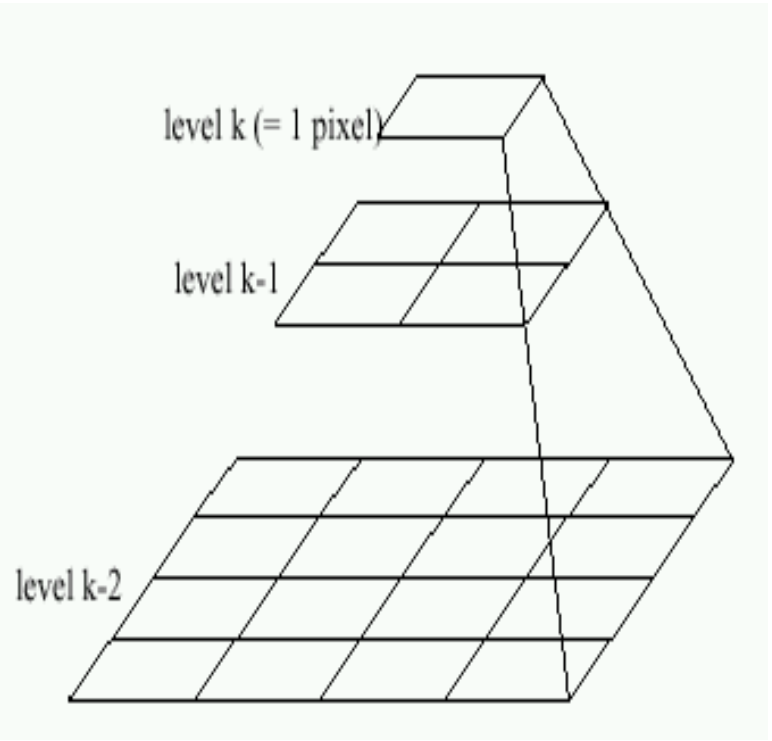

“Optimal” Window:  smooth but not ghosted

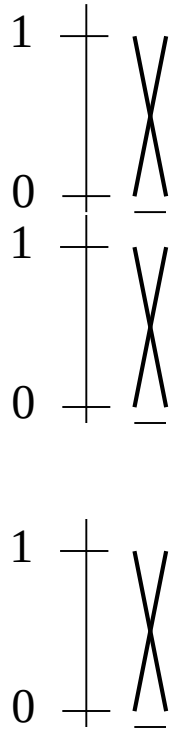
# How much should we blend?

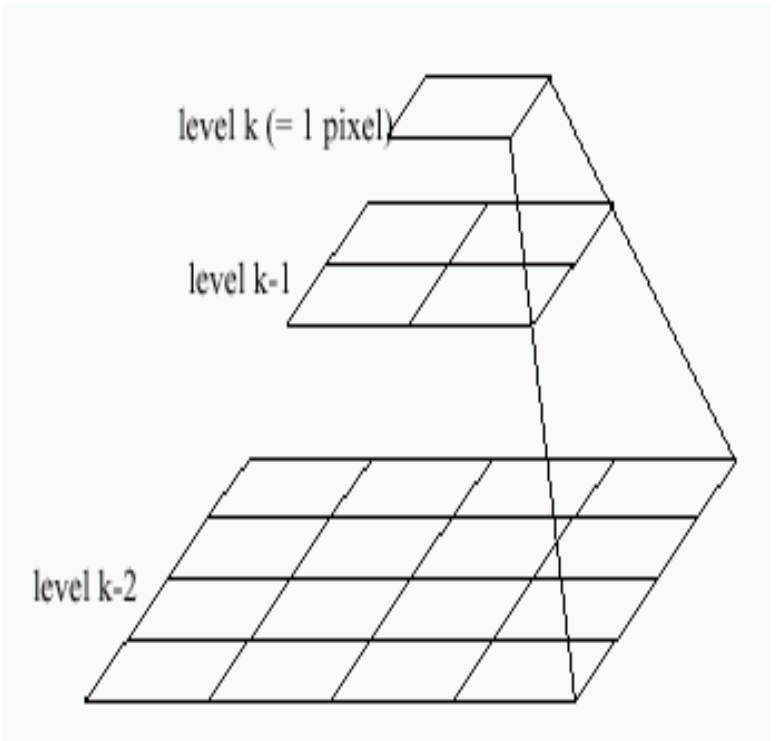


# Method 2: Pyramid Blending

- At low frequencies, blend slowly
- At high frequencies, blend quickly



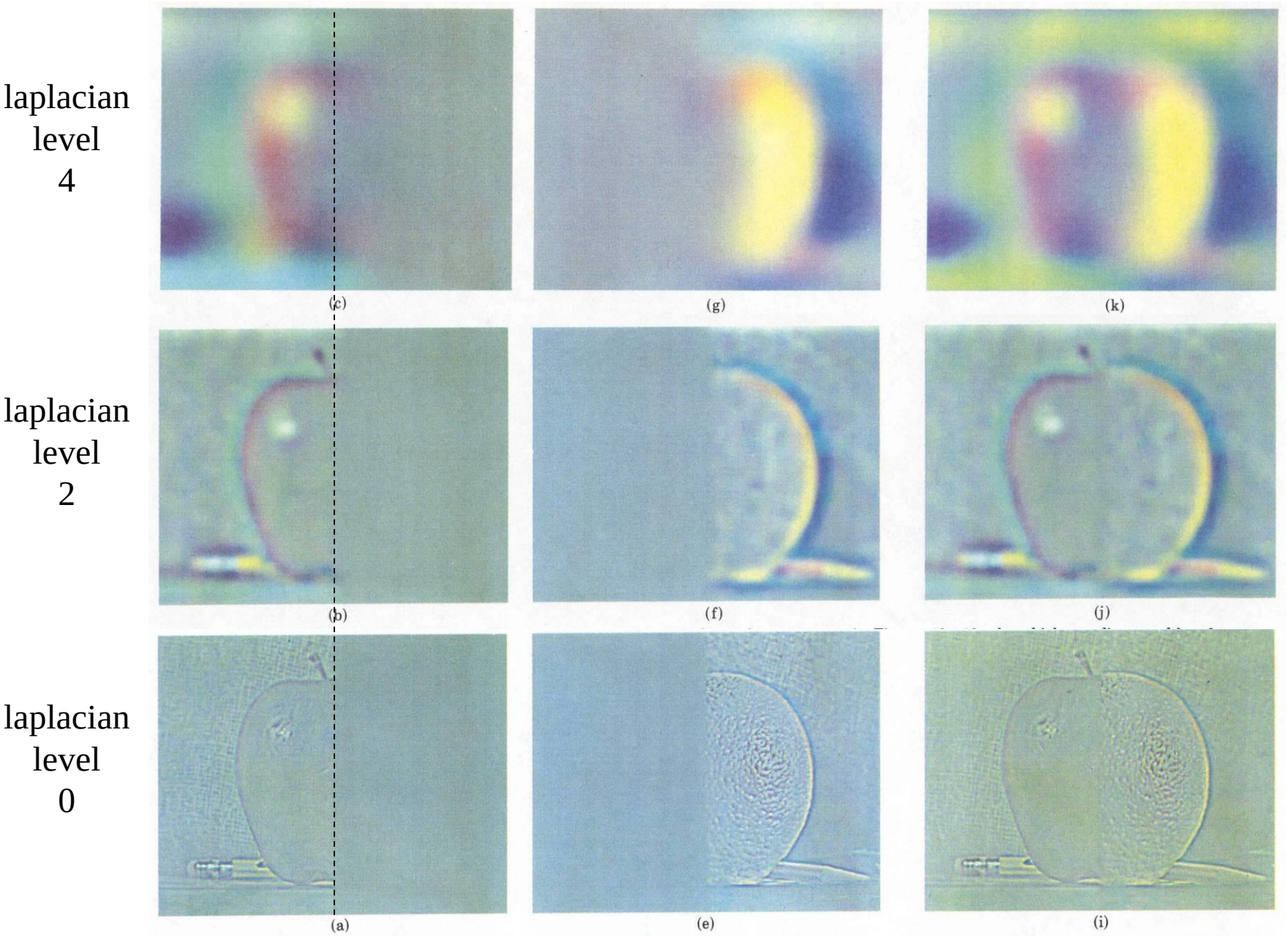



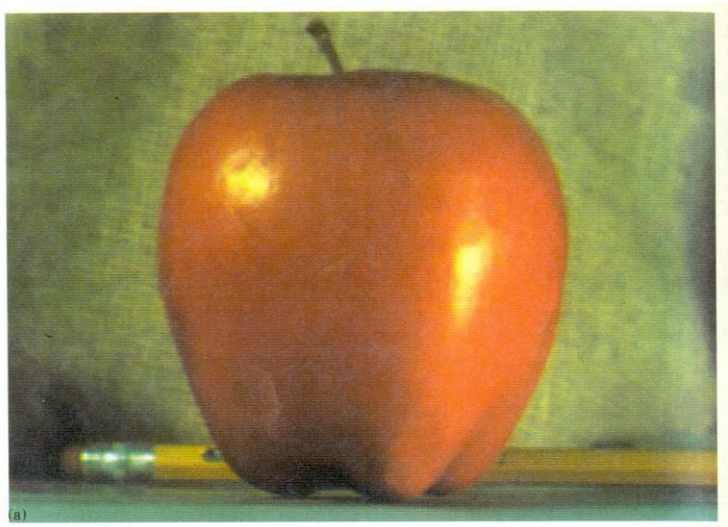

Left pyramid

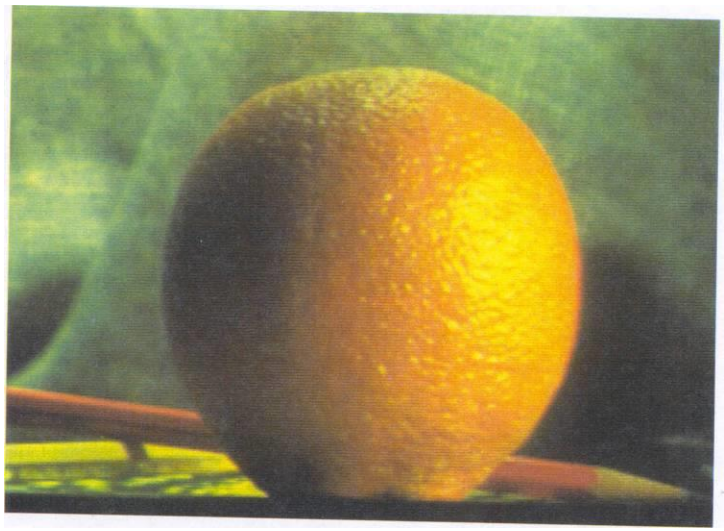
blend

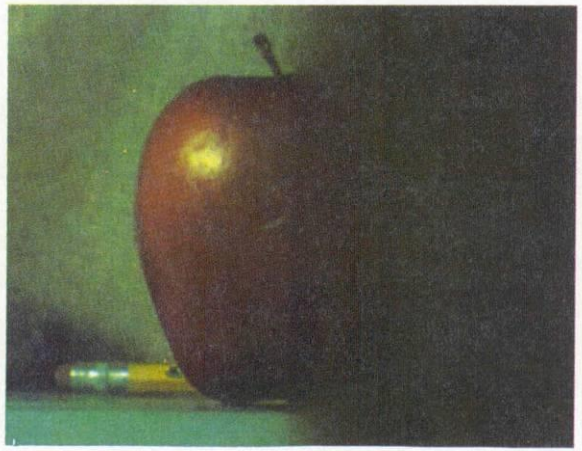
Right pyramid


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

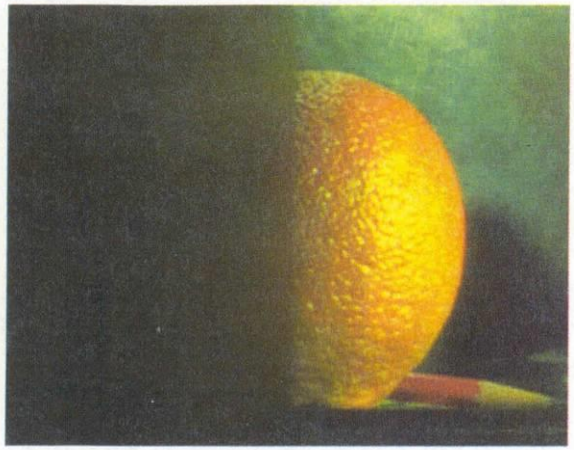
# Method 2: Pyramid Blending



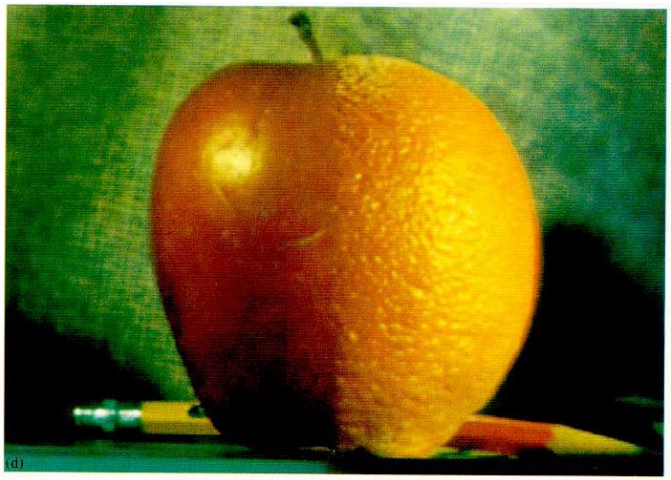




(d)



(h)



Burt and Adelson 1983

# Laplacian Pyramid Blending

Implementation:

1. Build Laplacian pyramids for each image
2. Build a Gaussian pyramid of region mask
3. Blend each level of pyramid using region mask from the same level

$$L_{12}^{i} = L_{1}^{i} \cdot R^{i} + L_{2}^{i} \cdot (1 - R^{i})$$

Image 1 at level i of Laplacian pyramid

Pointwise multiply

Region mask at level i of Gaussian pyramid

4. Collapse the pyramid to get the final blended image

Burt and Adelson 1983

# Simplification: Two-band Blending

- Brown & Lowe, 2003
  - Only use two bands: high freq. and low freq.
  - Blends low freq. smoothly

# 2-band Blending

Low frequency

High frequency

# Linear Blending

# 2-band Blending

# Blending Regions

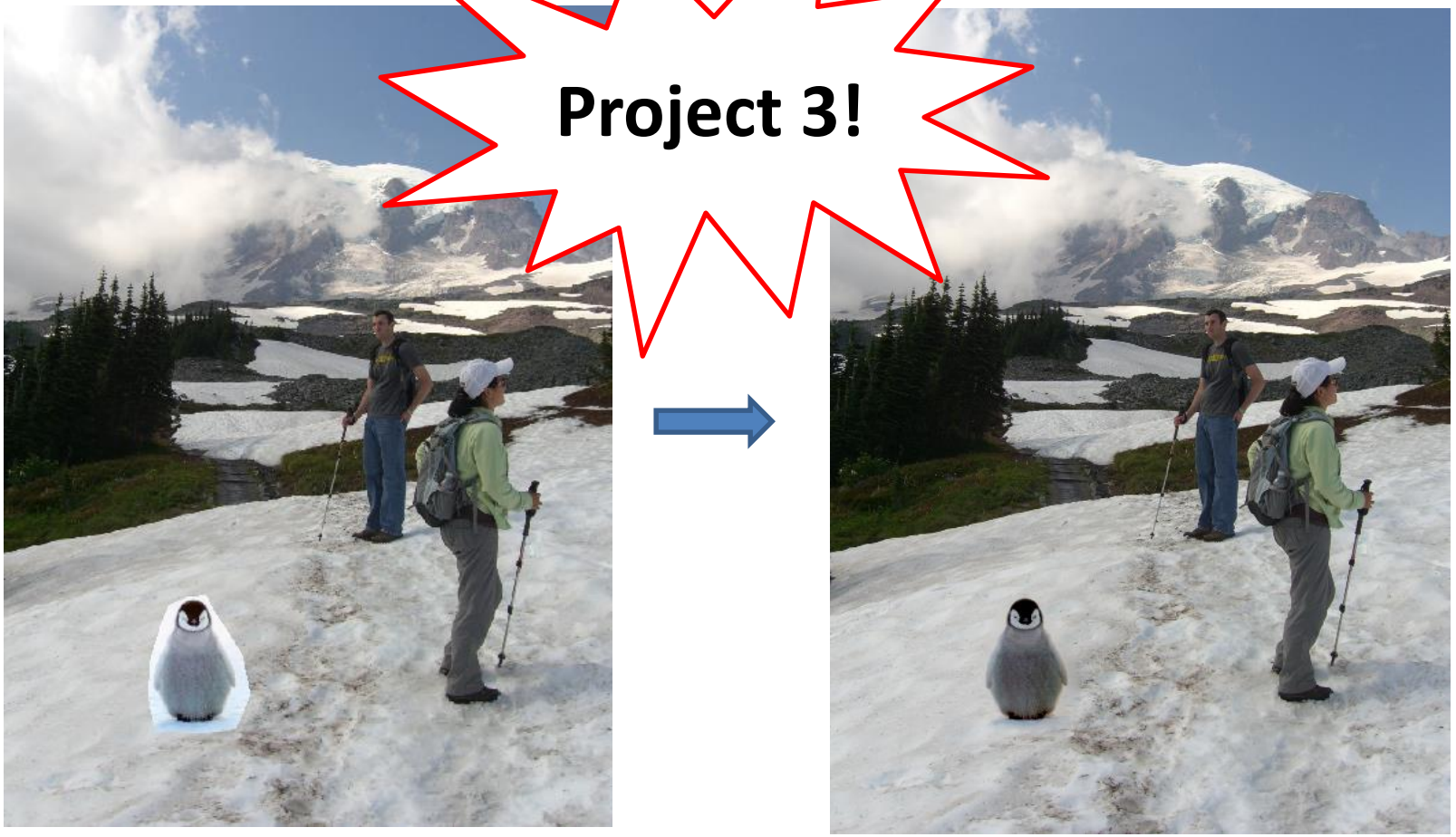
# Related idea: Poisson Blending

A good blend should preserve gradients of source region without changing the background

Perez et al. 2003

# Related idea: Poisson Blending

A good blend should preserve gradients of source region without changing the background




Perez et al. 2003

# Method 3: Poisson Blending

A good blend should preserve gradients of source region without changing the background

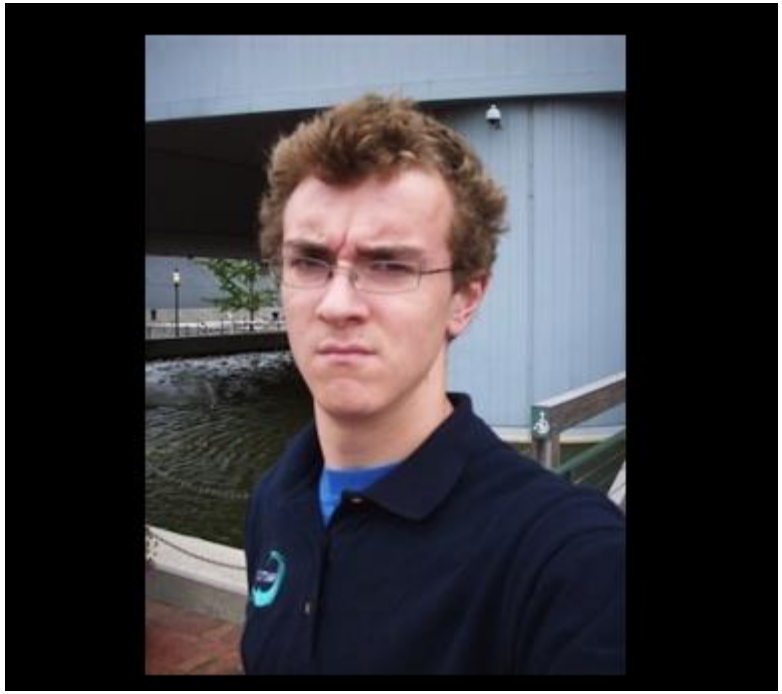
Treat pixels as variables to be solved

- Minimize squared difference between gradients of foreground region and gradients of target region
- Keep background pixels constant

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap \neg S} ((v_i - t_j) - (s_i - s_j))^2$$

Perez et al. 2003

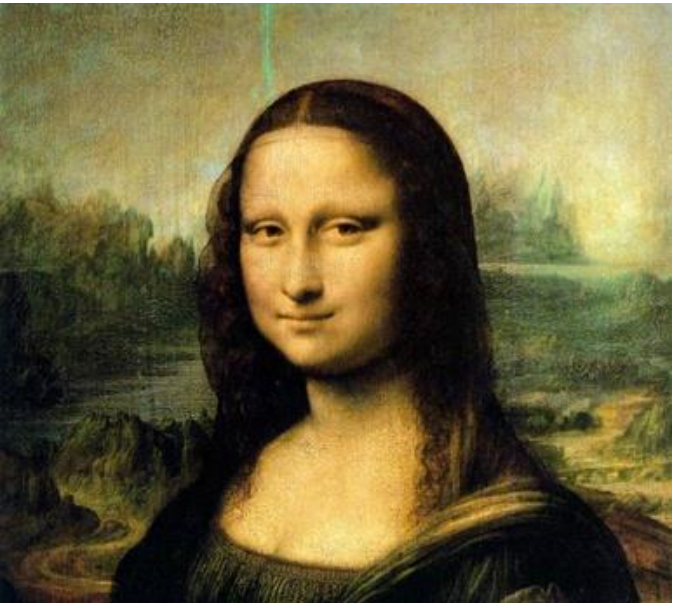
# Example



Gradient Visualization

Source: Evan Wallace

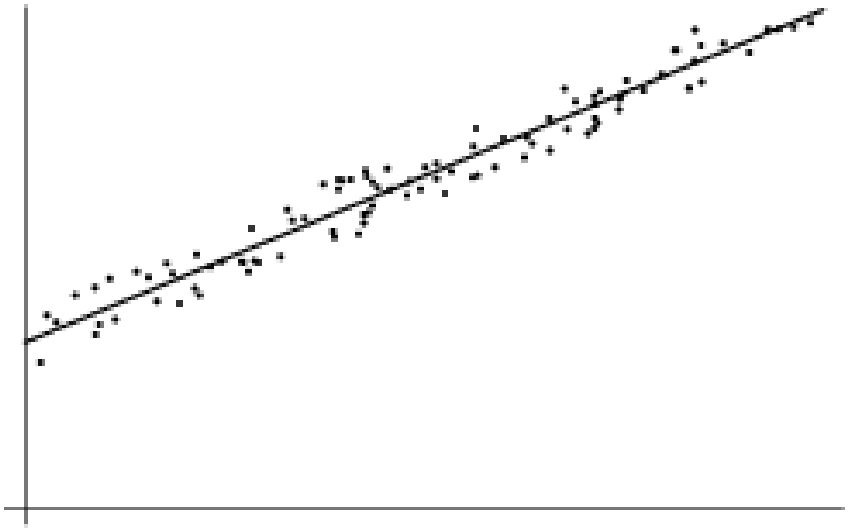
+



Source: Evan Wallace

# Gradient-domain editing

Creation of image = least squares problem in terms of: 1) pixel intensities; 2) differences of pixel intensities



Least Squares Line Fit in 2 Dimensions

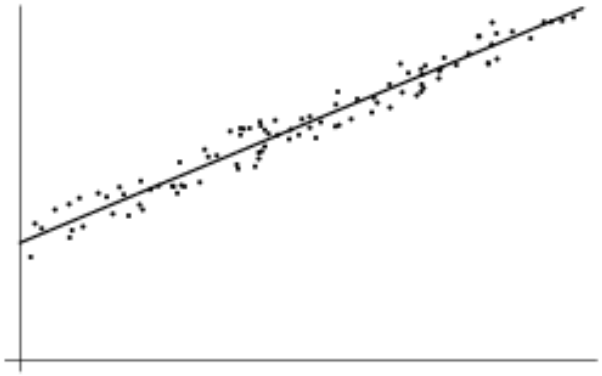
$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \sum_i \left(\mathbf{a}_i^T \mathbf{v} - b_i\right)^2$$

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \left(\mathbf{A}\mathbf{v} - \mathbf{b}\right)^2$$

Use Matlab least-squares solvers for numerically stable solution with sparse A

# Examples

1. Line-fitting: y=mx+b



# Examples

## 2. Gradient domain processing

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \sum_{i \in S, j \in N_i \cap S} ((v_i - v_j) - (s_i - s_j))^2 + \sum_{i \in S, j \in N_i \cap \neg S} ((v_i - t_j) - (s_i - s_j))^2$$

source image

| | | | |
|---|---|---|---|
| 1 **20** | 5 **20** | 9 **20** | 13 **20** |
| 2 **20** | 6 **80** | 10 **20** | 14 **20** |
| 3 **20** | 7 **20** | 11 **80** | 15 **20** |
| 4 **20** | 8 **20** | 12 **20** | 16 **20** |

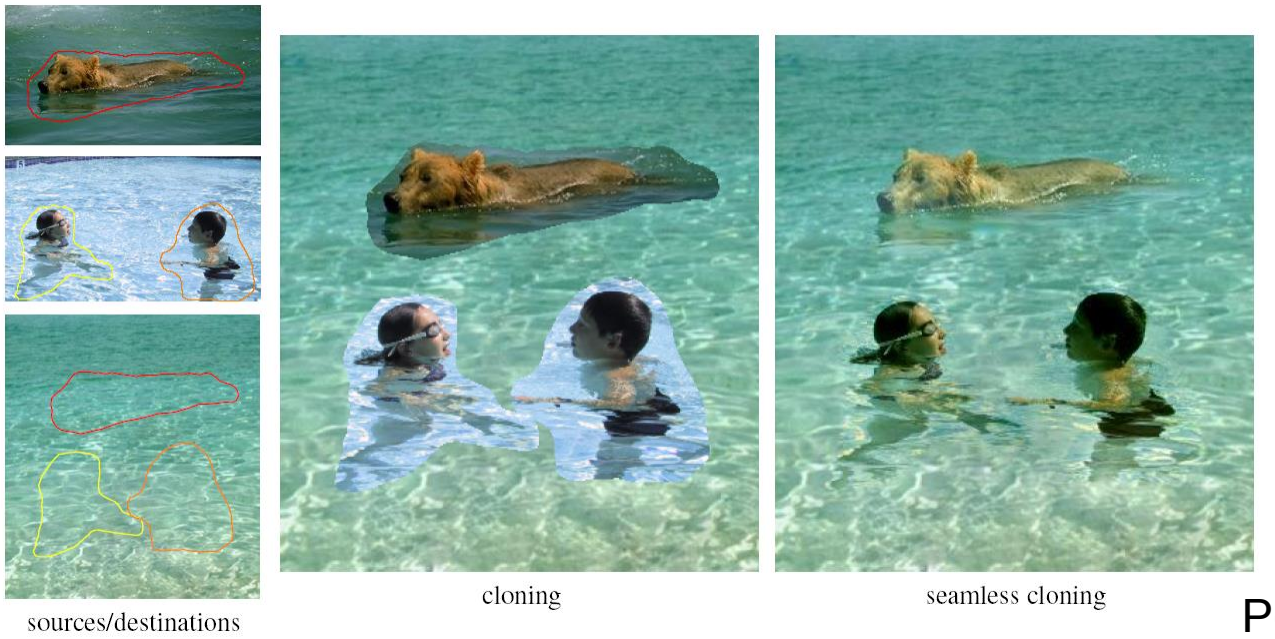
background image

| | | | |
|---|---|---|---|
| 1 **10** | 5 **10** | 9 **10** | 13 **10** |
| 2 **10** | 6 **10** | 10 **10** | 14 **10** |
| 3 **10** | 7 **10** | 11 **10** | 15 **10** |
| 4 **10** | 8 **10** | 12 **10** | 16 **10** |

target image

| | | | |
|---|---|---|---|
| 1 **10** | 5 **10** | 9 **10** | 13 **10** |
| 2 **10** | 6 $\mathbf{v_1}$ | 10 $\mathbf{v_3}$ | 14 **10** |
| 3 **10** | 7 $\mathbf{v_2}$ | 11 $\mathbf{v_4}$ | 15 **10** |
| 4 **10** | 8 **10** | 12 **10** | 16 **10** |

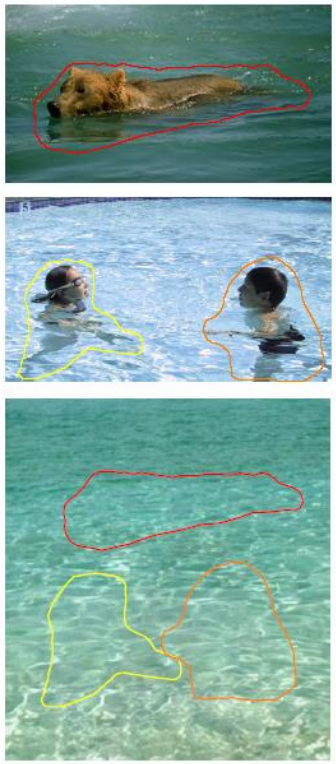
# Other results


Perez et al. 2003

# What do we lose?

- Foreground color changes
- Background pixels in target region are replaced



sources/destinations

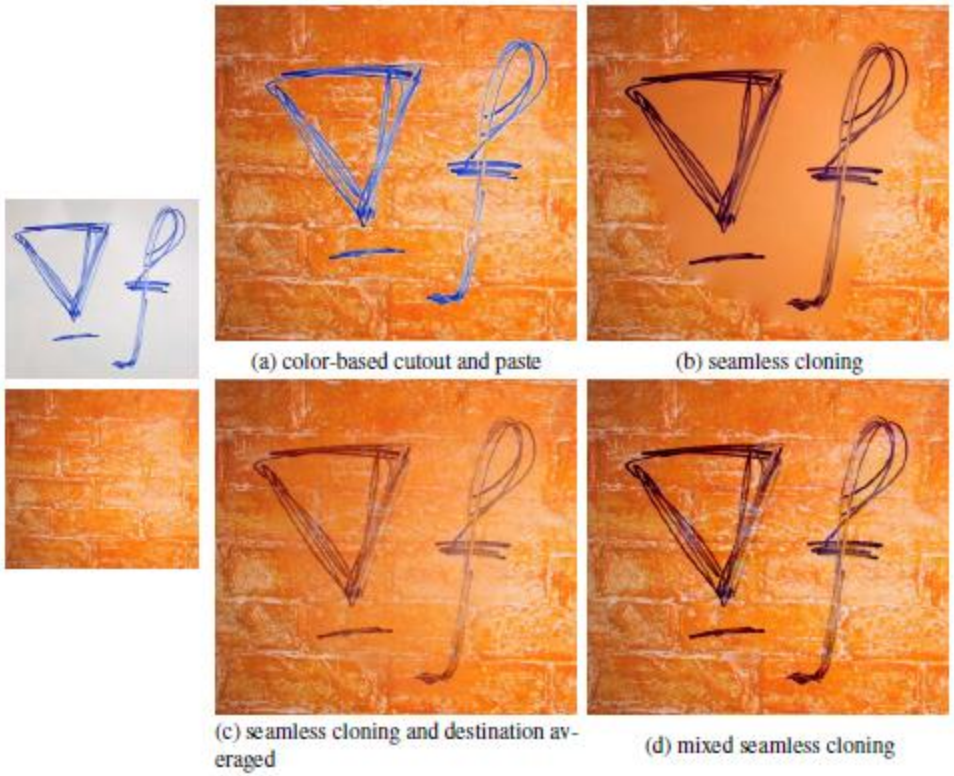
cloning

seamless cloning

Perez et al. 2003

# Blending with Mixed Gradients

- Use foreground or background gradient with larger magnitude as the guiding gradient



(a) color-based cutout and paste

(b) seamless cloning

(c) seamless cloning and destination averaged

(d) mixed seamless cloning

Perez et al. 2003

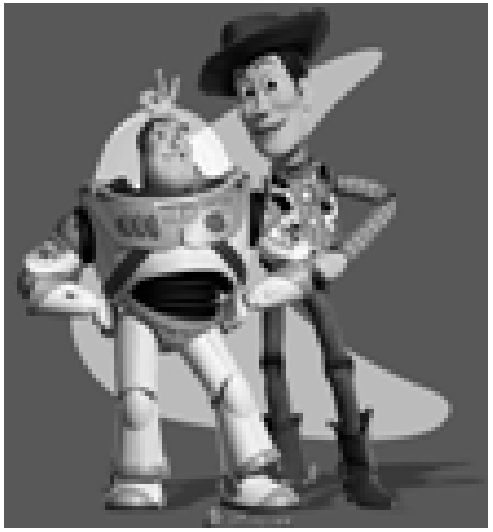
# Project 3: Gradient Domain Editing

General concept: Solve for pixels of new image that satisfy constraints on the gradient and the intensity

- Constraints can be from one image (for filtering) or more (for blending)

# Project 3: Reconstruction from Gradients

1. Preserve x-y gradients
2. Preserve intensity of one pixel



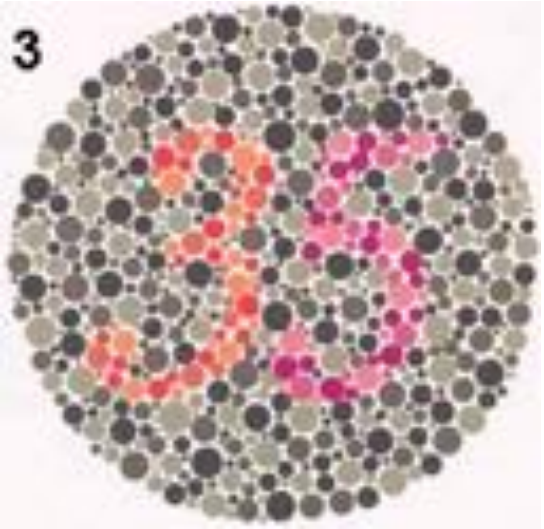
Source pixels: s

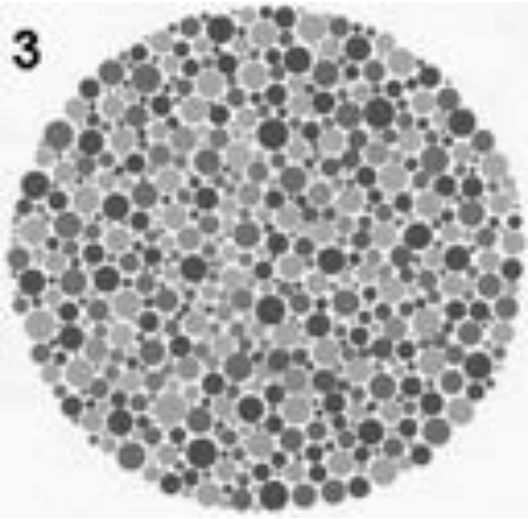
Variable pixels: v

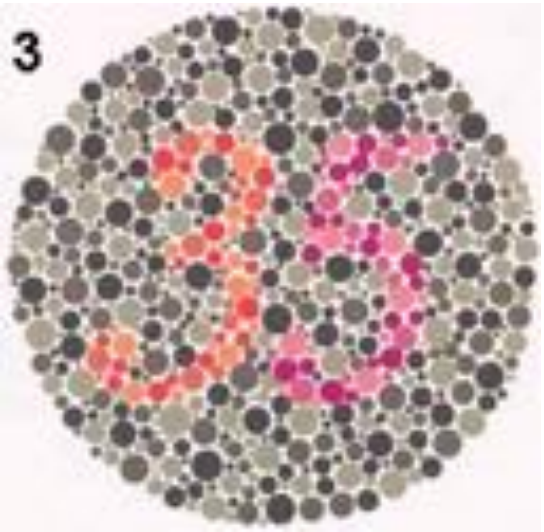
1. minimize (v(x+1,y)-v(x,y) - (s(x+1,y)-s(x,y))^2
2. minimize (v(x,y+1)-v(x,y) - (s(x,y+1)-s(x,y))^2
3. minimize (v(1,1)-s(1,1))^2

# Project 3 (extra): Color2Gray

rgb2gray




Gradient-domain editing

?

# Project 3 (extra): NPR

- Preserve gradients on edges
  - e.g., get canny edges with edge(im, ‘canny’)
- Reduce gradients not on edges
- Preserve original intensity

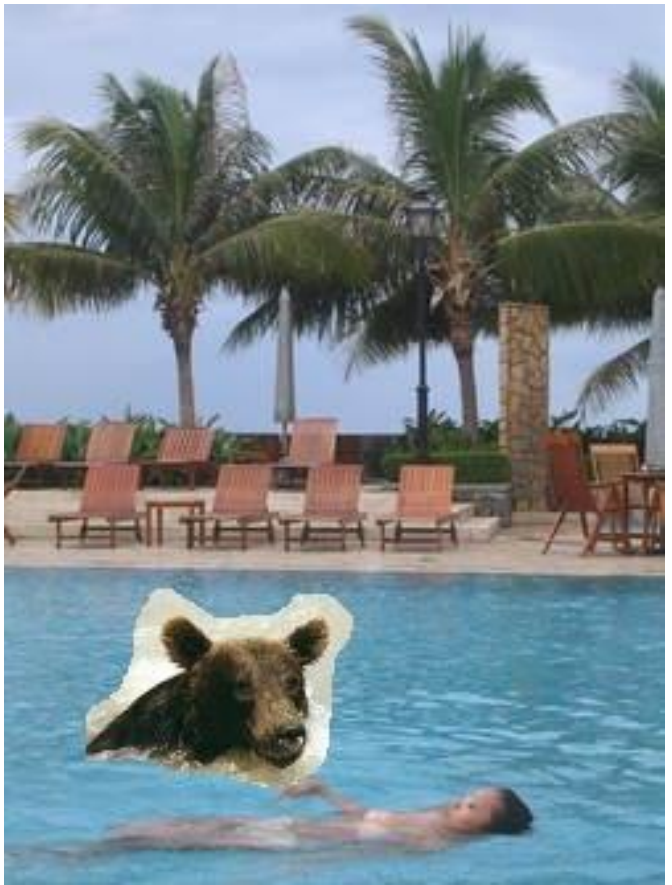
Perez et al. 2003

# Things to remember

- Three ways to blend/composite
  1. Alpha compositing
     - Need nice cut (intelligent scissors)
     - Should **feather**
  2. Laplacian pyramid blending
     - **Smooth blending at low frequencies, sharp at high frequencies**
     - Usually used for stitching
  3. Gradient domain editing
     - Also called **Poisson Editing**
     - Explicit control over what to preserve
     - Changes foreground color (for better or worse)
     - Applicable for many things besides blending

# Take-home questions

1) I am trying to blend this bear into this pool. What problems will I have if I use:
   a) Alpha compositing with feathering
   b) Laplacian pyramid blending
   c) Poisson editing?



# Take-home questions

2) How would you make a sharpening filter using gradient domain processing? What are the constraints on the gradients and the intensities?

# Next class

- Image warping: affine, projective, rotation, etc.