Image blending

15-463, 15-663, 15-862
Computational Photography
Fall 2017, Lecture 7

http://graphics.cs.cmu.edu/courses/15-463

Course announcements

- September 27th lecture tentatively rescheduled for

  Friday 29th, 12:00-1:30pm (same time, different day)

  - Still looking for room.
  - Will announce on Piazza and update course website once room is confirmed.

- Homework 1 scores have been uploaded on Canvas.
  - Mean score: 102.
  - Median score: 100.

- If you haven’t started Homework 2 yet, you should.

Overview of today's lecture

- Some motivating examples.
- Cut-and-paste.
- Alpha (linear) blending.
- Multi-band blending.
- Poisson blending.

Slide credits

Most of these slides were adapted from:

- Kris Kitani (15-463, Fall 2016).

Some slides were inspired or taken from:

- Fredo Durand (MIT).
- James Hays (Georgia Tech).

Some motivating examples

Gangster, Frankie Yale, killed by a drive-by in Brooklyn in 1928.

A tragic photo from 1959 after three-year-old Martha Cartagena was killed while riding her tricycle in Brooklyn

In 1958 there was a fatal fire at the Elkins Paper & Twine Co. on Wooster Street in SoHo. The building burned to the ground.


MEDIMAX RINGCENTER
MEDIMAX
(c)Sergey Larenkov

Berlin 65 years later

Berlin, 1945/2010, Mehringdamm

Forrest Gump (1994)

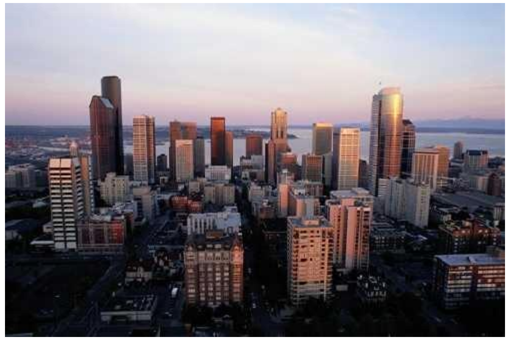
Techniques for compositing

- Cut-and-paste.
- Alpha (linear) blending.
- Multi-band blending.
- Poisson blending.
- Seam stitching (next lecture).

Cut-and-paste

Cut and paste procedure

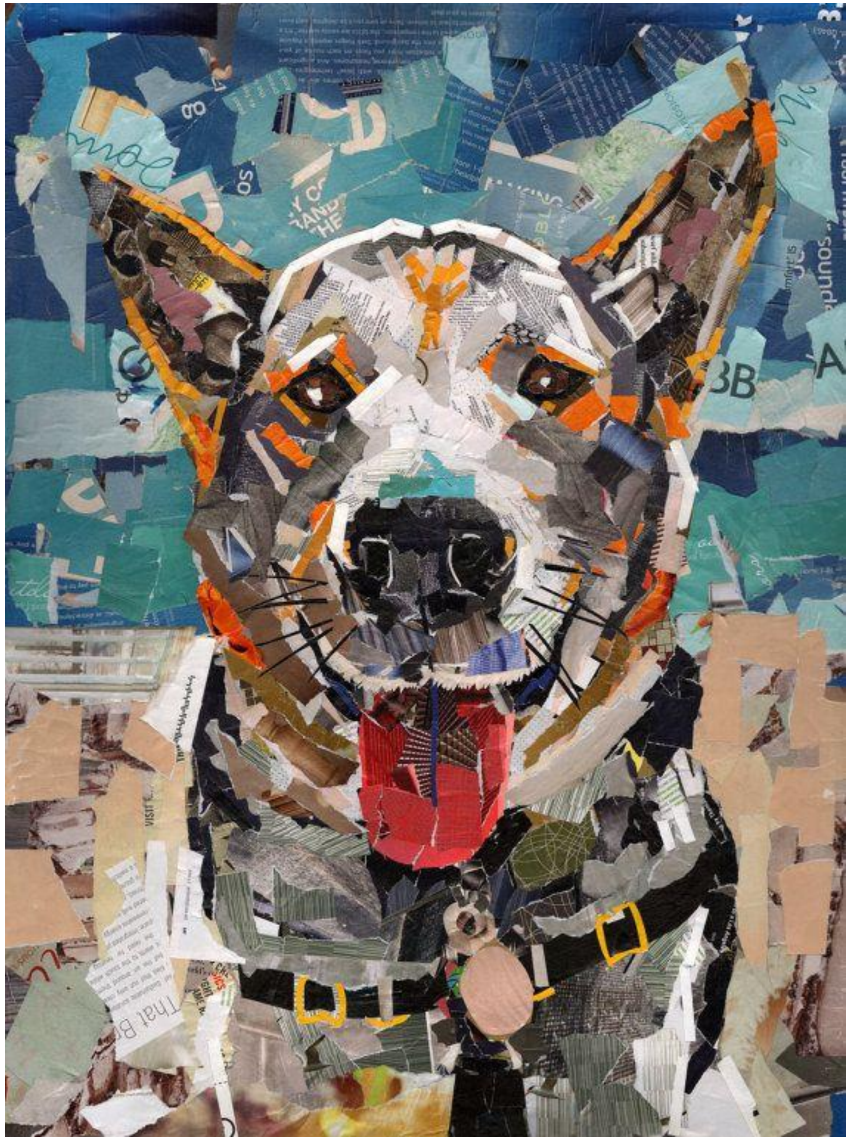
1. Extract Sprites (e.g., using *Intelligent Scissors* in Photoshop)



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

You may have also heard it as collaging

Cut and paste



Sometimes it produces visually compelling results.

Cut and paste

Other times, not so much.

What is wrong with this composite?

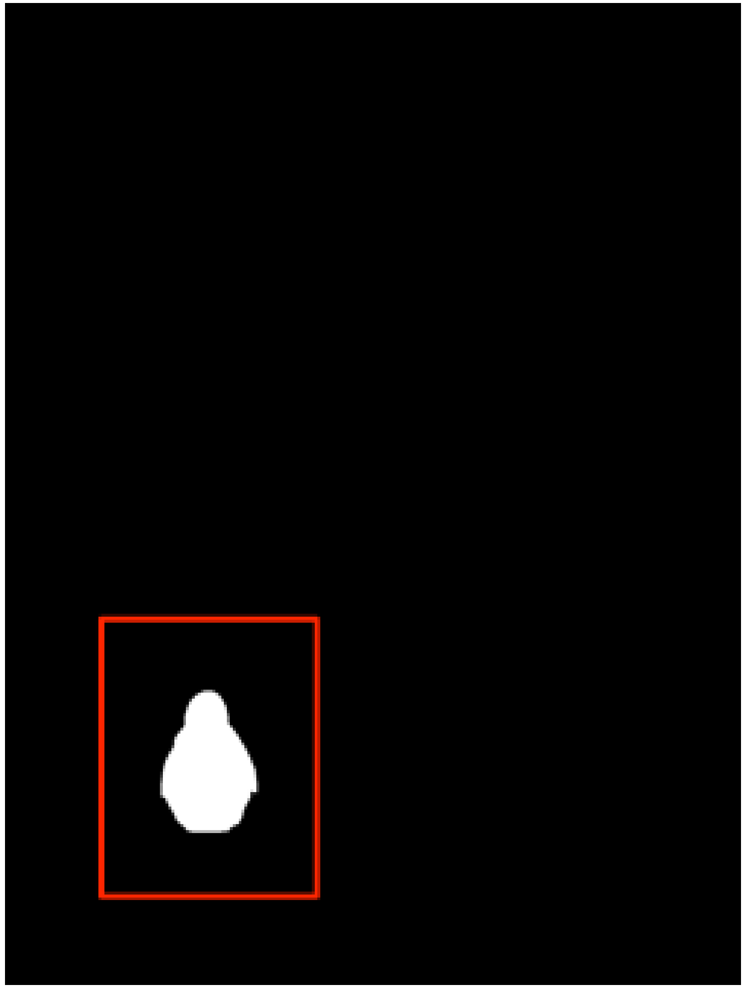
Alpha (linear) blending

Alpha blending

foreground

background

mask

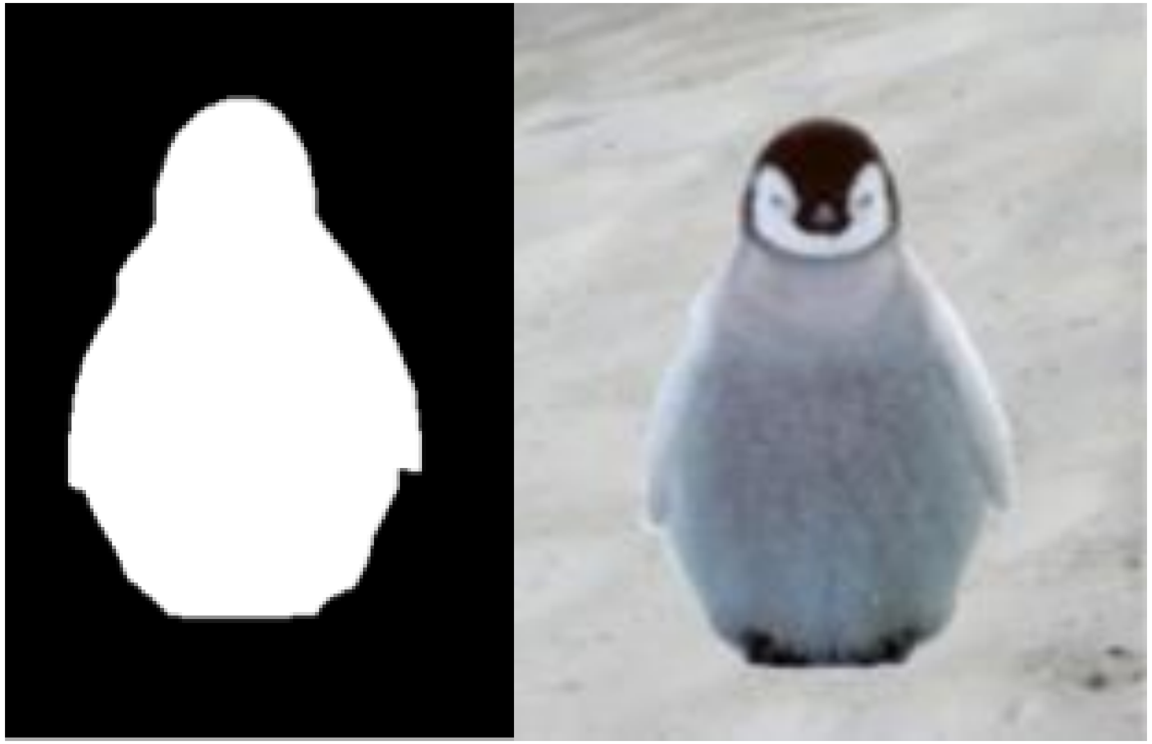


output

a.k.a. alpha matte
or alpha composite

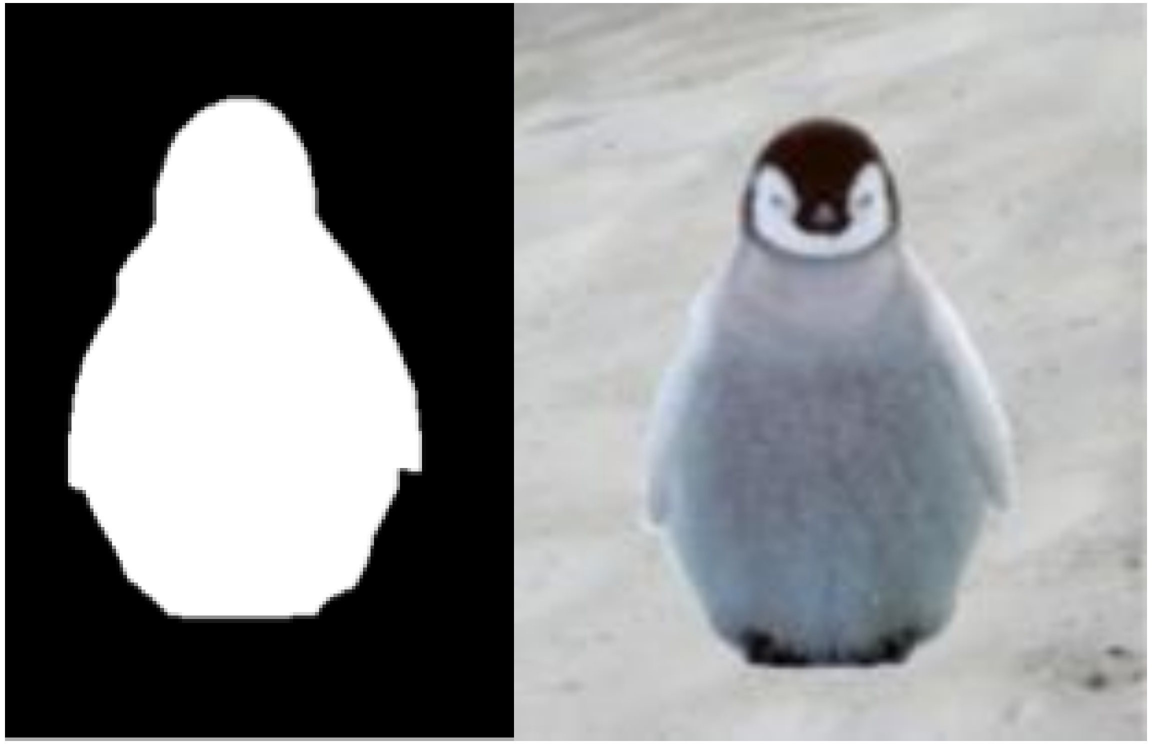
output = foreground * mask + background * (1-mask)

Binary alpha mask



Does this look unnatural?

Binary alpha mask

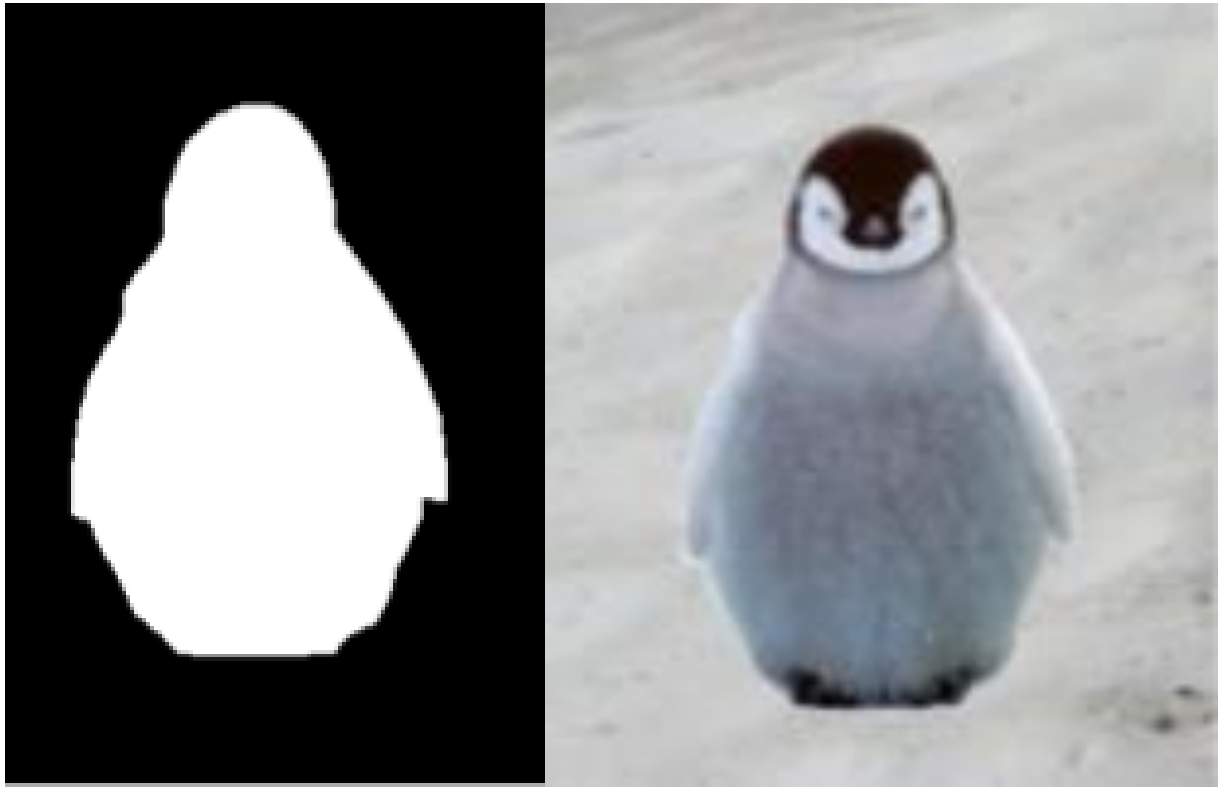


Does this look unnatural?

How can we fix it?

Non-binary alpha mask

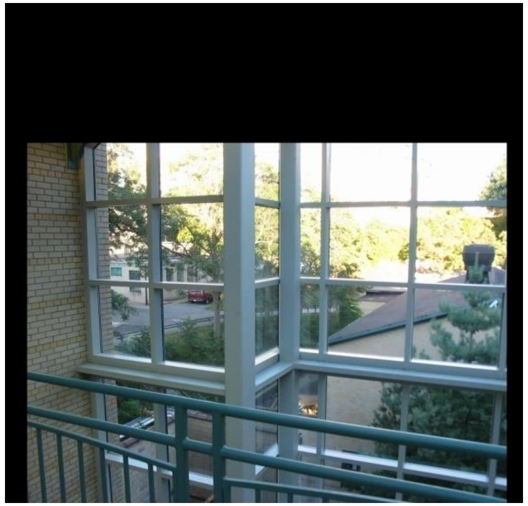
binary alpha mask



feathering (smoothed alpha)

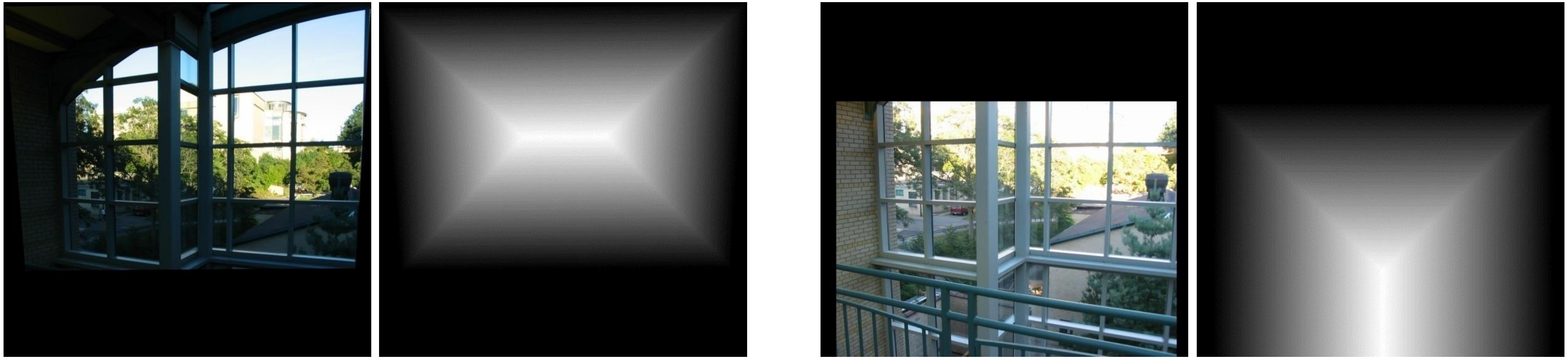
How would you implement feathering?

Setting the alpha mask: center seam



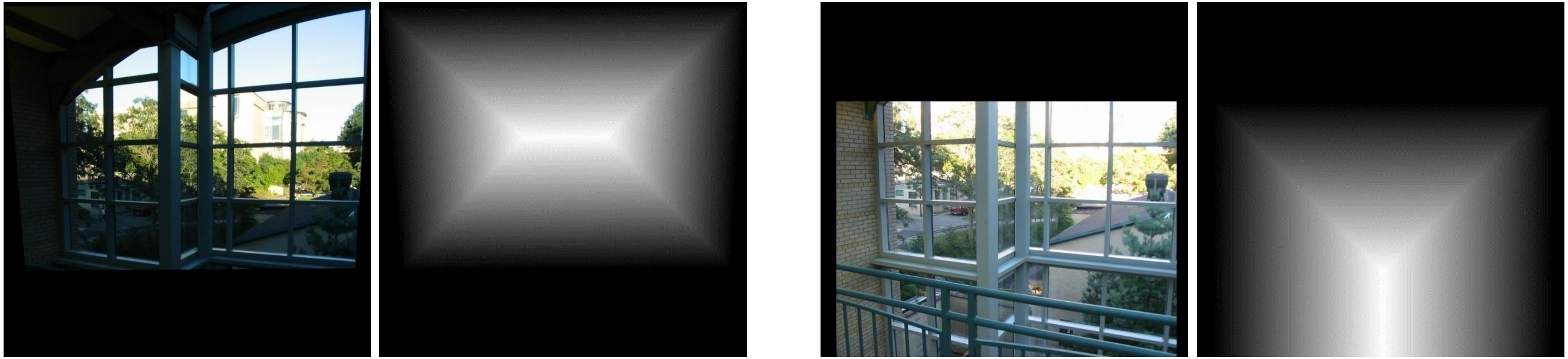
How would you create a binary alpha mask for these two images?

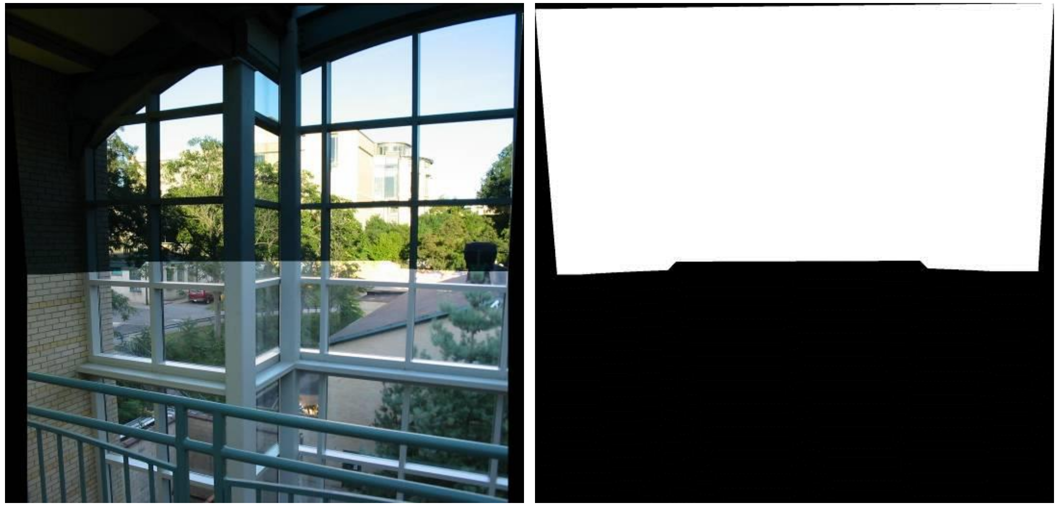
Setting the alpha mask: center seam



Step 1: Compute their distance transform (`bwdist`)

Setting the alpha mask: center seam





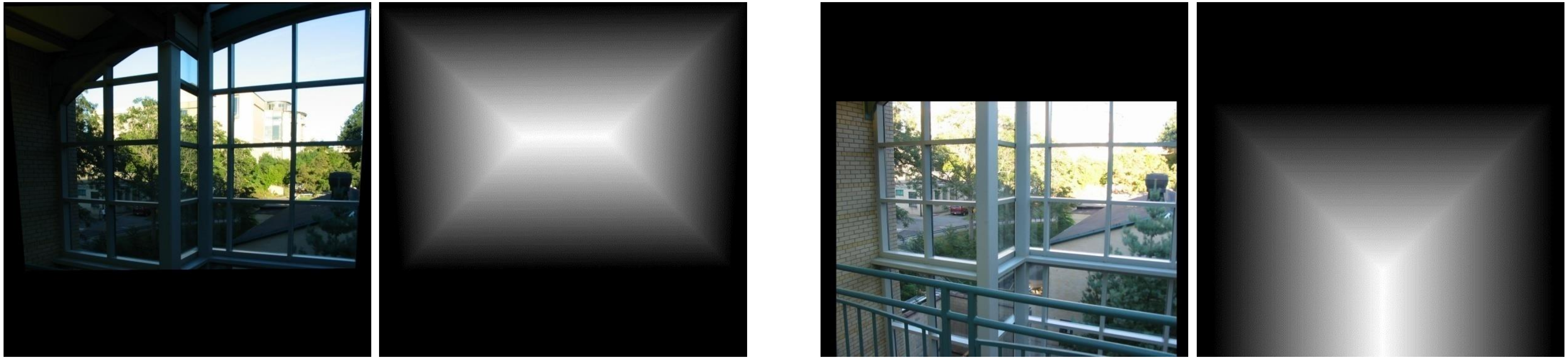
Step 2: set mask

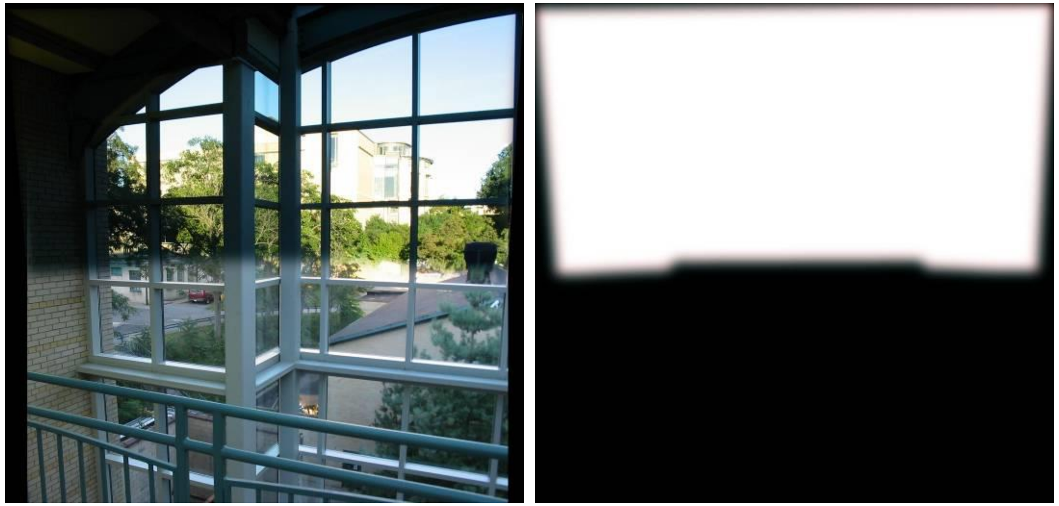
```
alpha = logical(dtrans1>dtrans2)
```

Setting the alpha mask: center seam

Anything wrong with this alpha matte?

Setting the alpha mask: center seam





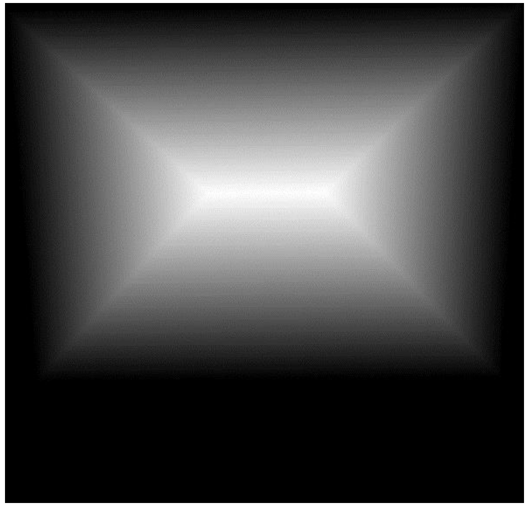
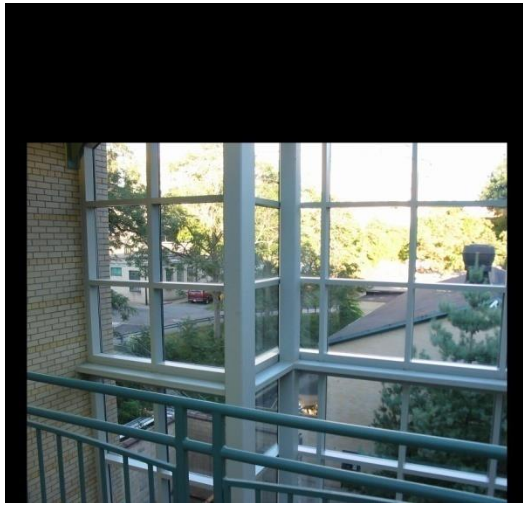
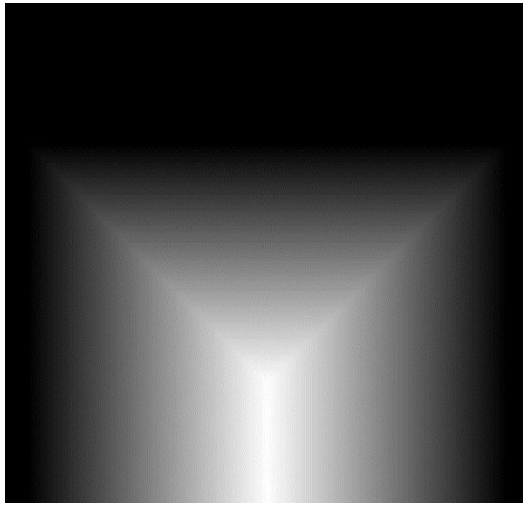
Step 3: blur the mask

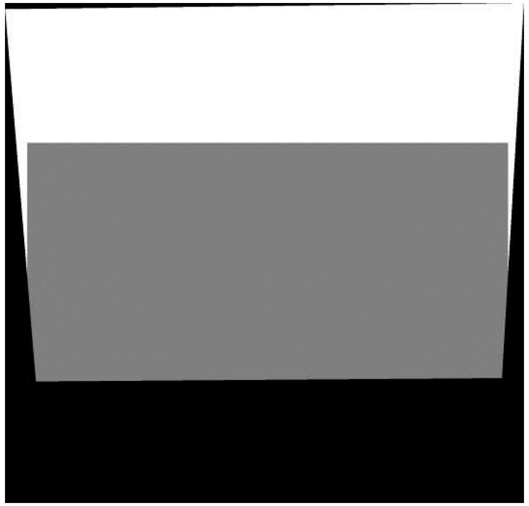
```
alpha = blur(alpha)
```

Setting the alpha mask: center seam

Still doesn't look terribly good

Setting the alpha mask: center seam

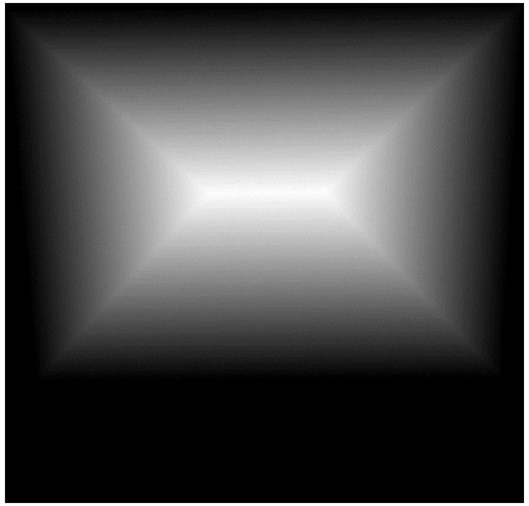
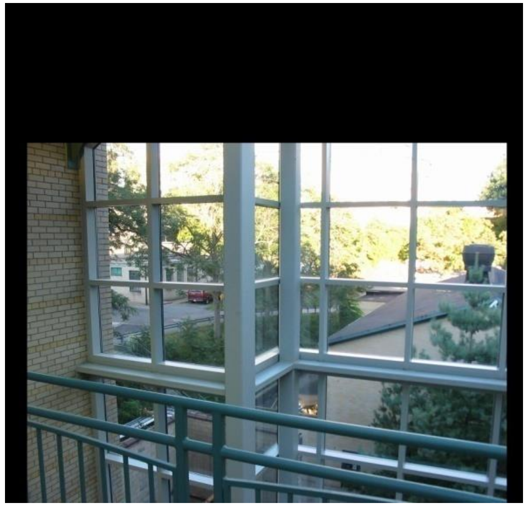
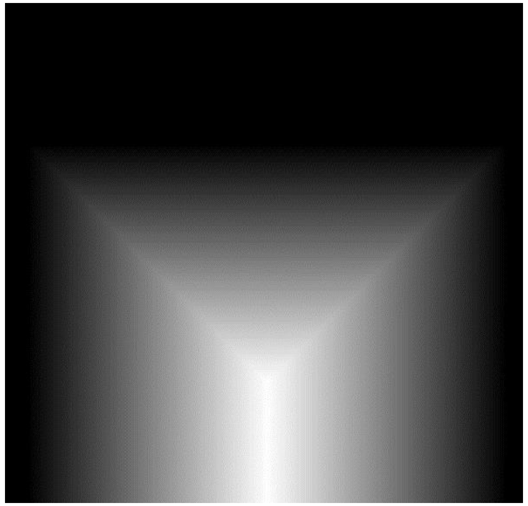
Step 4: go beyond blurring for non-binary

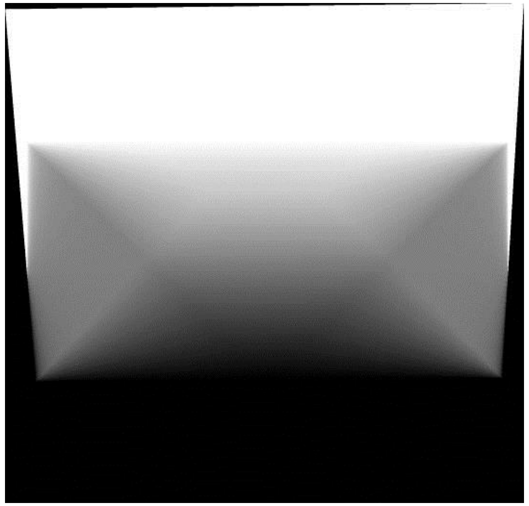
`alpha = 0.5` in overlap region

Setting the alpha mask: center seam

Still not OK

Setting the alpha mask: center seam

Step 5: more elaborate non-binary

```
alpha = dtrans1 / (dtrans1+dtrans2)
```

Setting the alpha mask: center seam

Looks better but some dangers remain.

Another blending example

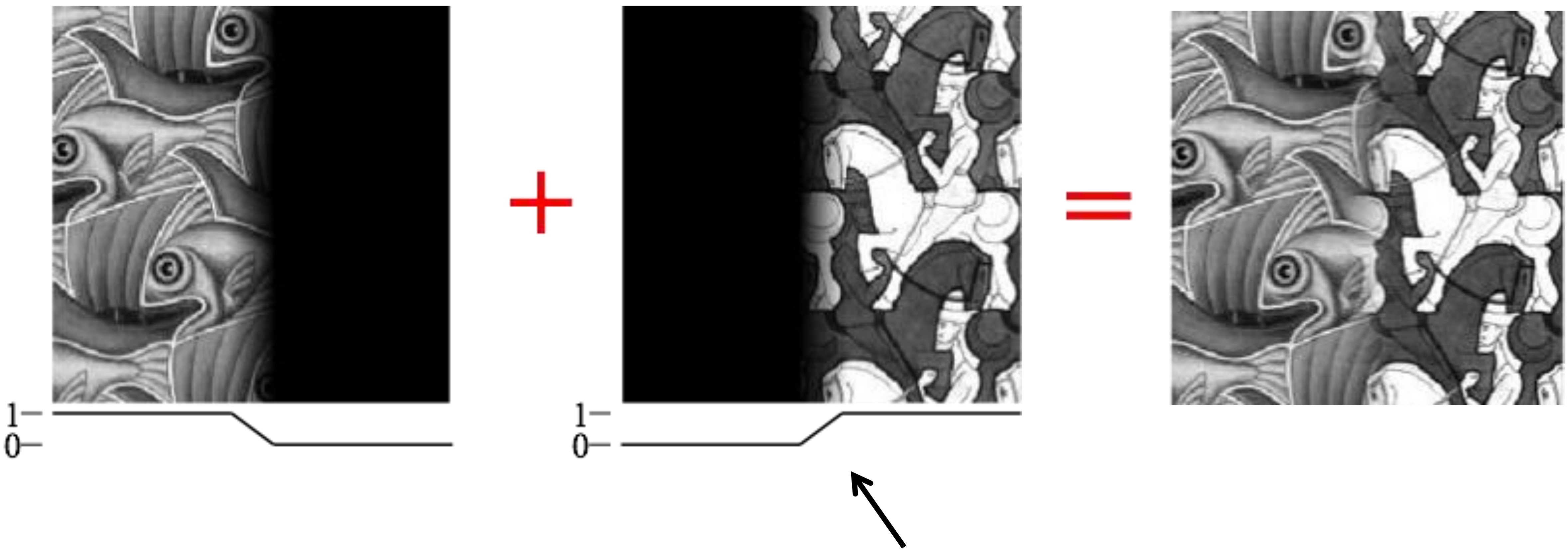
Let’s blend these two images...

left side

right side

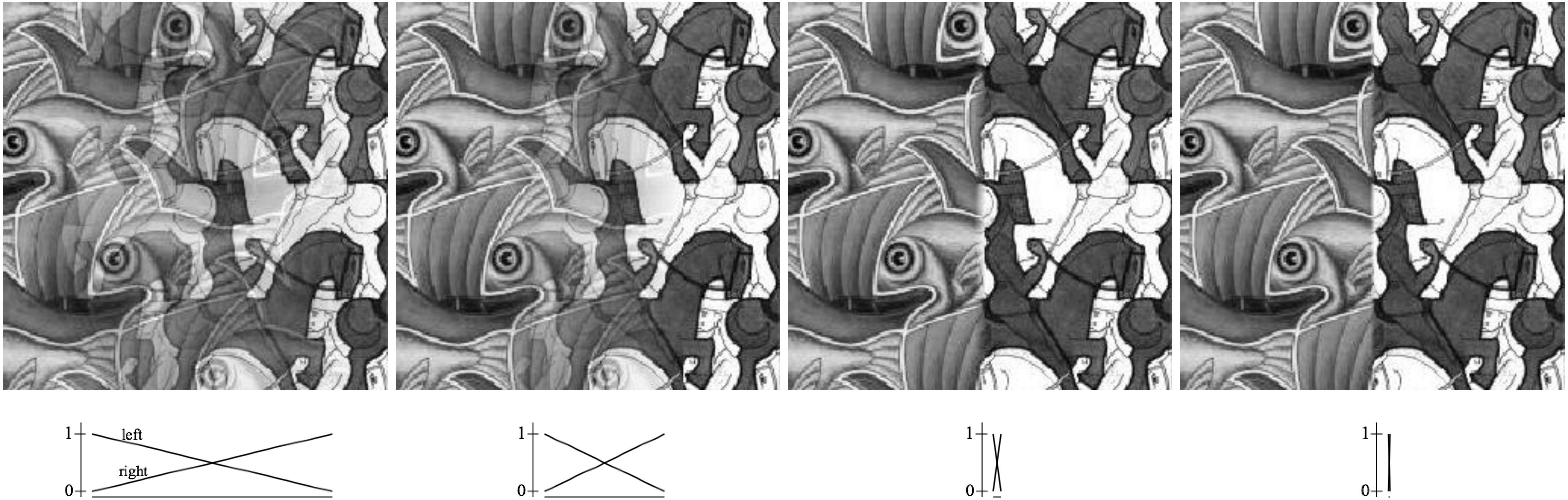
What kind of mask would you use?

Another blending example



How would you select this window?

Effects of different windows

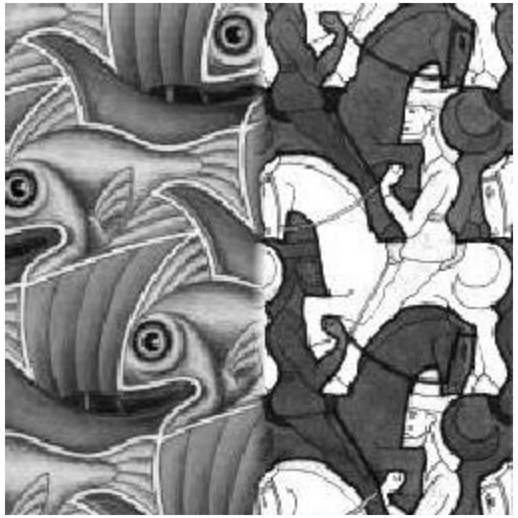


Bad windows: ghosting.

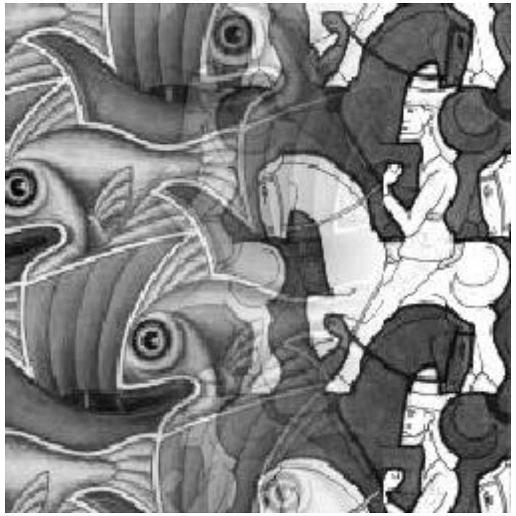
Good window: smooth but no ghosting.

Bad window: non-smooth seam.

What is a good window size?

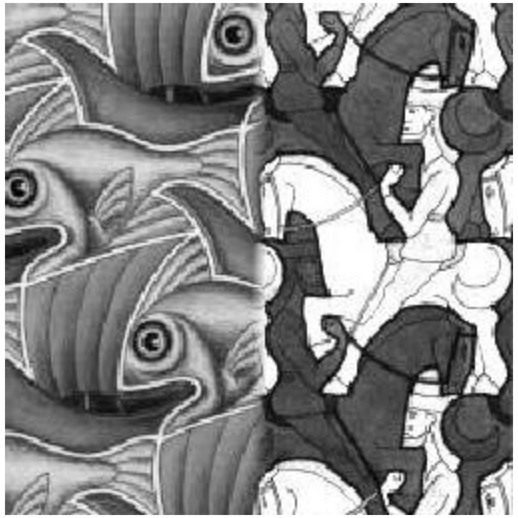


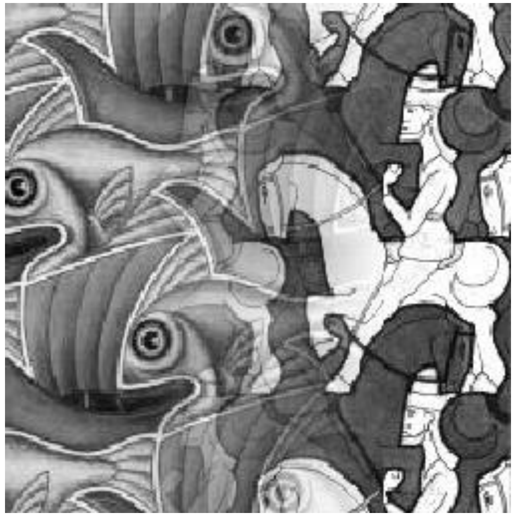
To avoid discontinuities:
window = size of largest prominent feature



To avoid ghosting:
window <= 2*size of smallest prominent feature

What is a good window size?





Fourier domain interpretation:

To avoid discontinuities:
window = size of largest prominent feature

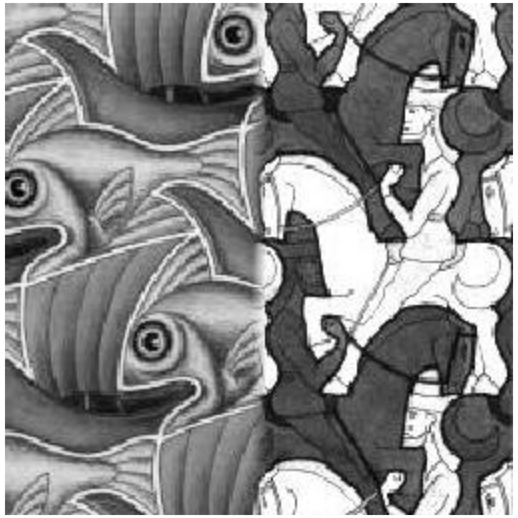
linear blending should work when:
image frequency content occupies roughly one “octave” (power of two)

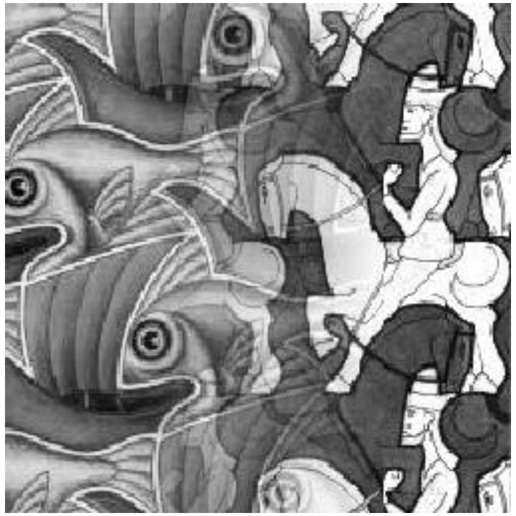
To avoid ghosting:
window <= 2*size of smallest prominent feature

linear blending should work when:
largest frequency <= 2*size of smallest frequency

What if the frequency spread is too wide?

What is a good window size?





Fourier domain interpretation:

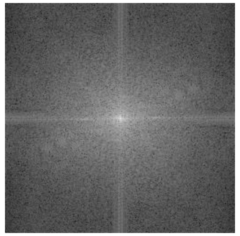
To avoid discontinuities:
window = size of largest prominent feature

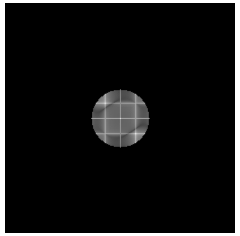
linear blending should work when:
image frequency content occupies roughly one “octave” (power of two)

To avoid ghosting:
window <= 2*size of smallest prominent feature

linear blending should work when:
largest frequency <= 2*size of smallest frequency

Most natural images have a very wide frequency spread. What do we do then?

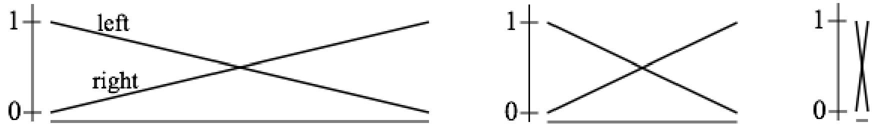




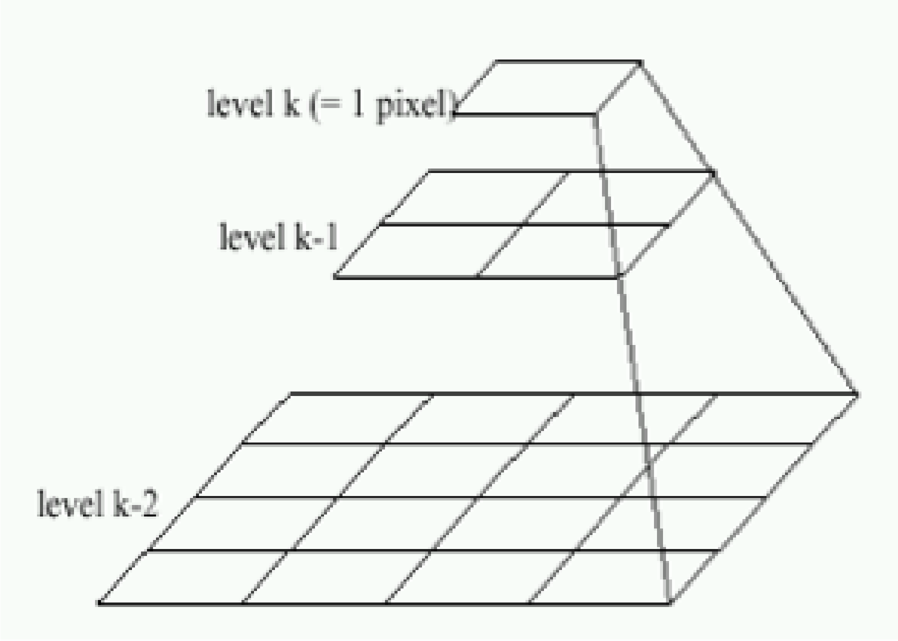
Multi-band blending

Time to use pyramids again

At low frequencies, blend slowly to avoid seams
At high frequencies, blend quickly to avoid ghosts



Which mask goes where?



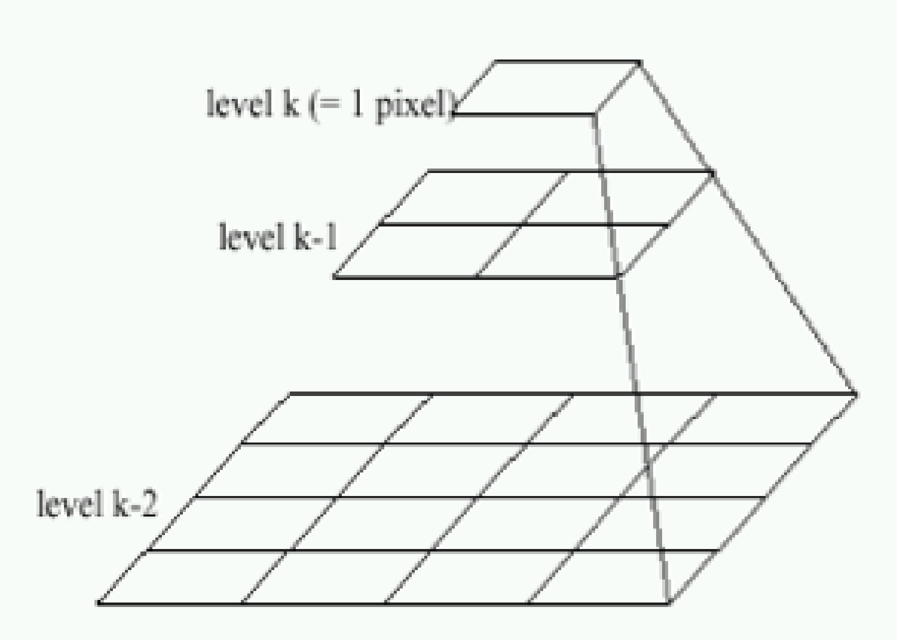
left image

?

?

?

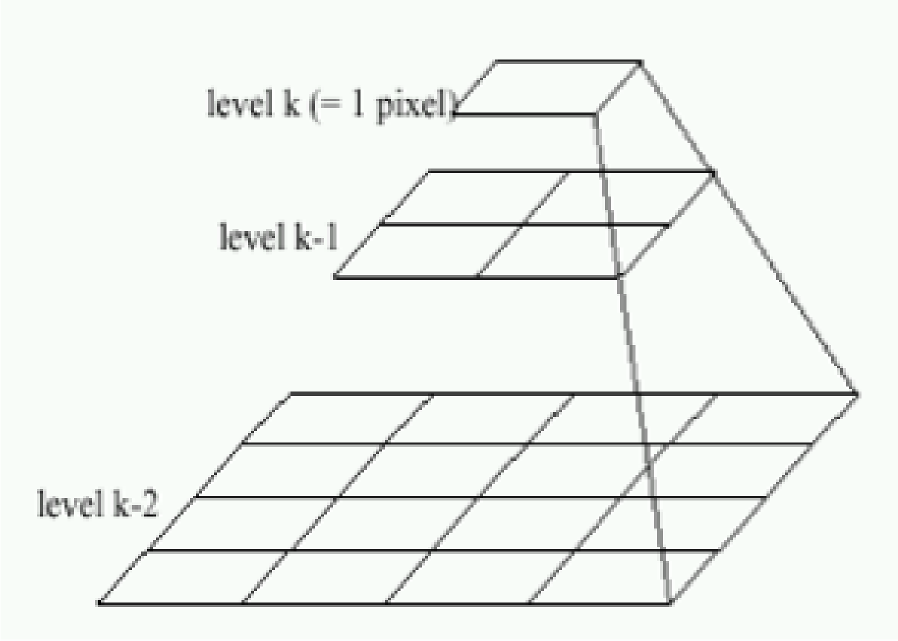
alpha mask



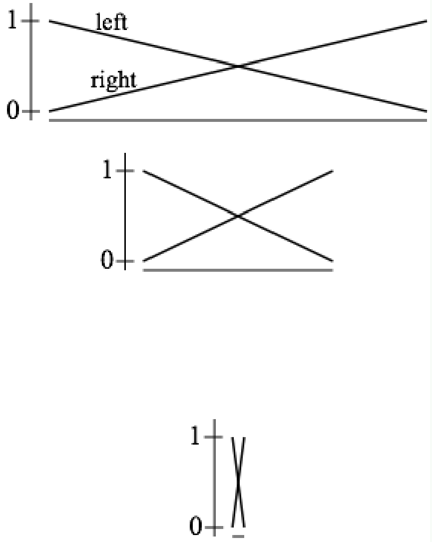
right image

Time to use pyramids again

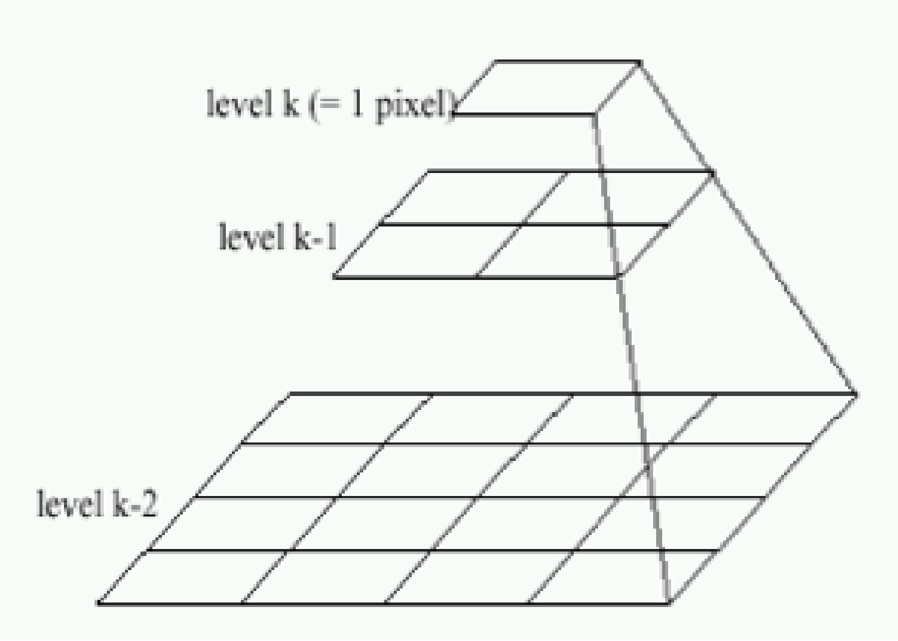
At low frequencies, blend slowly to avoid seams
At high frequencies, blend quickly to avoid ghosts



left image

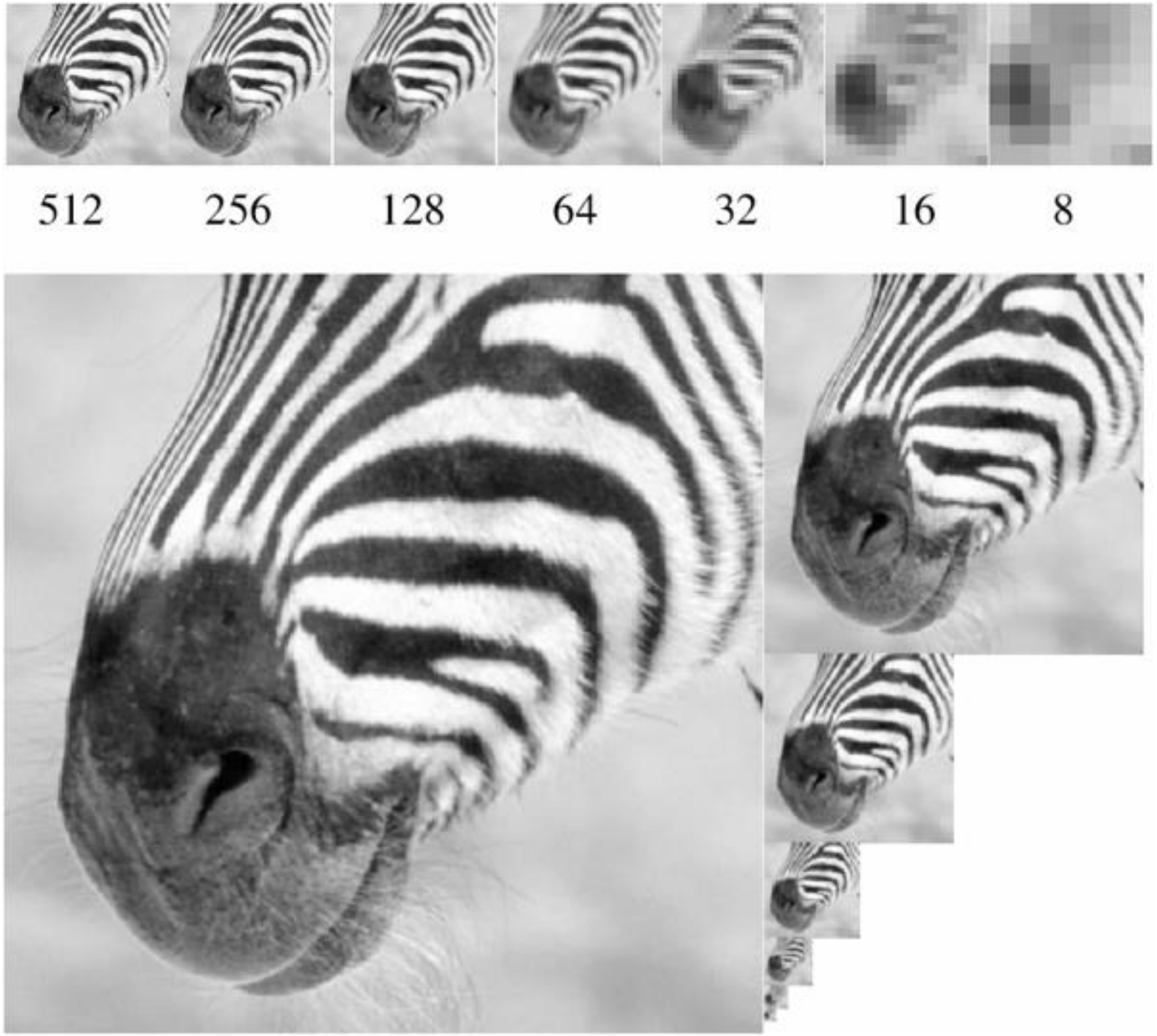


alpha mask

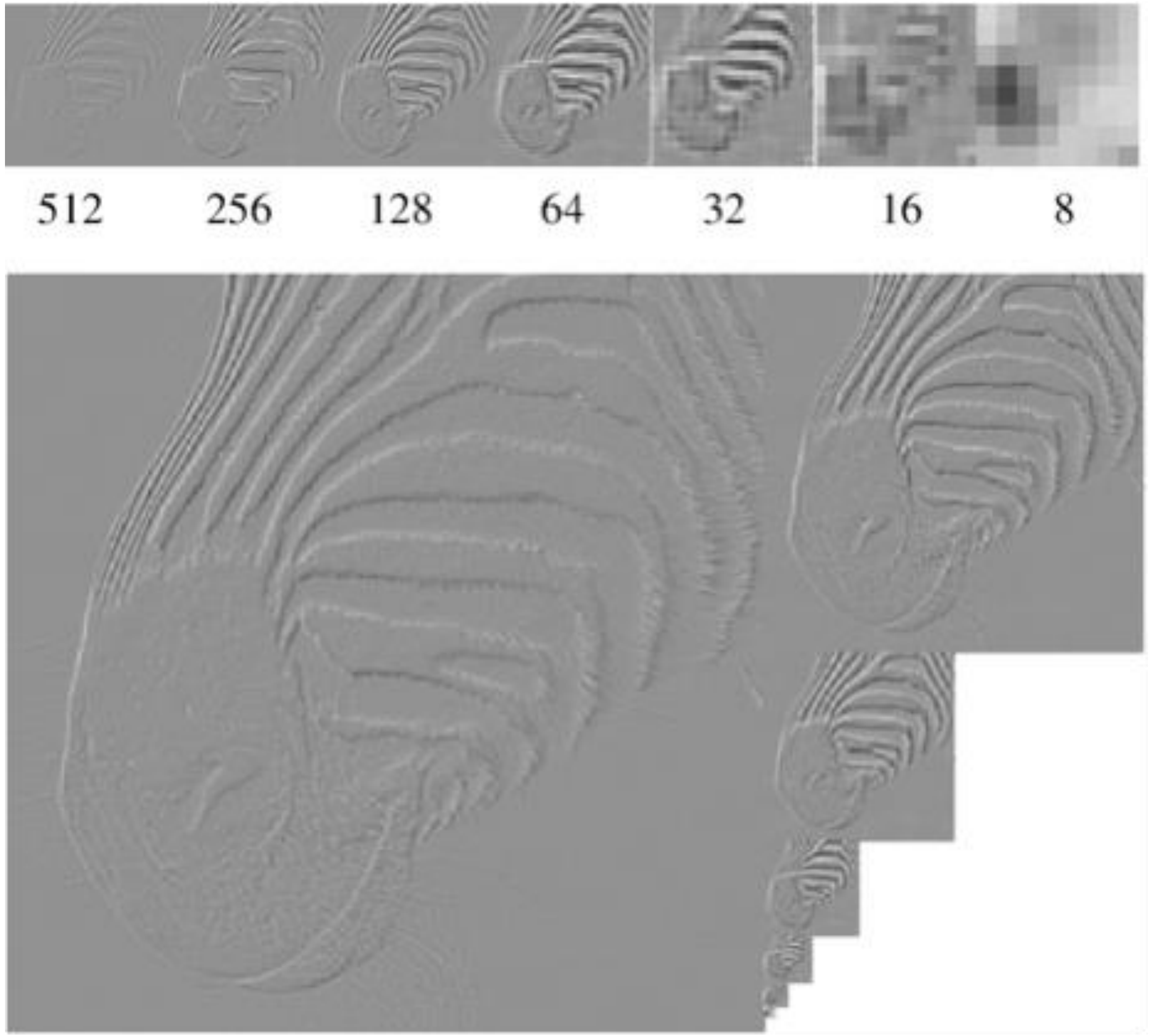


right image

Remember our two types of pyramids



Gaussian pyramid



Laplacian pyramid

Remember our two types of pyramids

1. Build Laplacian pyramids for each image

2. Blend each level of pyramid using region mask

$$L_{12}^i = L_1^i \cdot R^i + L_2^i \cdot (1 - R^i)$$

image 1 at level i — image 2 at level i — region mask at level i

How large should the blending region be at each level?

3. Collapse the pyramid to get the final blended image

Remember our two types of pyramids

1. Build Laplacian pyramids for each image

2. Blend each level of pyramid using region mask

$$L_{12}^i = L_1^i \cdot R^i + L_2^i \cdot (1 - R^i)$$

image 1 at level i | image 2 at level i | region mask at level i

How large should the blending region be at each level?

About the size of that level's blur

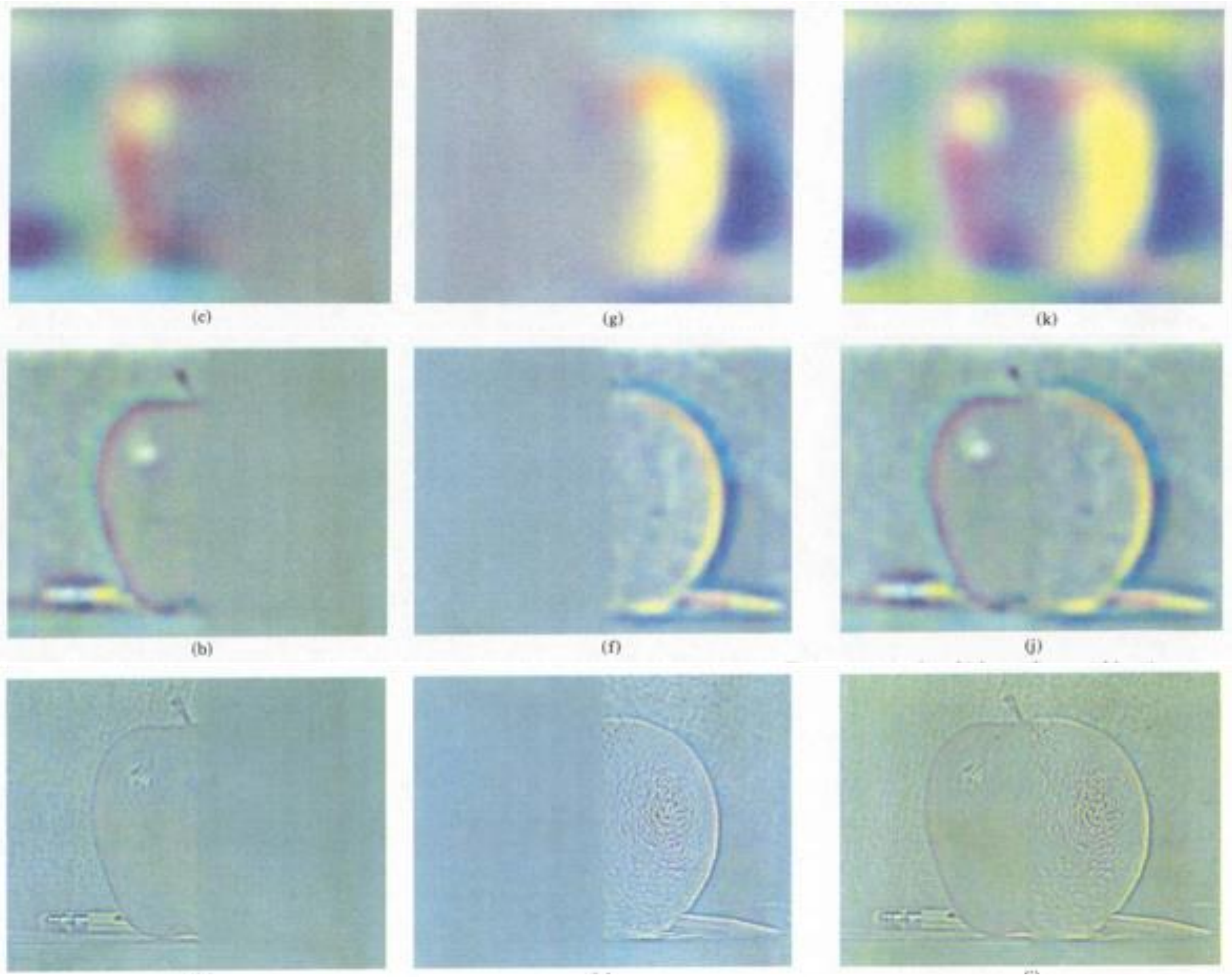
3. Collapse the pyramid to get the final blended image

Multi-band blending using the Laplacian pyramid

Laplacian level 4

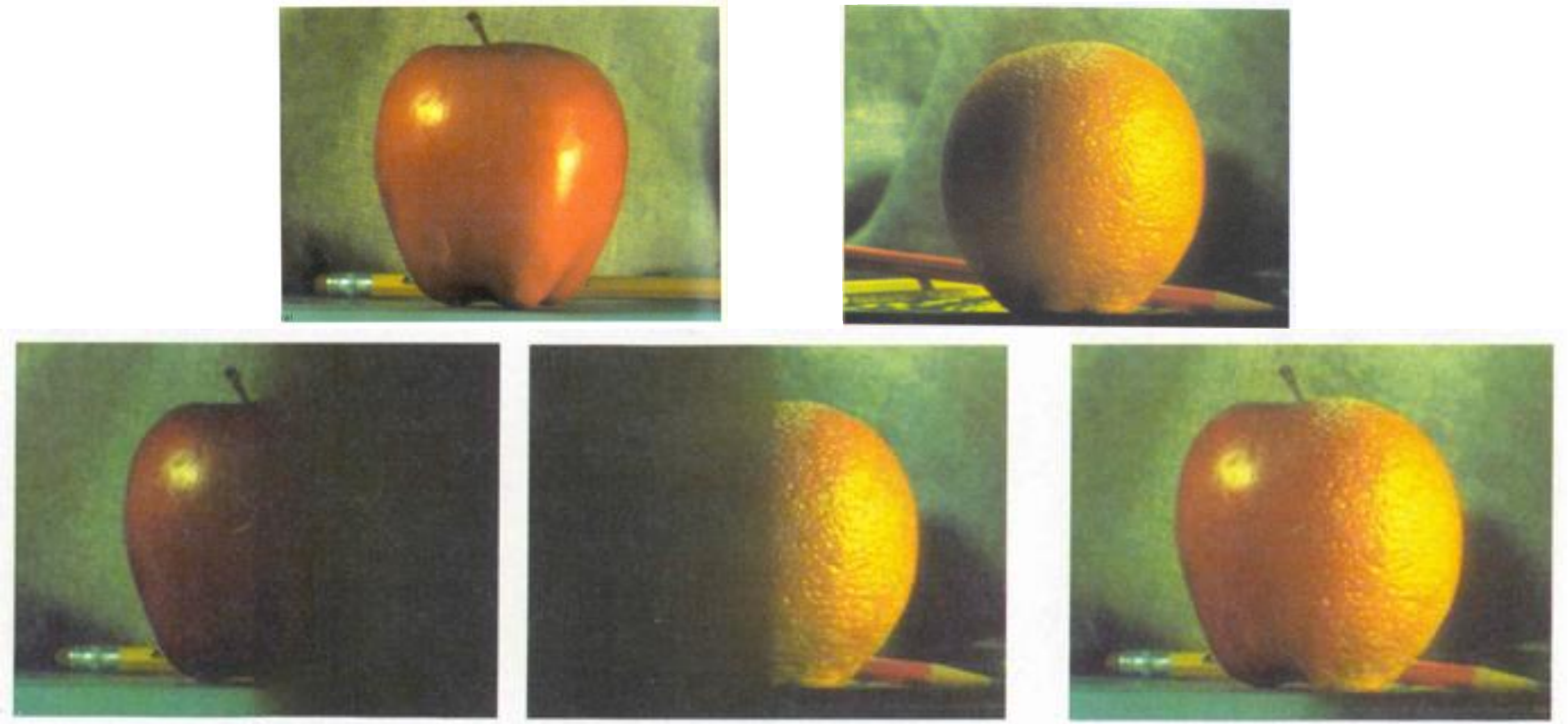
Laplacian level 2

Laplacian level 0



left pyramid right pyramid blended pyramid

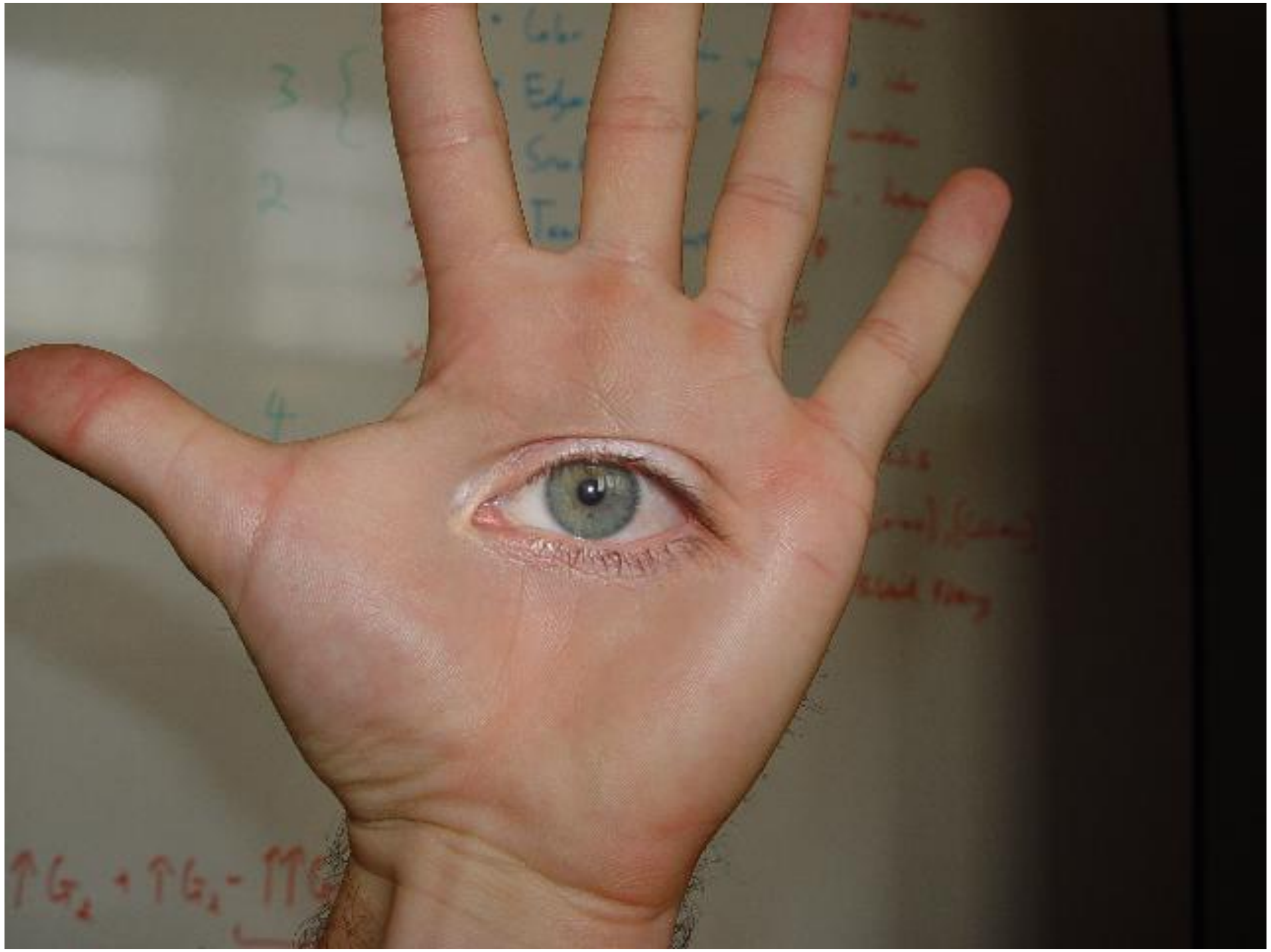
A famous result (for its time)



(d) (h) (l)

A famous result (for its time)

A creepier result

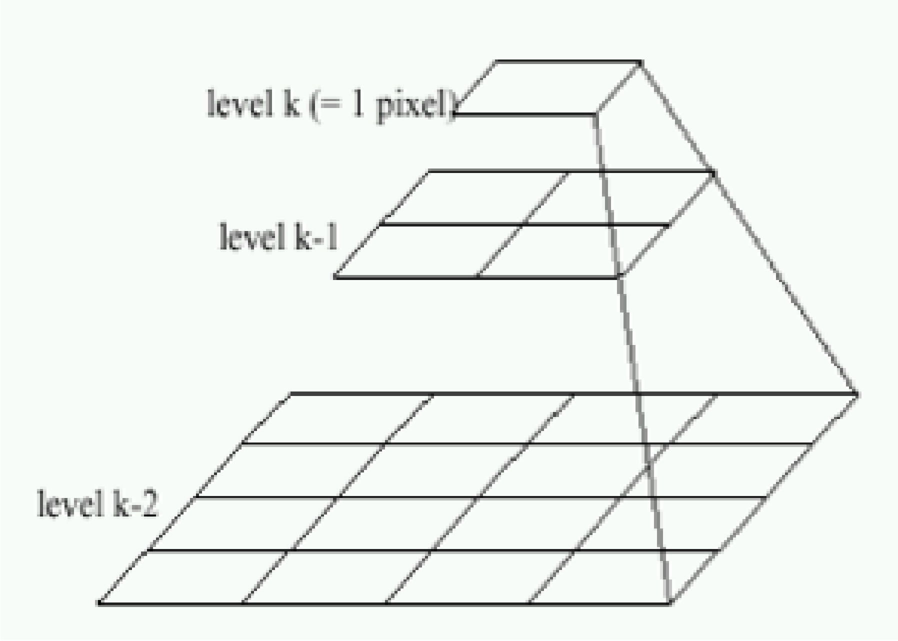


Can we get the same result with less computation?

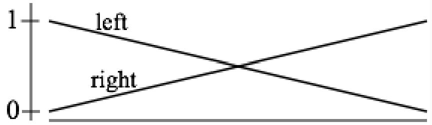
Two-band blending

Only use two bands: high frequency and low frequency

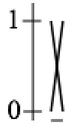
- Blend low frequency with smooth alpha
- Blend high frequency with binary alpha



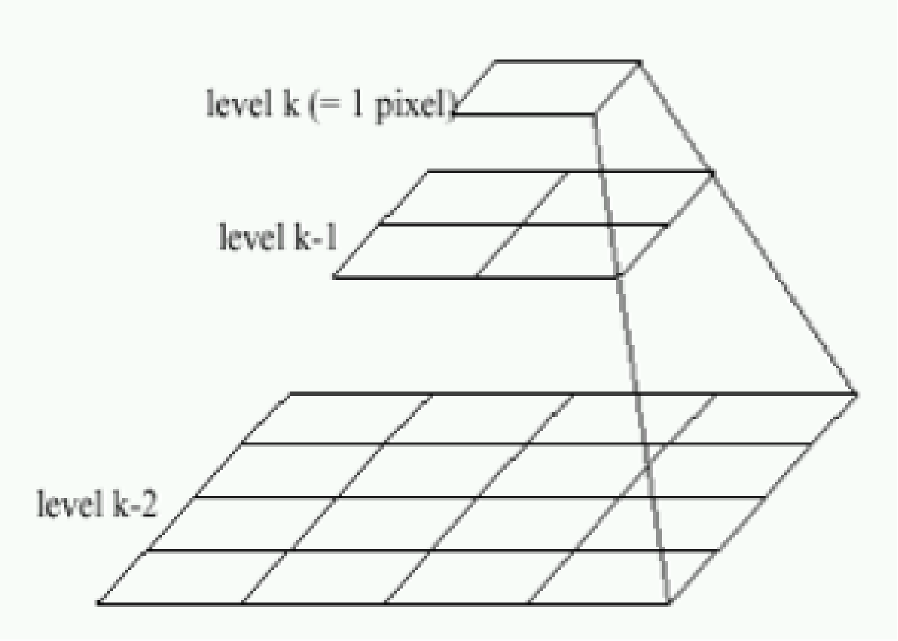
left image



do not use



alpha mask



right image

Example: blending panoramas

original collage

blended collage

Example: blending panoramas

low frequency blend

high frequency blend

Linear blending

Two-band blending

One more comparison

copy-paste | linear | two-band

Why do these images look weirdly cropped?

They were warped using homographies before being aligned.

Homework 6: autostitching

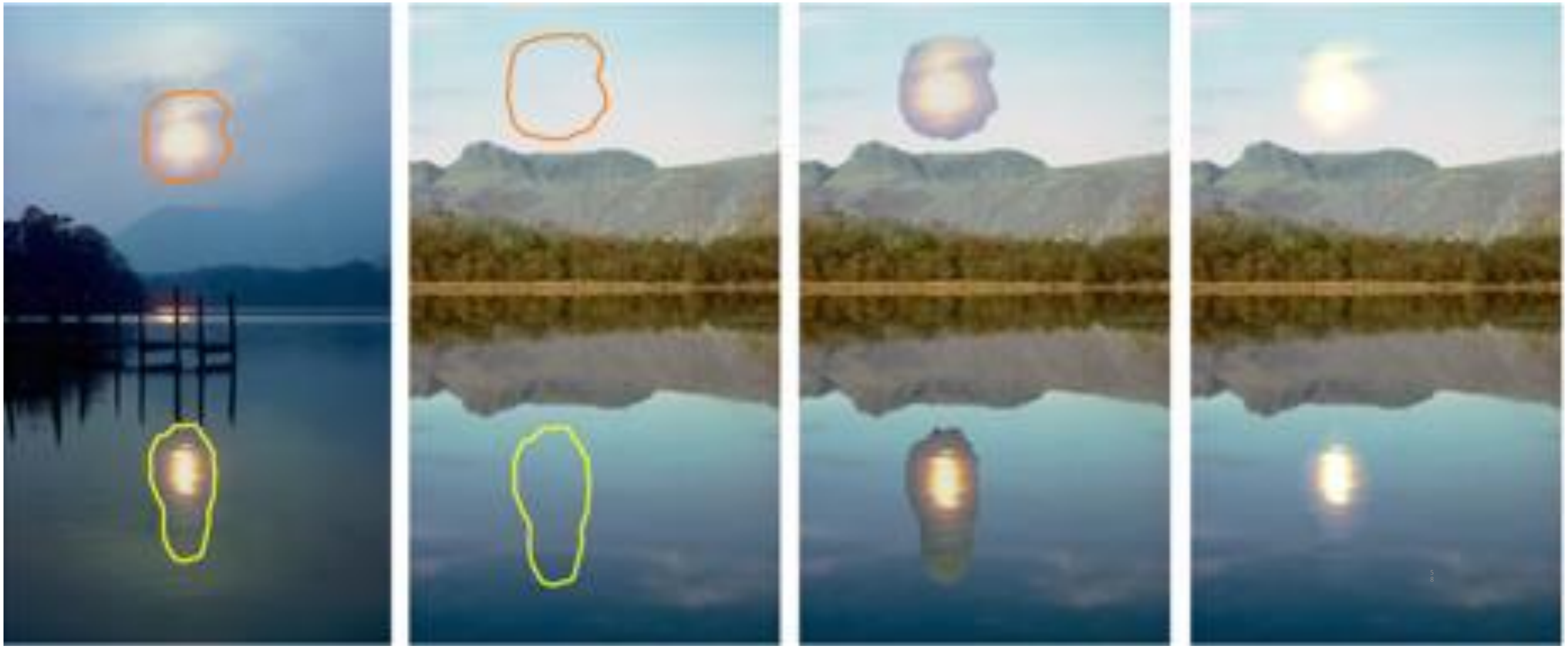
Poisson blending

Someone leaked season 8 of Game of Thrones

or, more likely, they made some creative use of Poisson blending

Key idea

When blending, retain the gradient information as best as possible



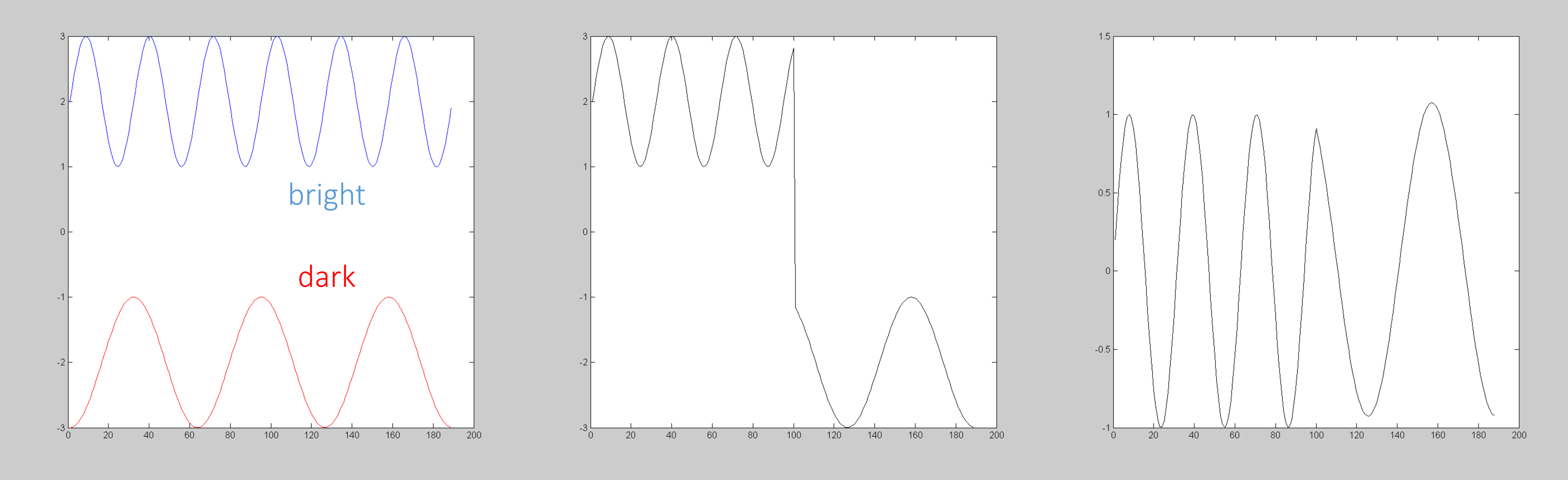
source destination copy-paste Poisson blending

Example

How come the colors get smoothed out after blending?

originals copy-paste Poisson blending

Poisson blending: 1D example



two signals

regular blending

blending derivatives

Warning: math ahead

Also note: you'll implement this for homework 3.

Definitions and notation

Notation

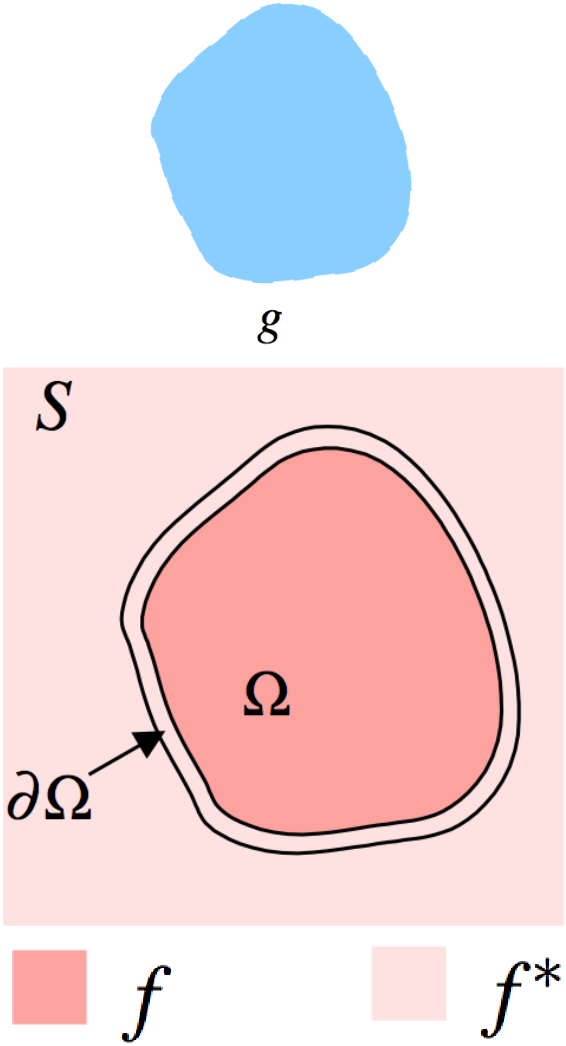
g: source function

S: destination

Ω: destination domain

f: interpolant function

f*: destination function



Which one is the unknown?

Definitions and notation

Notation

g: source function

S: destination

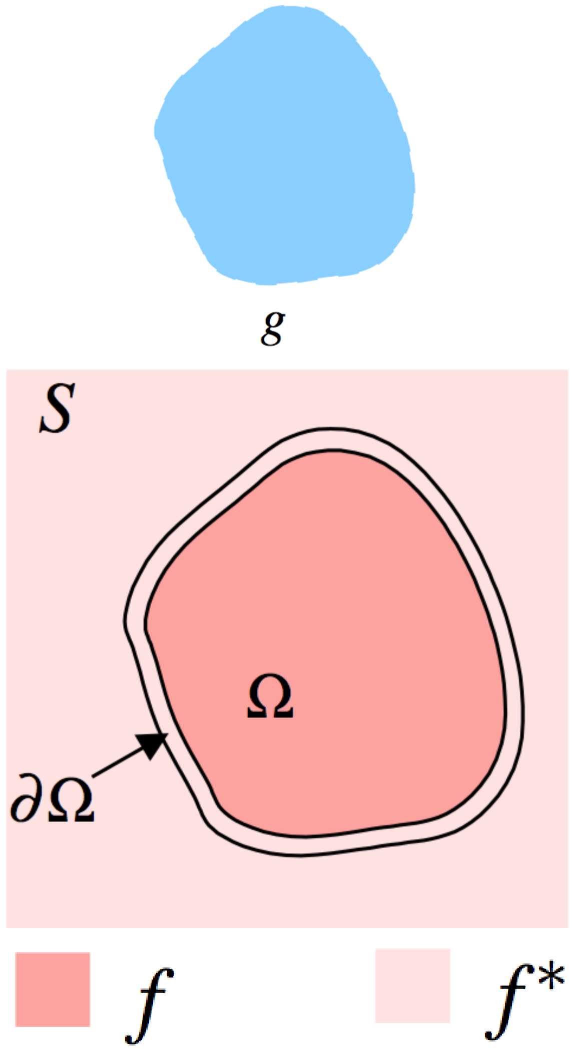
Ω: destination domain

f: interpolant function

f*: destination function

How should we determine f?

- should it look like g?
- should it look like f*?



Interpolation criterion

"Variational" means optimization where the unknown is an entire function

Variational problem

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

what does this term do?

what does this term do?

Recall ...

Image gradient

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

is this known?

$$\mathbf{v} = (u, v) = \nabla g$$

Interpolation criterion

"Variational" means optimization where the unknown is an entire function

Variational problem

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

gradient of f looks like gradient of g

f is equivalent to f* at the boundaries

Recall ...

Image gradient

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Yes, since the source function g is known

$$\mathbf{v} = (u, v) = \nabla g$$

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

$$\Delta f = \text{div } \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

what does this term do?

Gradient $\mathbf{v} = (u, v) = \nabla g$

Laplacian $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Divergence $\text{div } \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$

$$\begin{aligned}\text{div } \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ &= \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \\ &= \Delta g\end{aligned}$$

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

$$\Delta f = \text{div } \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

Laplacian of f same as g

Gradient $\mathbf{v} = (u, v) = \nabla g$

Laplacian $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Divergence $\text{div } \mathbf{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$

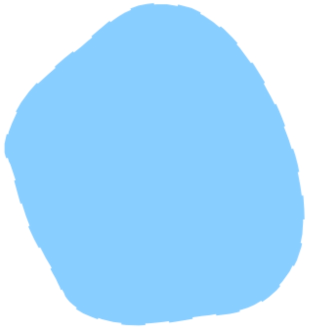
$$\begin{aligned} \text{div } \mathbf{v} &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ &= \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \\ &= \Delta g \end{aligned}$$

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

$$\Delta f = \text{div}\ \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

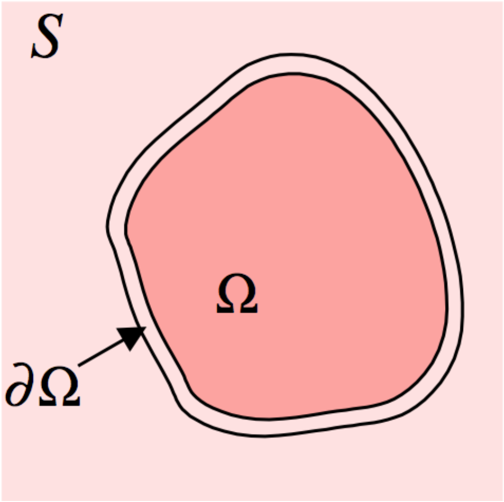


g

so make these guys ...

Δg Δf

the same



How can we do this?

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

$$\Delta f = \text{div } \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

So for each pixel p, do:

$$\Delta f_p = \Delta g_p$$

Or for discrete images:

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

How did we go from one to the other?

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

$$\Delta f = \operatorname{div} \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

So for each pixel p, do: $$\Delta f_p = \Delta g_p$$

Or for discrete images: $$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

Recall...

Laplace filter

0	1	0
1	-4	1
0	1	0

What’s known and what’s unknown?

Equivalently

This is where *Poisson* blending comes from

Poisson equation (with Dirichlet boundary conditions)

$$\Delta f = \text{div}\ \mathbf{v} \quad \text{over} \quad \Omega, \quad \text{with} \quad f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

So for each pixel p, do:

$$\Delta f_p = \Delta g_p$$

Or for discrete images:

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

f is unknown except at the boundary

g and its Laplacian are known

Recall...

Laplace filter

0	1	0
1	-4	1
0	1	0

We can rewrite this as

linear equation of N variables

$$4f_p - \sum_{q \in N_p} f_q = 4g_p - \sum_{q \in N_p} g_q$$

one for each pixel in destination

In vector form:

$$[0 \cdots 1 \cdots 1\ 4\ 1 \cdots 1 \cdots 0] \begin{bmatrix} f_1 \\ \vdots \\ f_{q_1} \\ \vdots \\ f_{q_2} \\ f_p \\ f_{q_3} \\ \vdots \\ f_{q_4} \\ \vdots \\ f_N \end{bmatrix} = \begin{bmatrix} \Delta g_1 \\ \vdots \\ \Delta g_{q_1} \\ \vdots \\ \Delta g_{q_2} \\ \Delta g_p \\ \Delta g_{q_3} \\ \vdots \\ \Delta g_{q_4} \\ \vdots \\ \Delta g_N \end{bmatrix}$$

What is this? ↗

(each pixel adds another 'sparse' row here)

⇒ Linear system of equations

$$Af = b$$

How would you solve this?

WARNING: requires special treatment at the borders (target boundary values are same as source)

Solving the linear system

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{f} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{f}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{f} - 2\boldsymbol{f}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{f} = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $\boldsymbol{f} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$

Solving the linear system

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{f} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{f}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{f} - 2\boldsymbol{f}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

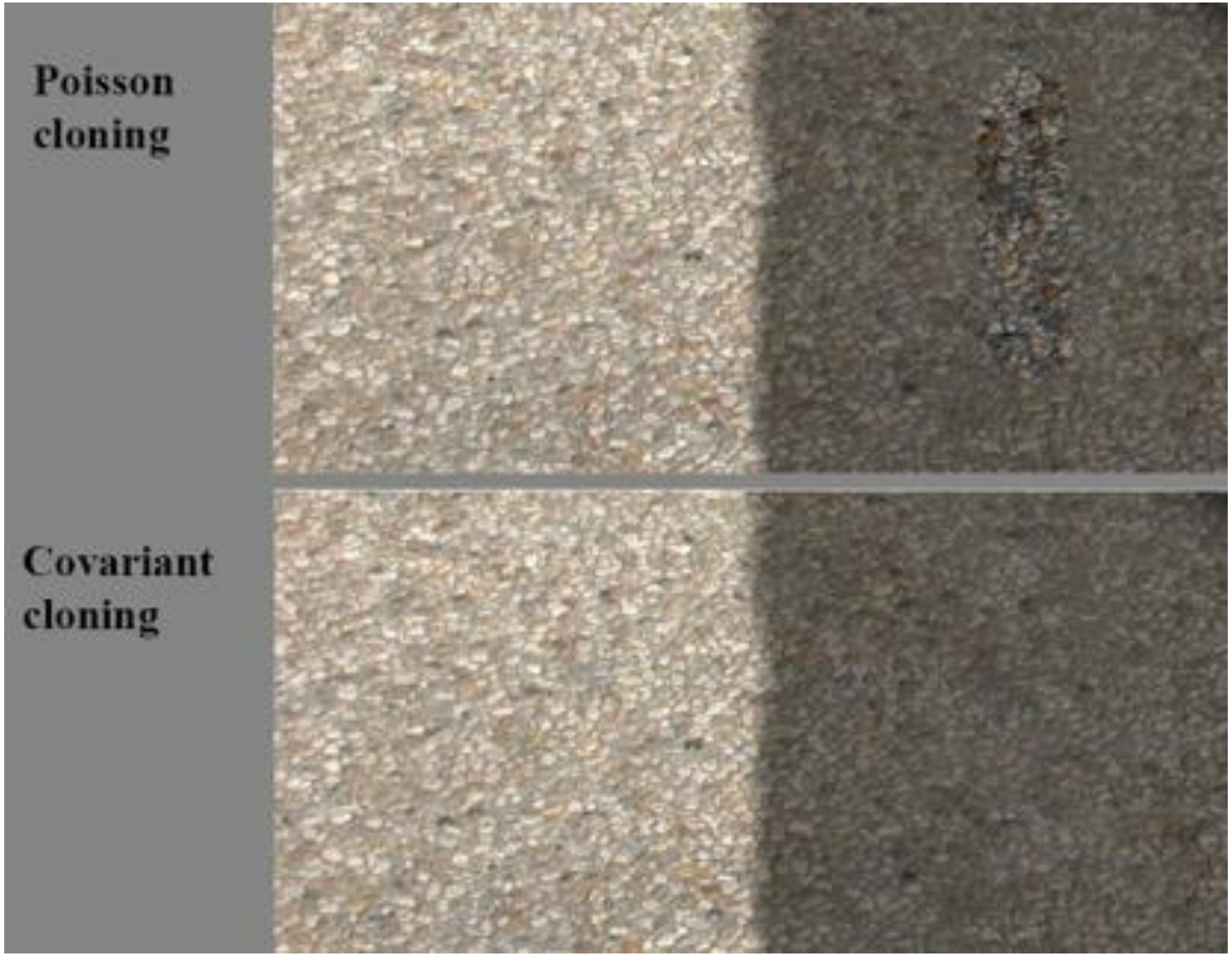
Set derivative to 0 $$(\mathbf{A}^\top\mathbf{A})\boldsymbol{f} = \mathbf{A}^\top\boldsymbol{b}$$

Solve for x $$\boldsymbol{f} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$$ ⟵ Note: You almost never want to compute the inverse of a matrix.

In Matlab:

```
f = A \ b
```

Photoshop's "healing brush"



Slightly more advanced version of what we covered here:

- Uses higher-order derivatives

Contrast problem

Loss of contrast when pasting from dark to bright:

- Contrast is a multiplicative property.
- With Poisson blending we are matching linear differences.

Contrast problem

Loss of contrast when pasting from dark to bright:

- Contrast is a multiplicative property.
- With Poisson blending we are matching linear differences.

Solution: Do blending in log-domain.

More blending

originals

copy-paste

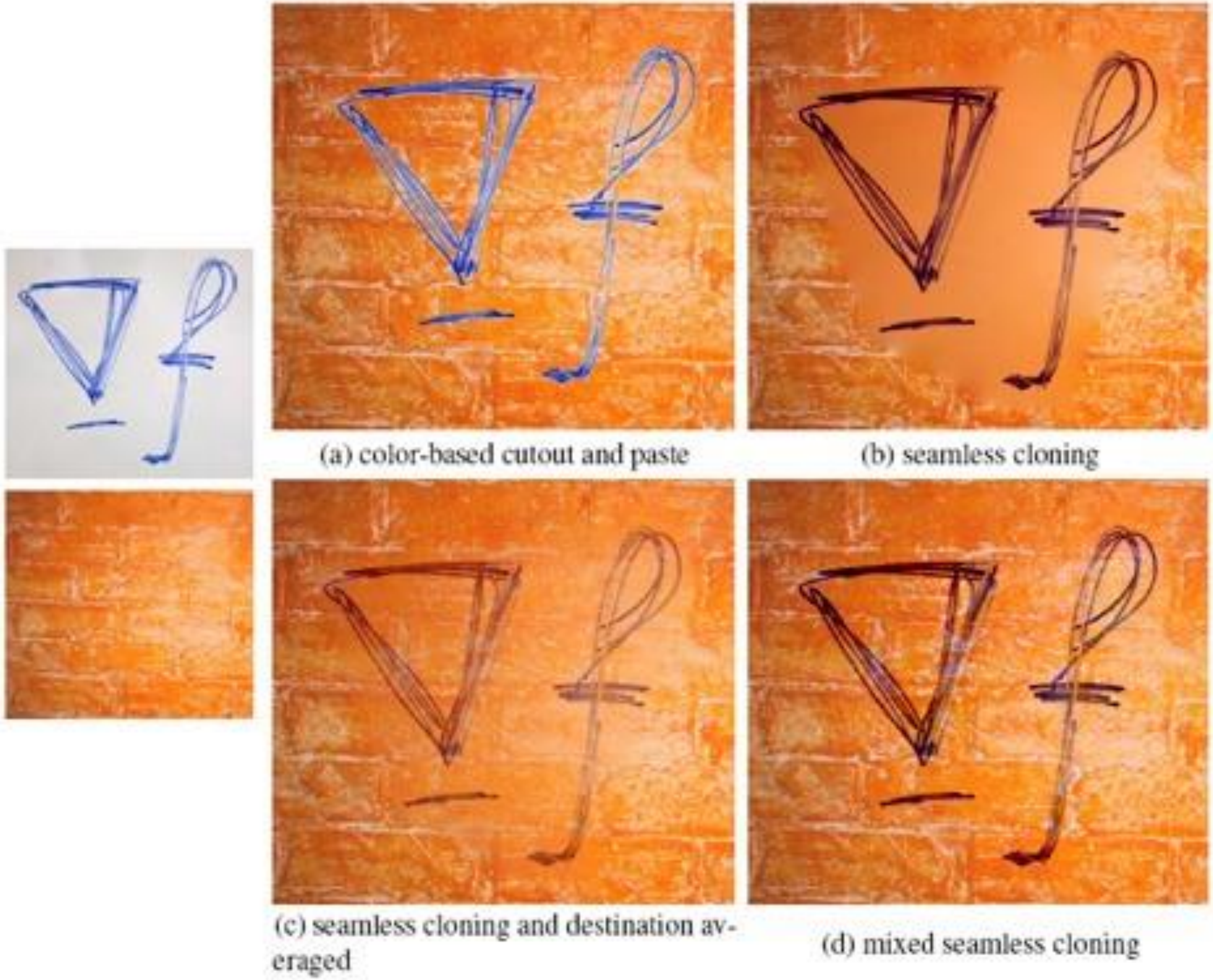
Poisson blending

Blending transparent objects

source

destination

Blending objects with holes



Editing

Concealment

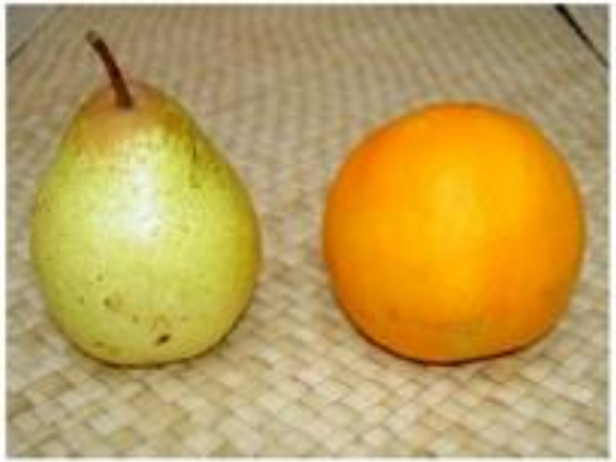
How would you do this with Poisson blending?

Concealment

How would you do this with Poisson blending?

- Insert a copy of the background.

Texture swapping



References

Basic reading:
- Szeliski textbook, Sections 3.13, 3.5.5, 9.3.4, 10.4.3.

Additional reading:
- Pérez et al., "Poisson Image Editing," SIGGRAPH 2003.
  the original Poisson blending paper.
- Georgiev, "Covariant Derivatives and Vision," ECCV 2006.
  a paper from Adobe describing the version of Poisson blending implemented in Photoshop's "healing brush".
- Elder and Goldberg, "Image editing in the contour domain", PAMI 2001.
- Bhat et al., "GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering," ToG 2010.
- Agrawal and Raskar, "Gradient Domain Manipulation Techniques in Vision and Graphics," ICCV 2007 course, http://www.amitkagrawal.com/ICCV2007Course/
  the above references provide an overview of gradient-domain processing as a general image processing paradigm, which can be used for a broad set of applications beyond blending, including tone-mapping, colorization, converting to grayscale, edge enhancement, image abstraction and non-photorealistic rendering.