Dartmouth

Domain Transforms, Warping & Morphing

CS 89.15/189.5, Fall 2015

Wojciech Jarosz

wojciech.k.jarosz@dartmouth.edu

Most slides stolen from Frédo Durand

Last time

HDR and tone mapping

- Questions?

Filtering + convolution assignment was due last night

- Questions?

HDR + tone mapping assignment out now, due next Wed

- includes solutions to filtering assignment
- compare yours to the solution

Domain, range

Domain vs. range

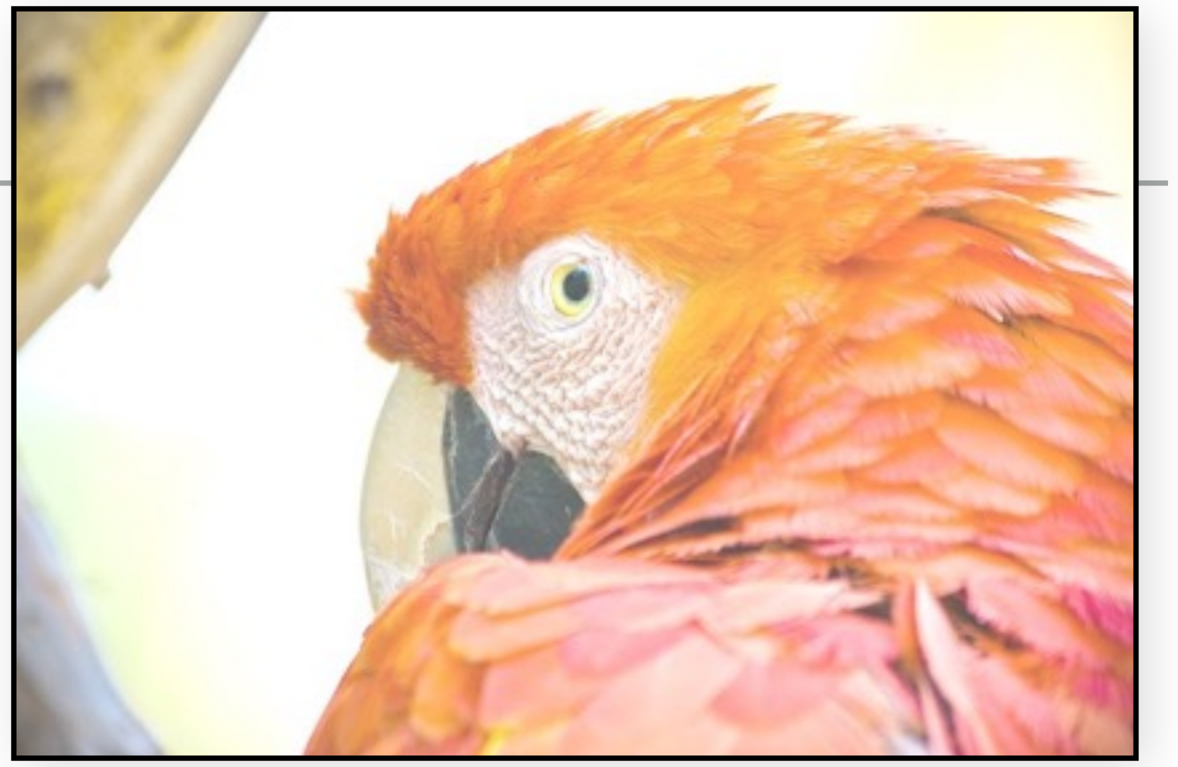
2D plane: domain of images

color value: range (R^3 for us)

- red, green and blue components stored in im(x, y, 0), im(x, y, 1), im(x, y, 2), respectively

Basic types of operations

output(x,y)



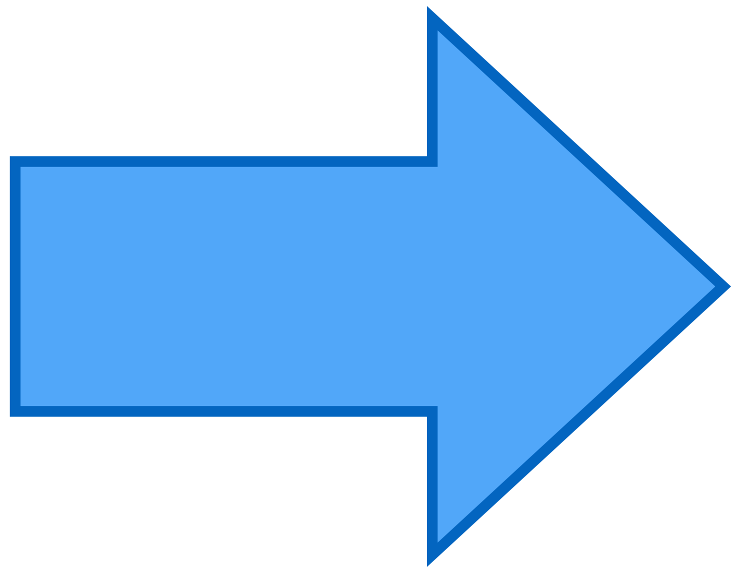
output(x,y) = f(image(x,y))

Point operations:

range only

Assignment 2

image(x,y)



Basic types of operations

output(x,y)

output(x,y) = f(image(x,y))

Point operations:
range only
Assignment 2

image(x,y)

Neighborhood operations:
domain and range
Assignments 3, 4, 5

Basic types of operations

image(x,y)

output(x,y)

$$\text{output}(x,y) = f(\text{image}(x,y))$$

Point operations:
range only
Assignment 2

$$\text{output}(x,y) = \text{image}(f(x,y))$$

Domain
operations
Assignment 6

Neighborhood operations:
domain and range
Assignments 3, 4, 5

Domain operations

Domain transform

Apply a function *f* from R^2 to R^2 to the image domain

if im(x, y) had color c in the input, then im(*f*(x, y)) should have color c in the output

Transformation

Simple parametric transformations

- linear, affine, perspective, etc

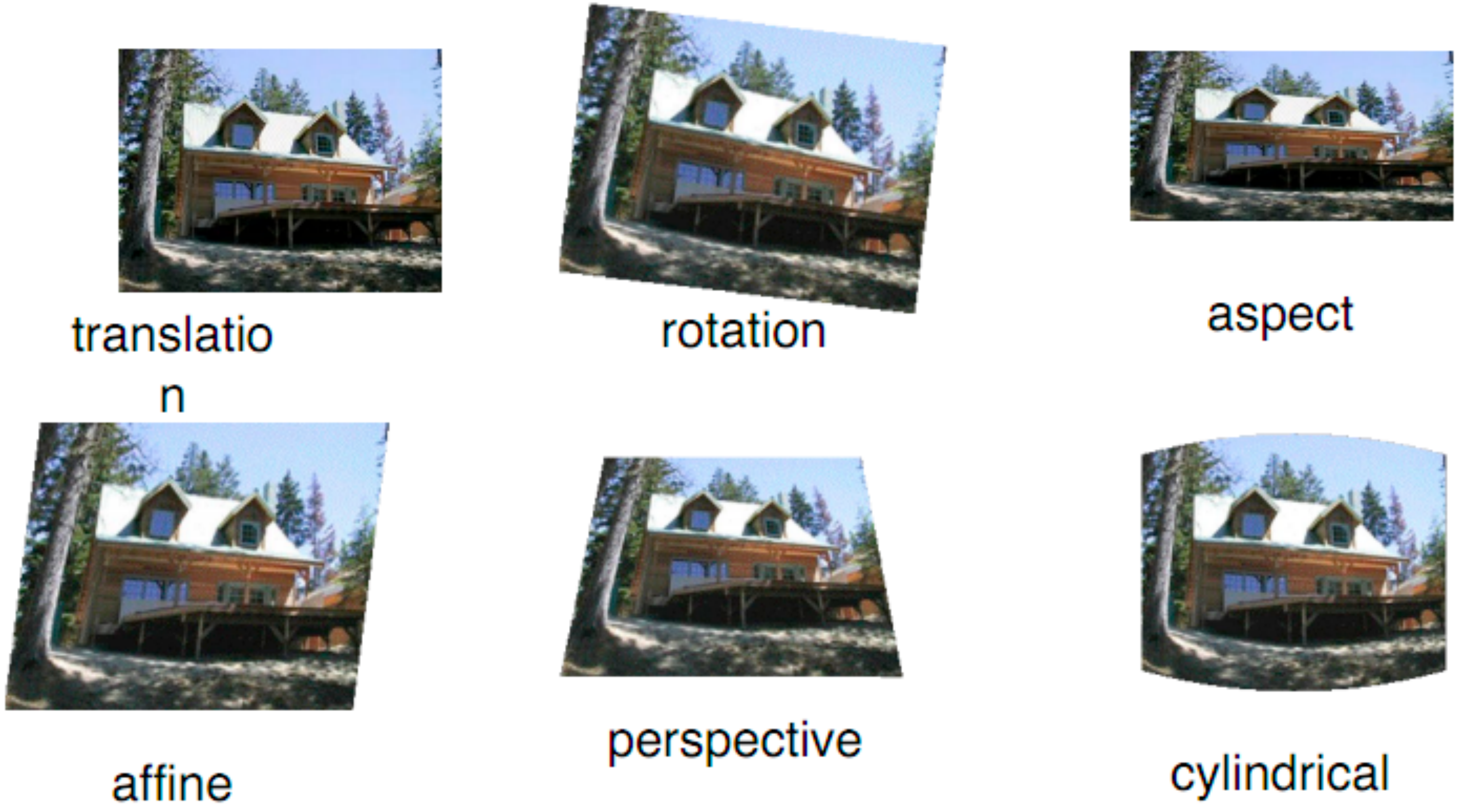


illustration by Rick Szeliski

Warping

Imagine your image is made of rubber; warp the rubber

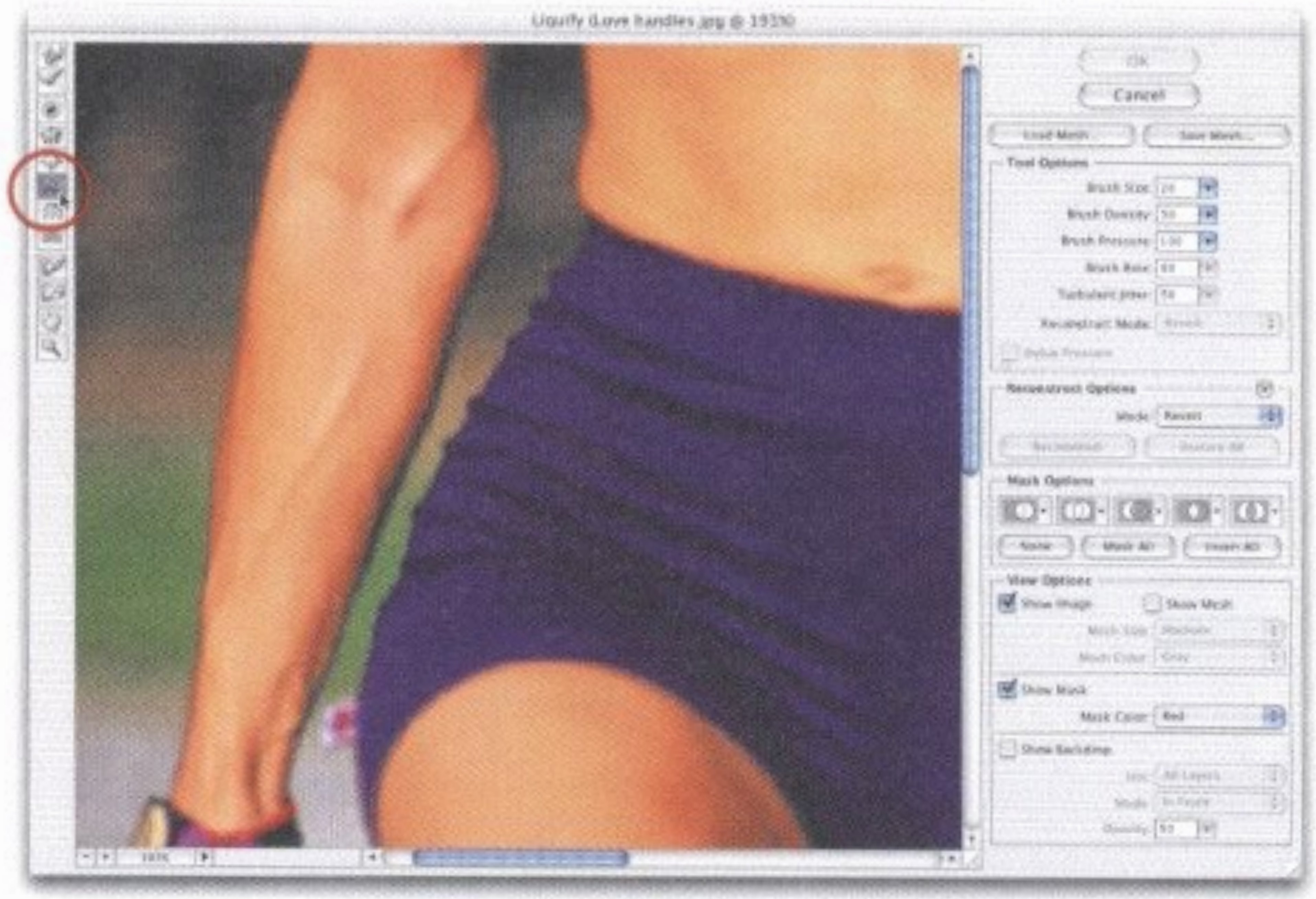
No prairie dogs were armed when creating this image

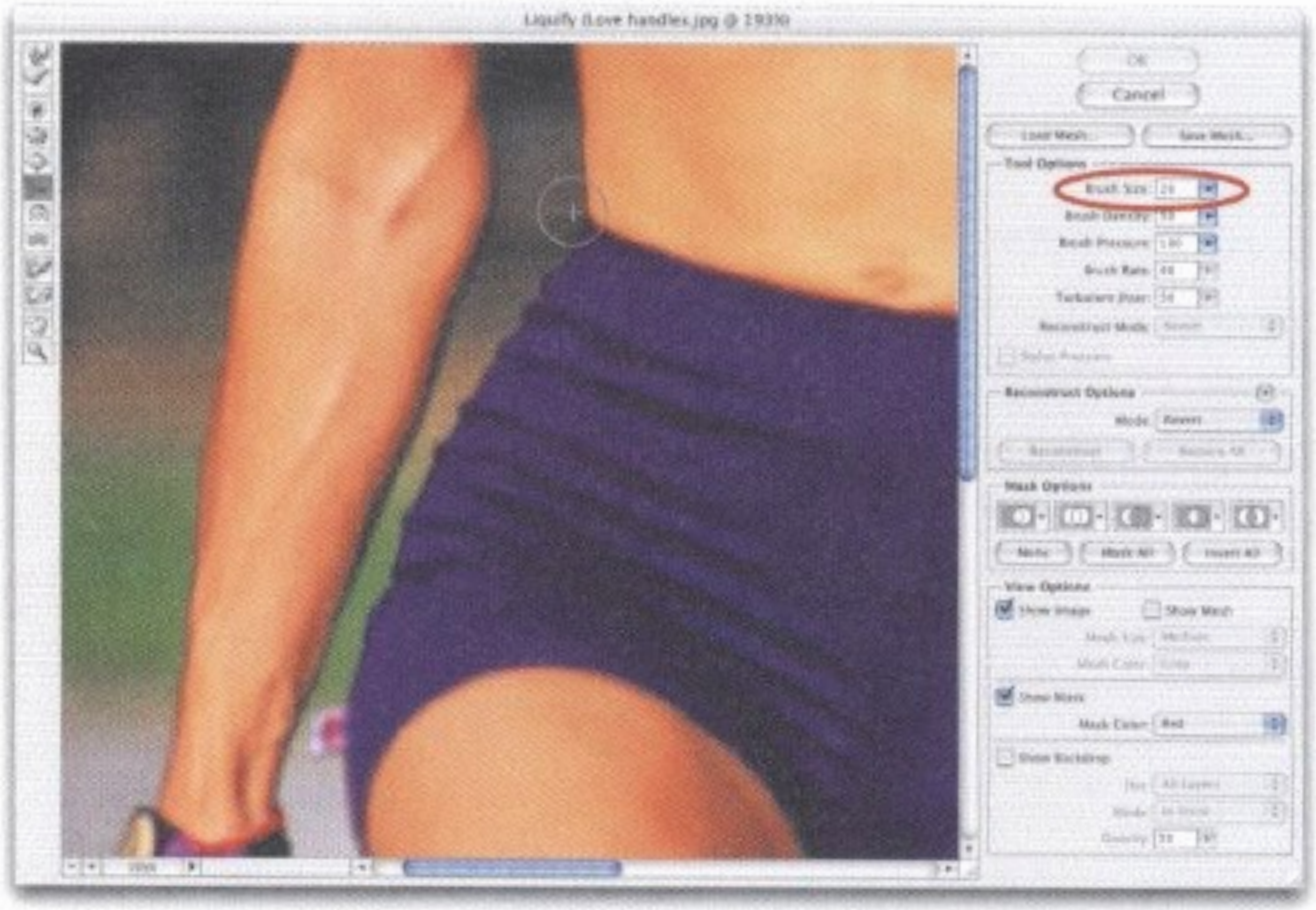
Application of warping: weight loss

Liquify in photoshop

figure 9.39

The Liquify filter's Warp tool pushes pixels forward as you drag.





Step Three:

Get the Push Left tool from the Toolbar (as shown here). It was called the Shift Pixels tool in Photoshop 6 and 7, but Adobe realized that you were getting used to the name, so they changed it, just to keep you off balance.

Step Four:

Choose a relatively small brush size (like the one shown here) using the Brush Size field near the top-right of the Liquify dialog. With it, paint a downward stroke starting just above and outside the love handle and continuing downward. The pixels shifts back in toward the body, removing the love handle as you paint. (Note: If you need to remove love handles on the left side of the body, paint upward rather than downward. Why? That's just the way it works.) When you click OK, the love handle repair is complete.

Domain transform issues

Apply a function f from R^2 to R^2 to the image domain

looks easy enough

But 2.5 big issues:

- which direction do we transform
- how do we deal with non-integer coordinates?

- And for warping: how do we specify f?

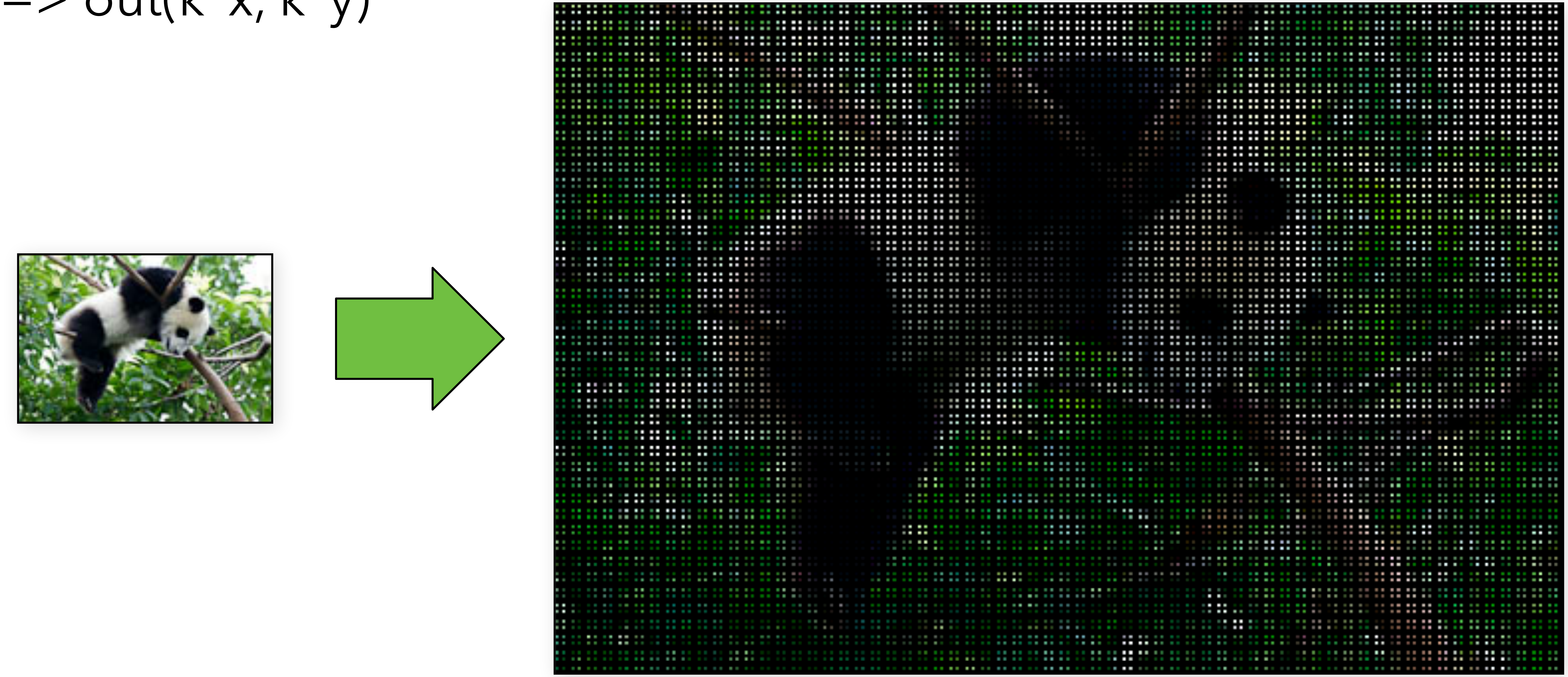
Questions?

Basic resampling

Naive scaling

Loop over input pixels and transform them to their output location

- im(x, y) => out(k*x, k*y)



Use the inverse transform!!!!!

Main loop on **output** pixels

- out(x, y) <= im(x/k, y/k)

Take-home message

Main loop over OUTPUT pixels

use INVERSE transform

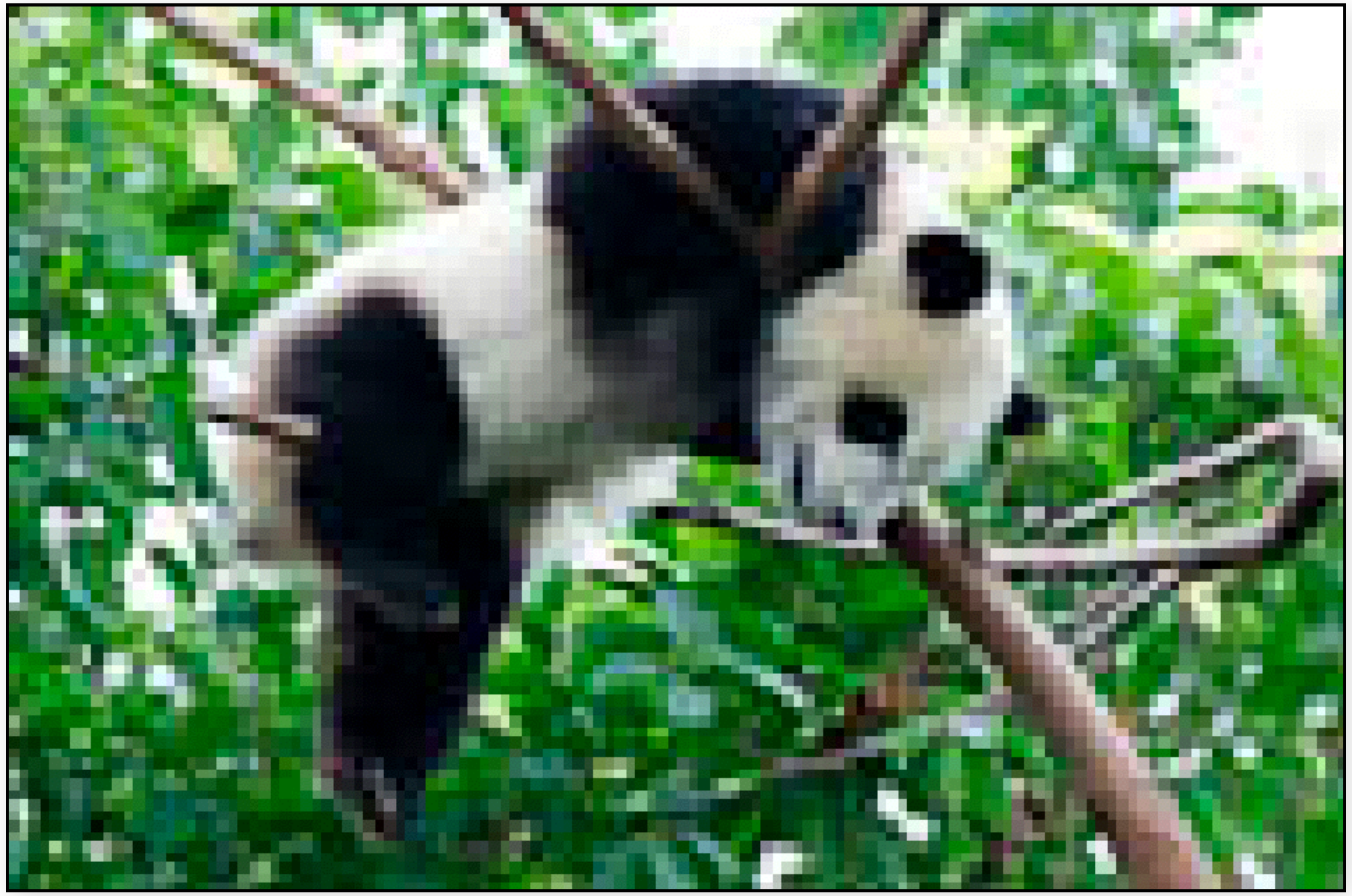
Questions?

Remaining problem

A little too "blocky"

Because we round to the nearest integer pixel coord.

- called nearest neighbor reconstruction



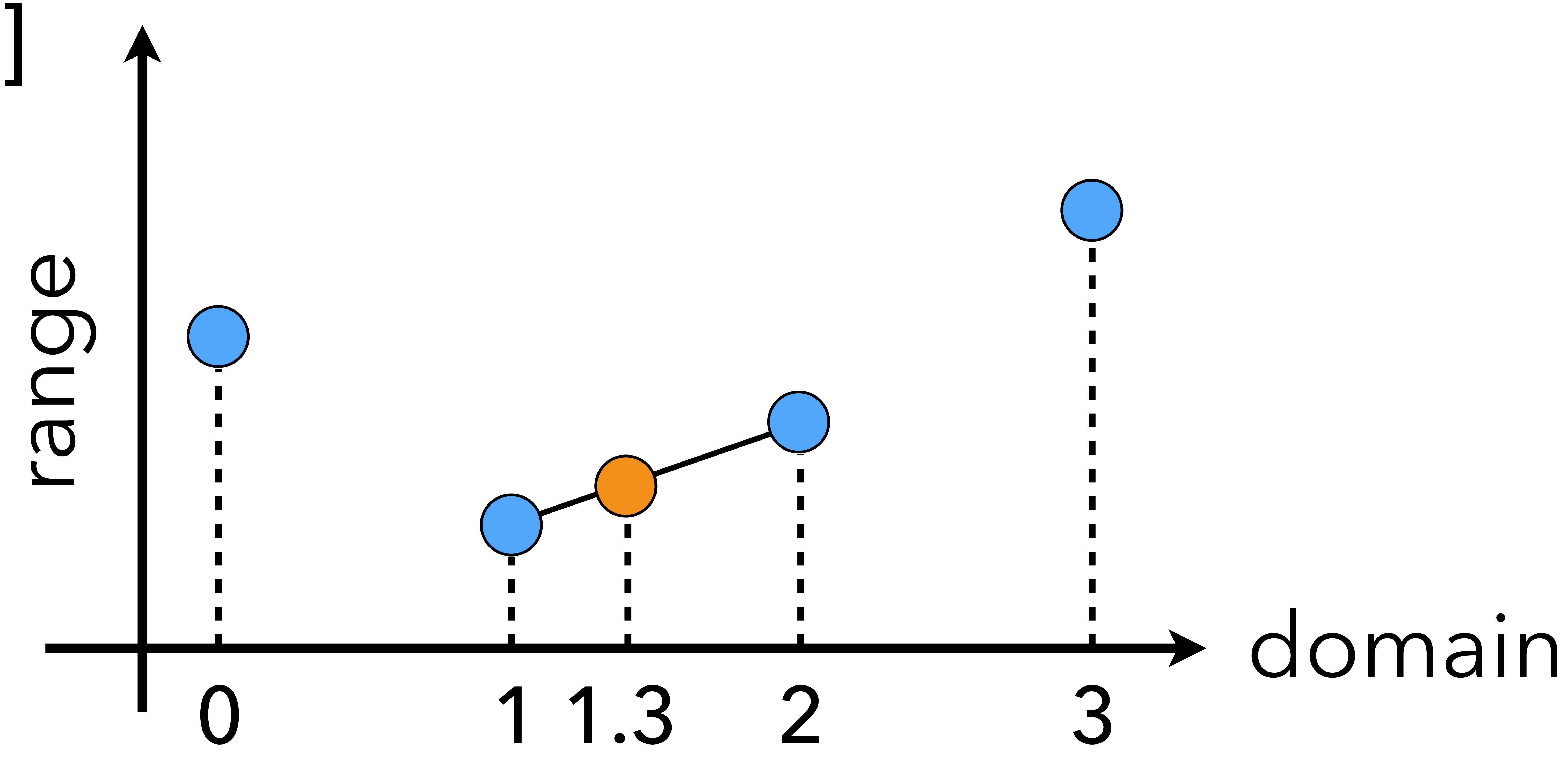
Linear reconstruction

Consider a 1D image/array (im) along x

reconstruct im[1.3]

=0.7*im[1]+0.3*im[2]

lerp function

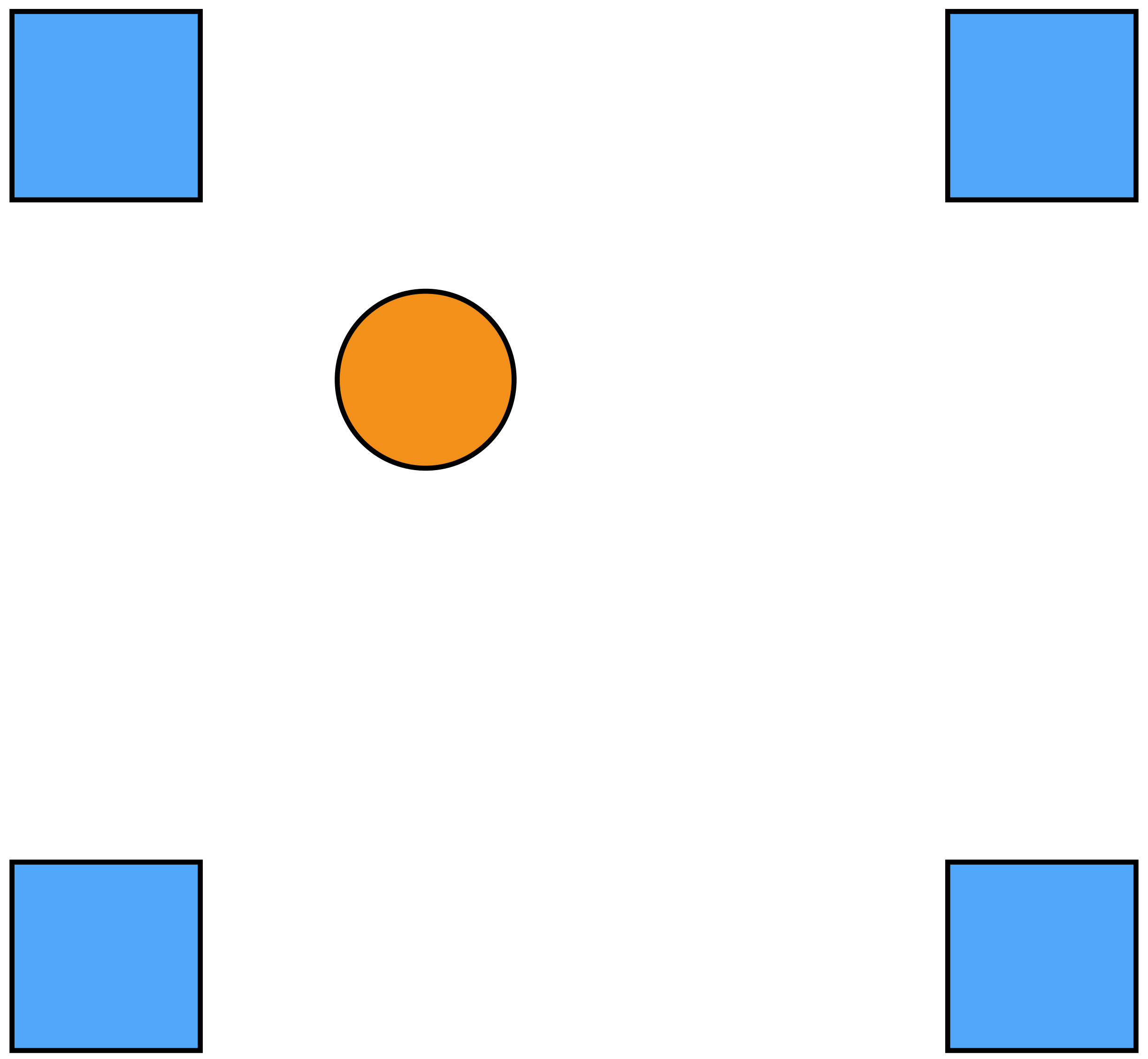


Bilinear reconstruction

Take 4 nearest neighbors

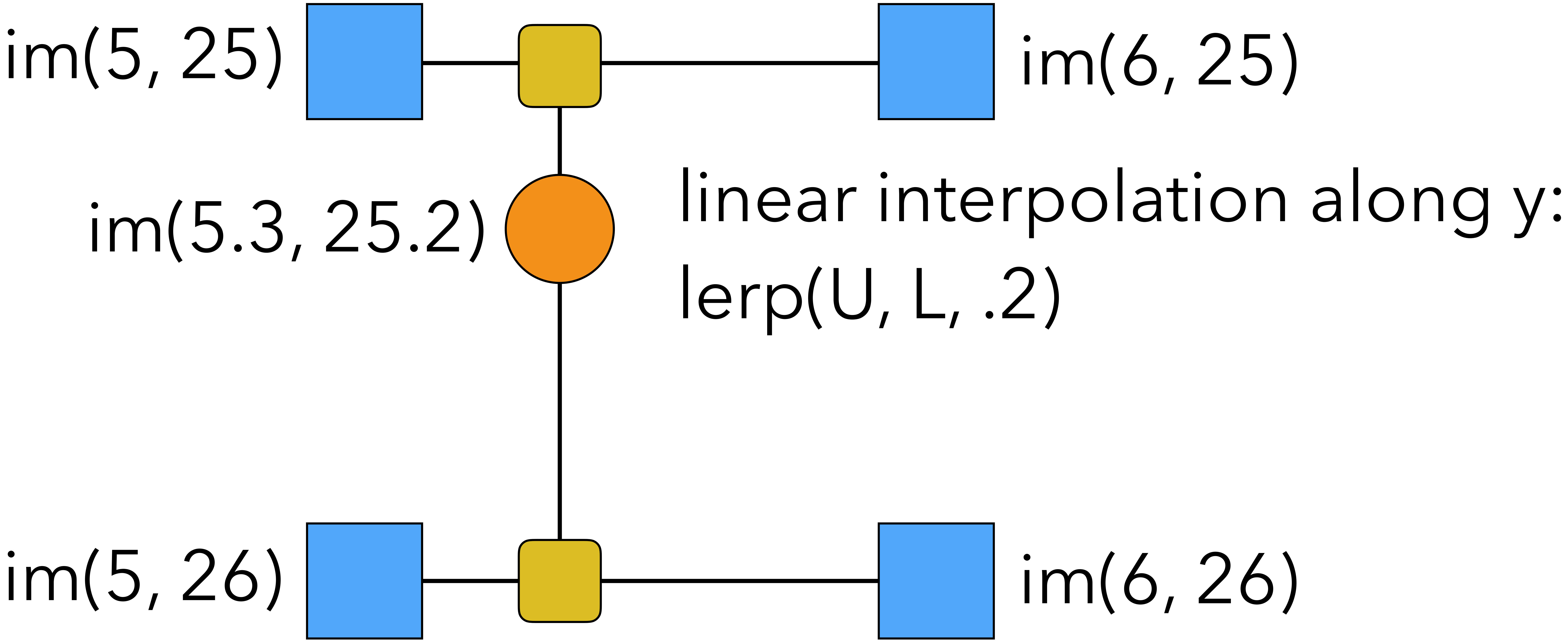
Weight according to x & y fractional coordinates

Can be done using two 1D linear reconstructions along x then y (or y then x)



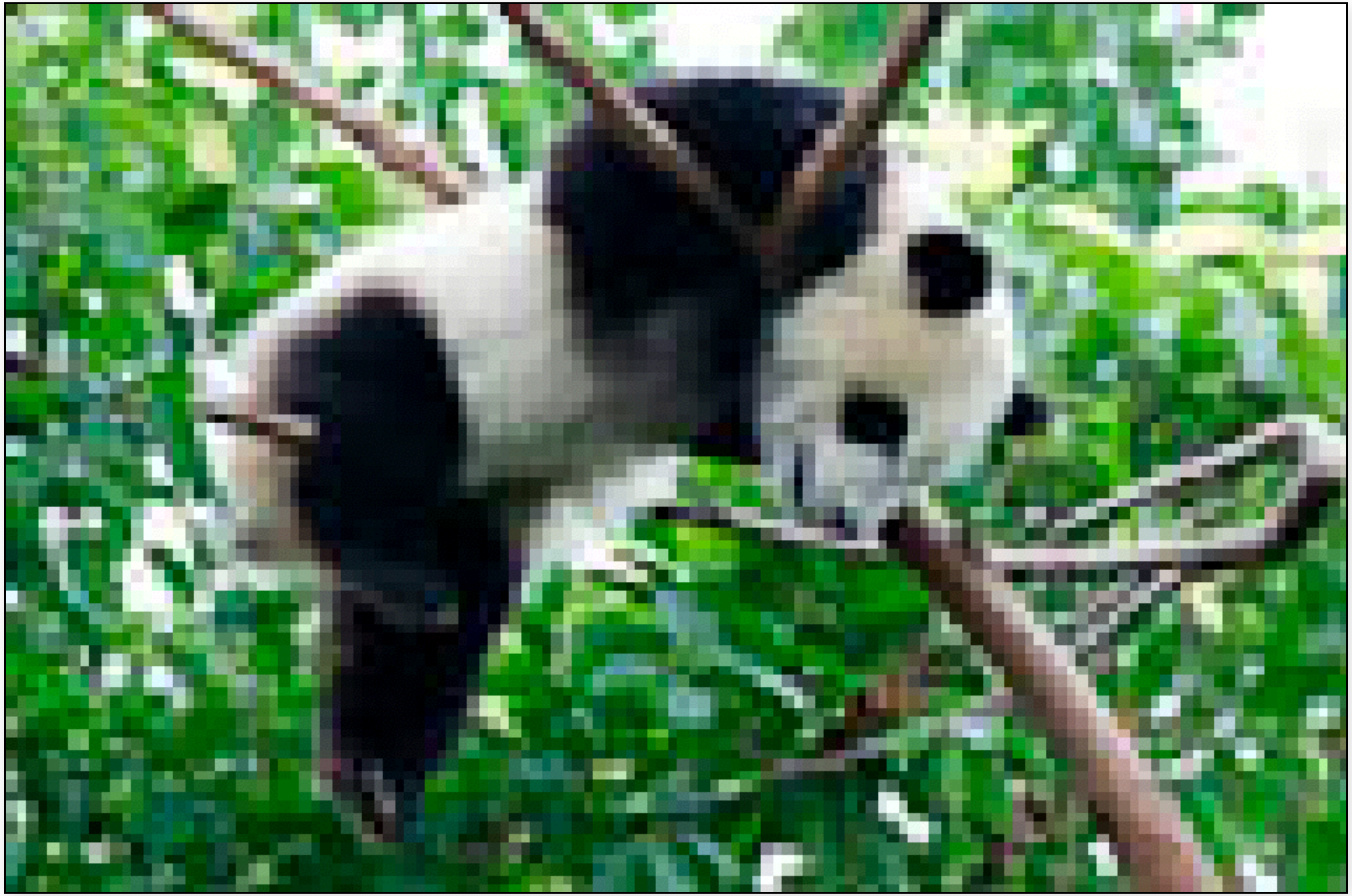
Bilinear

linear interpolation along x: U = lerp(im(5,25), im(6,25), .3)



linear interpolation along x: L = lerp(im(5,26), im(6,26), .3)

Recall nearest neighbor



Bilinear

Take home messages

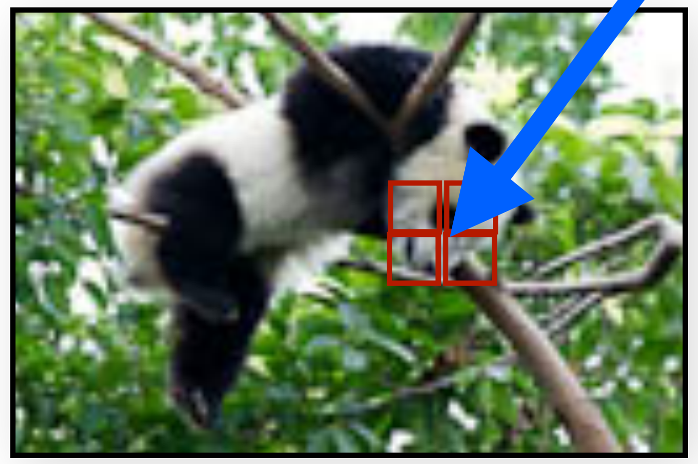
Main loop over OUTPUT pixels

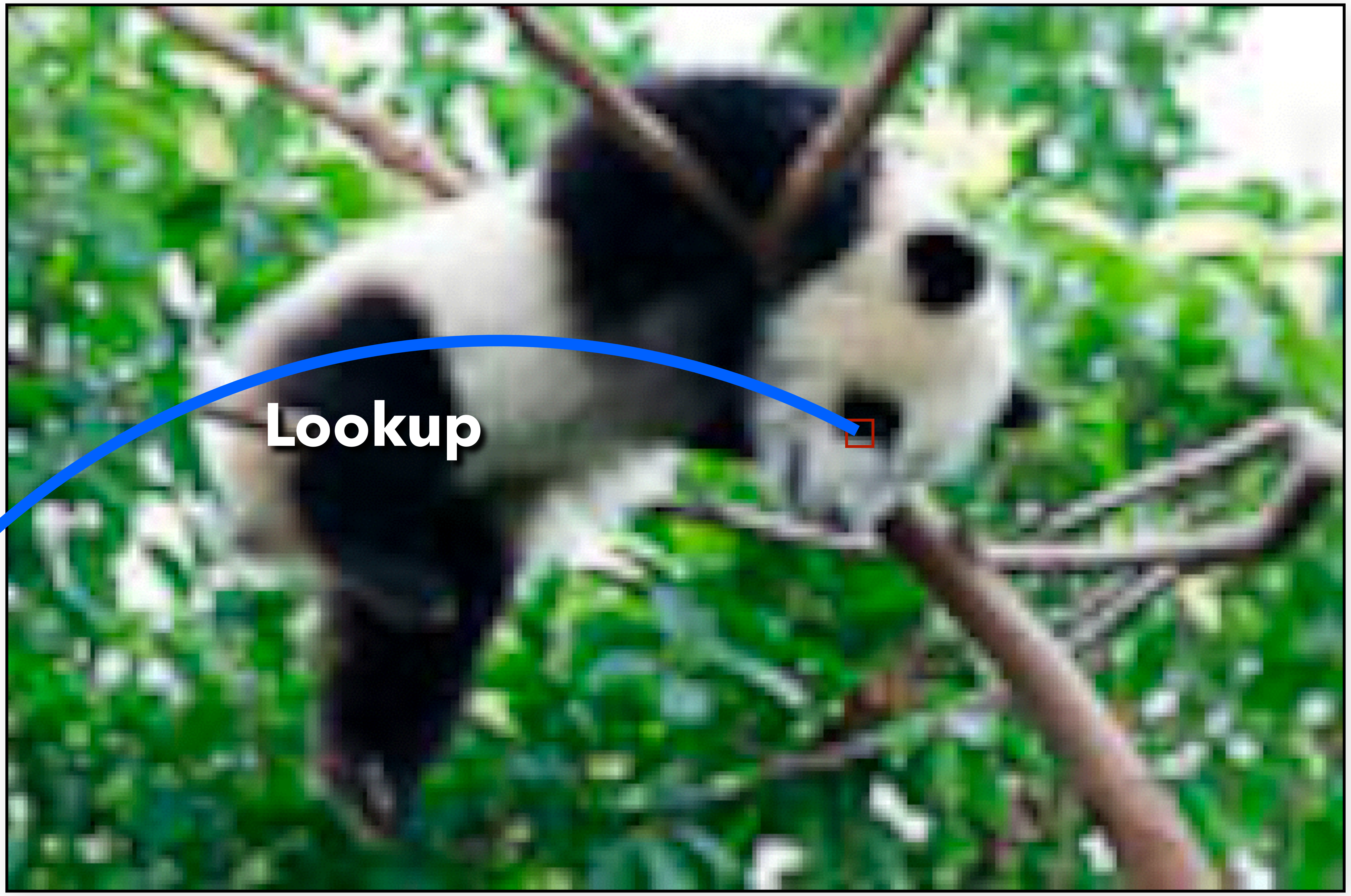
- Makes sure you cover all of them

Use INVERSE transform

Reconstruction makes a difference

- Linear much better than nearest neighbor





Questions?

Better reconstruction

Consider more than 4 pixels:

- bicubic, Lanczos, etc.

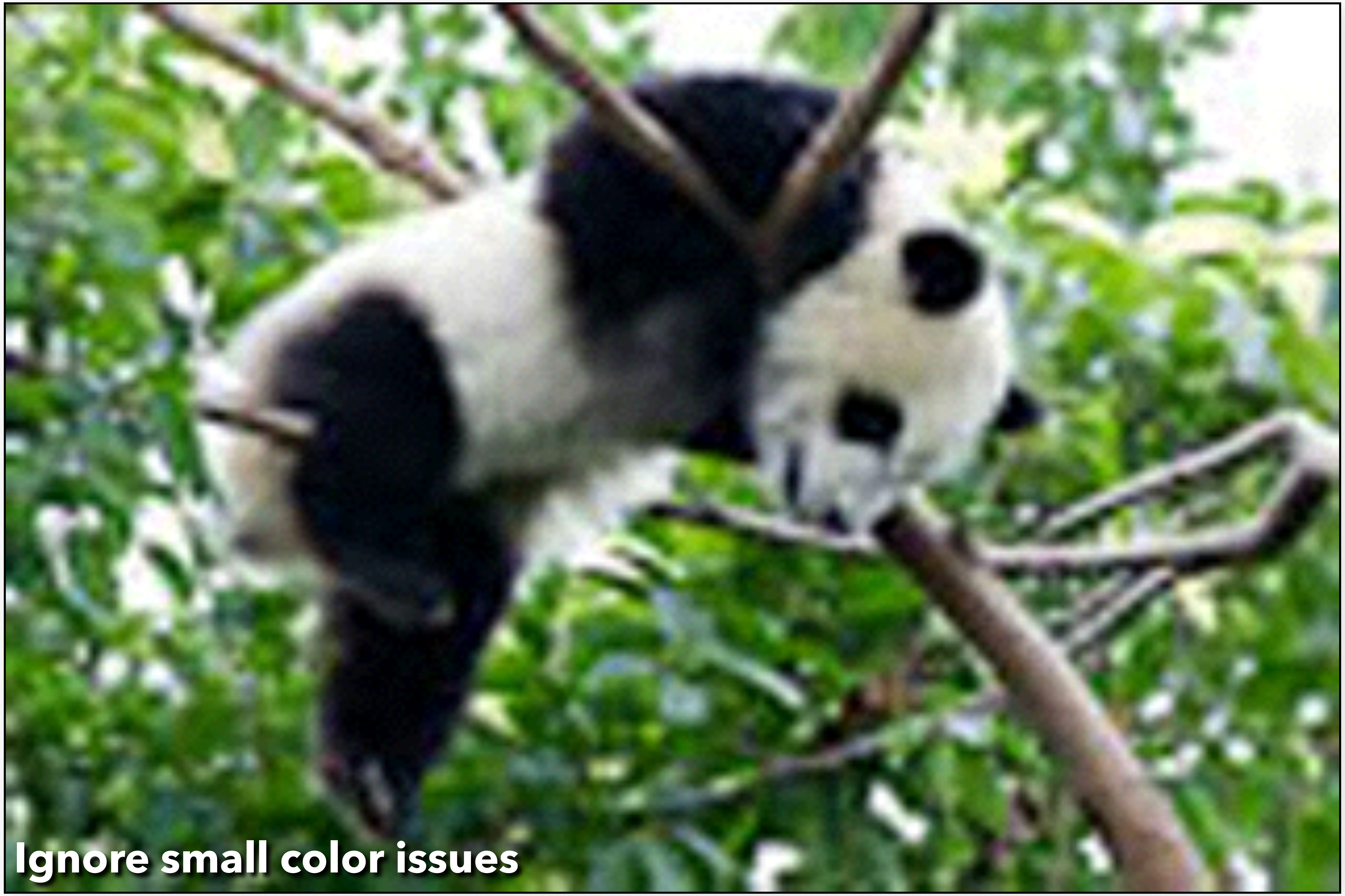
Try to sharpen/preserve edges

Use training database of low-res/high-res pairs

- http://people.csail.mit.edu/billf/superres/index.html

Bilinear

Bicubic (Photoshop)



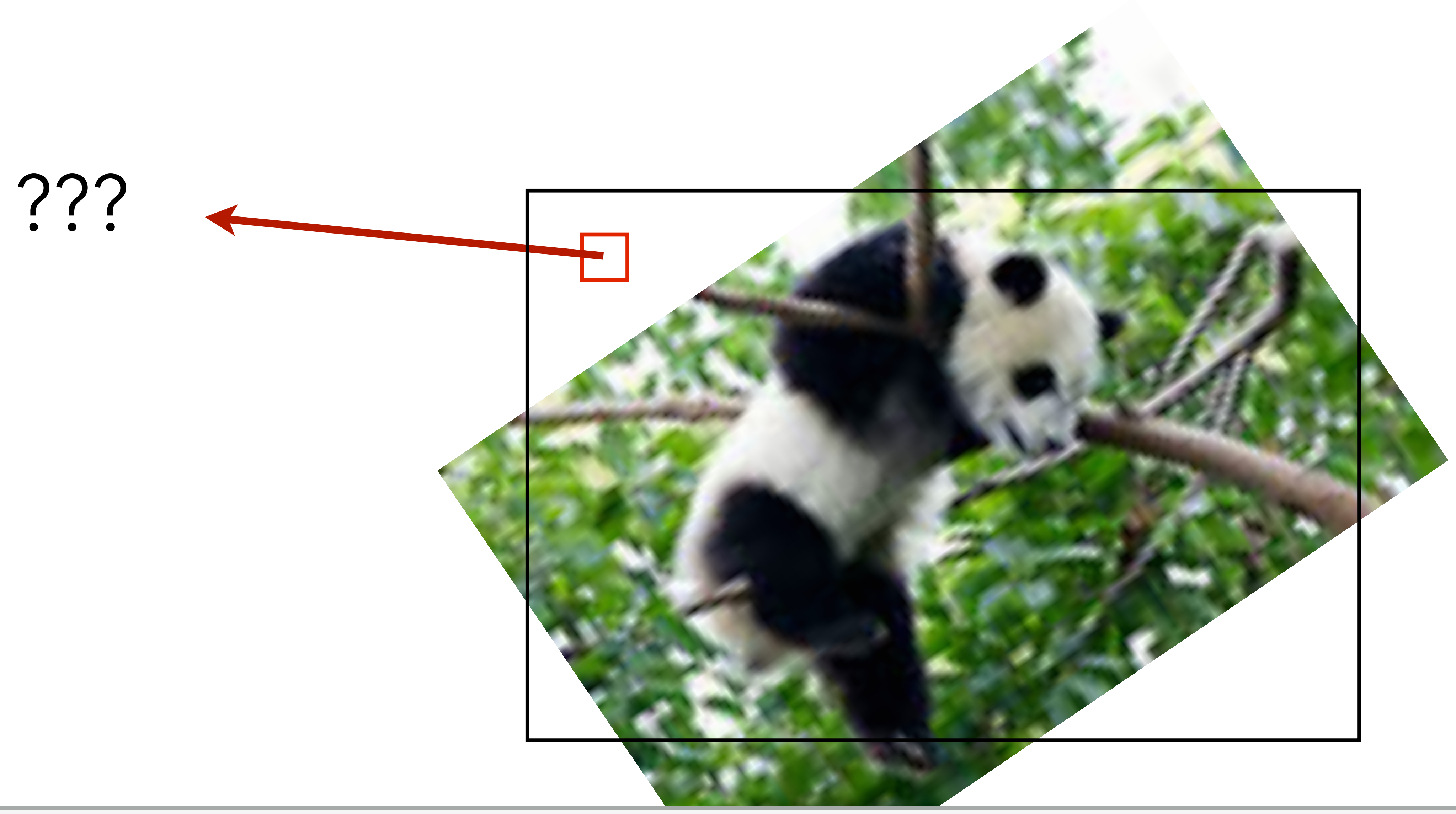
Questions?

Padding

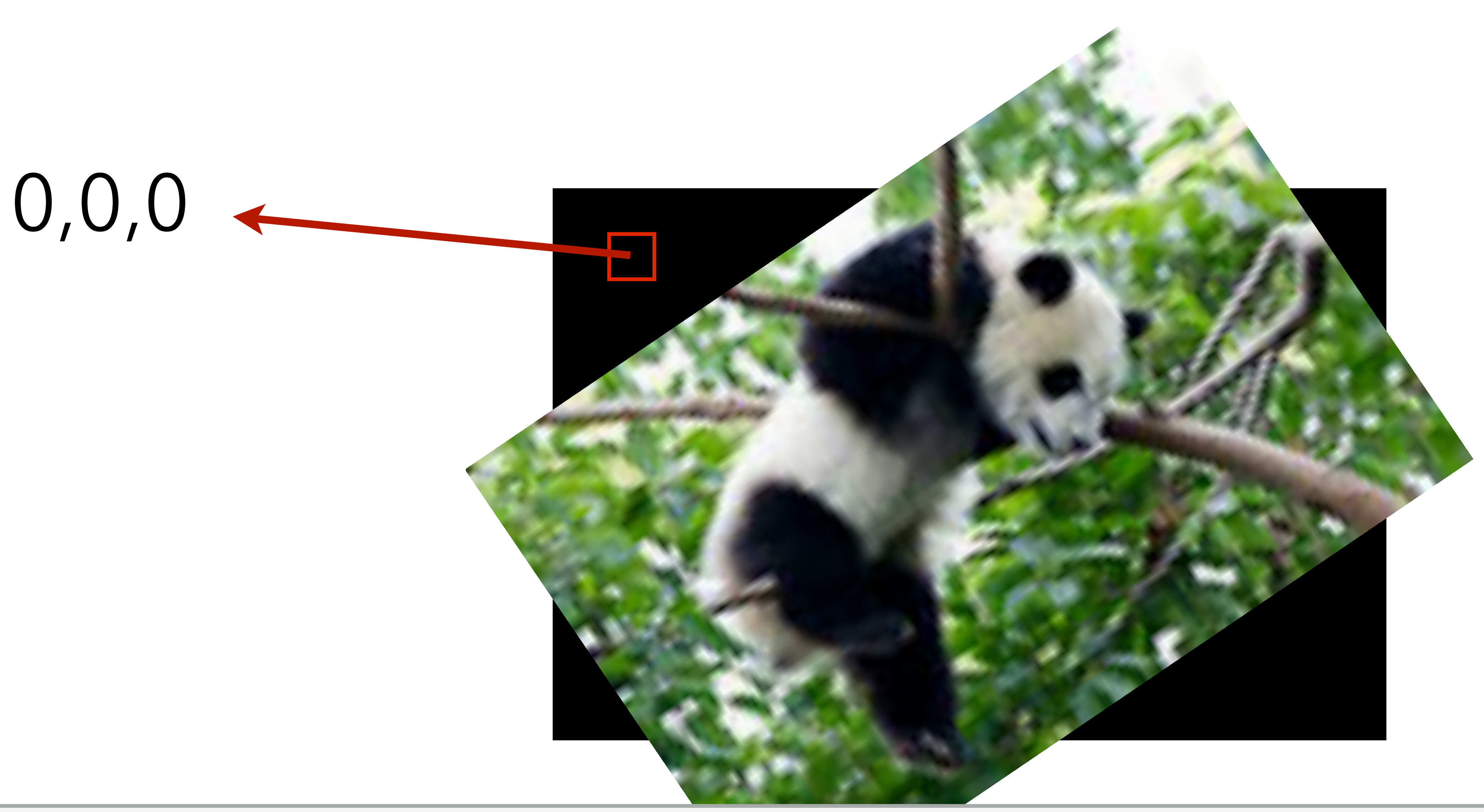
Padding problems

Sometimes, we try to read outside the image

- e.g. x, y are negative
- For example, we try to rotate an image



Black Padding



Edge Padding

Questions?

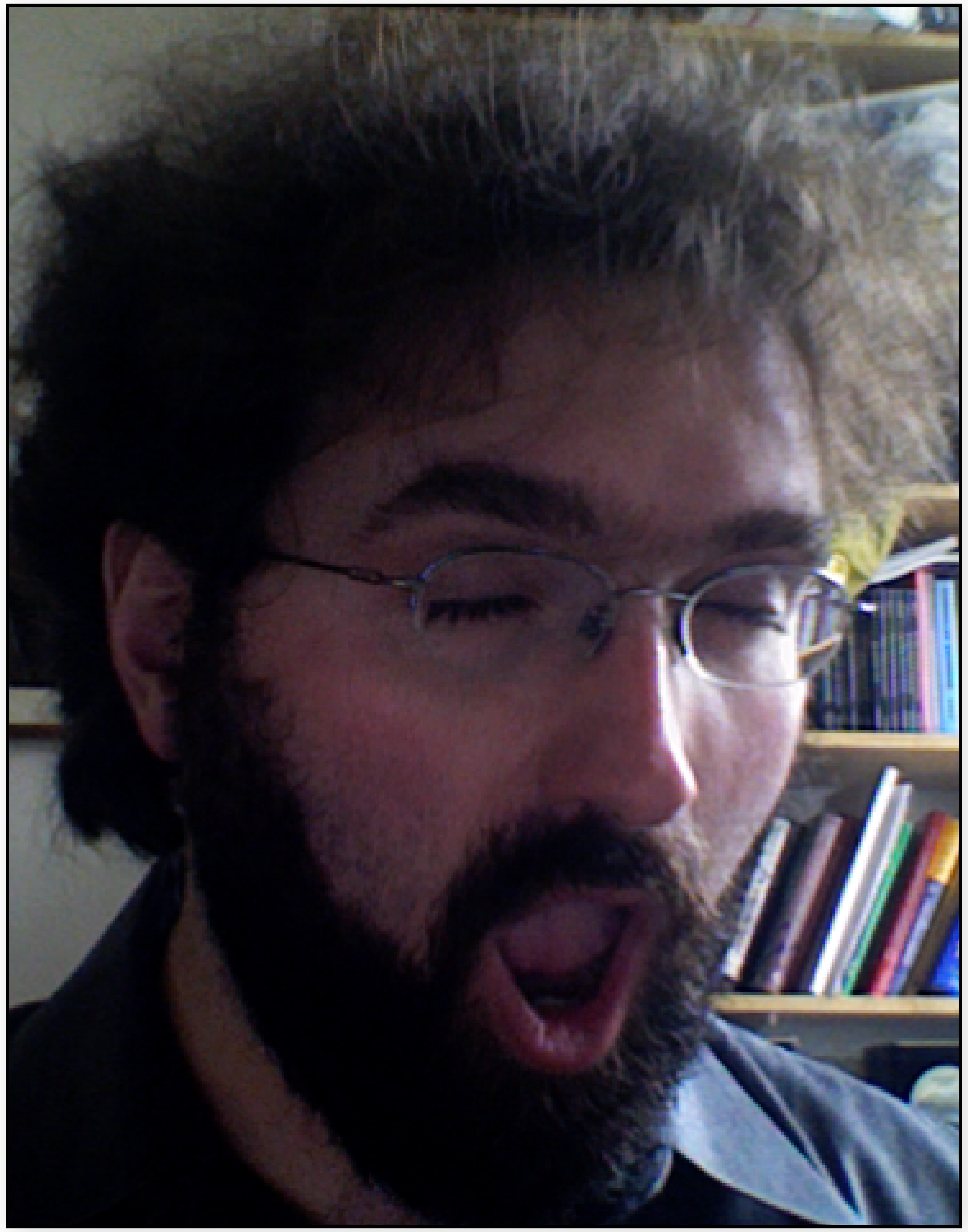
Warping & Morphing

Important scientific question Angry Fredo

How to turn Dr. Jekyll into Mr. Hyde?

How to turn a man into a werewolf?

Powerpoint cross-fading?



Important scientific question

American Werewolf
in London

How to turn Dr. Jekyll into Mr. Hyde?

How to turn a man into a werewolf?

Powerpoint cross-fading?

or

Image Warping & Morphing

Digression: old metamorphoses

http://en.wikipedia.org/wiki/The_Strange_Case_of_Dr._Jekyll_and_Mr._Hyde

http://www.eatmybrains.com/showtopten.php?id=15

http://www.horror-wood.com/next_gen_jekyll.htm

Unless I'm mistaken, both employ the trick of making already-applied makeup turn visible via changes in the color of the lighting, something that works only in black-and-white cinematography. It's an interesting alternative to the more familiar Wolf Man time-lapse dissolves. This technique was used to great effect on Fredric March in Rouben Mamoulian's 1932 film of Dr. Jekyll and Mr. Hyde, although Spencer Tracy eschewed extreme makeup for his 1941 portrayal.

Dr. Jekyll and Mr. Hyde, 1932

Dr. Jekyll and Mr. Hyde, 1932

Dr. Jekyll and Mr. Hyde, 1932

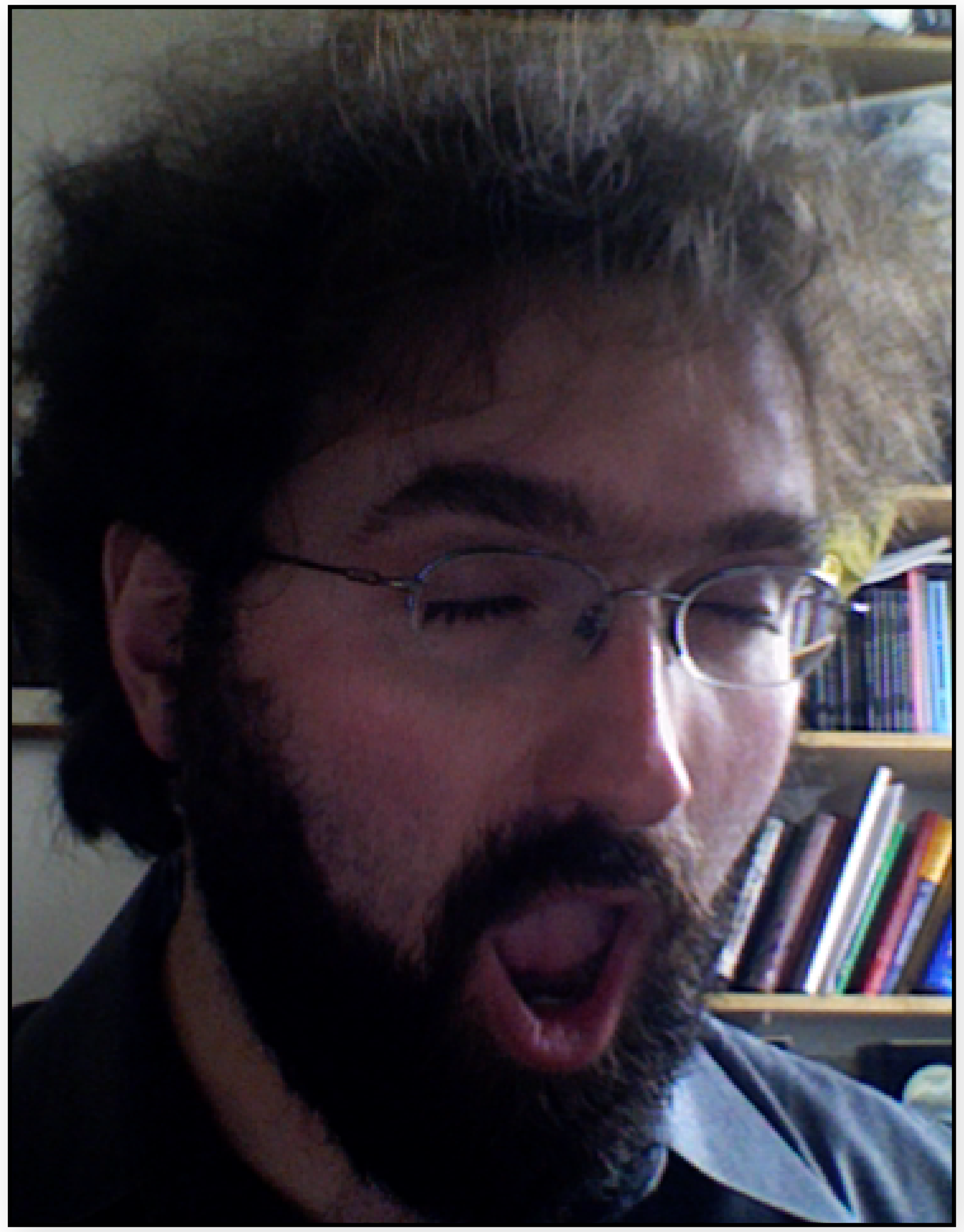
Dr. Jekyll and Mr. Hyde, 1941

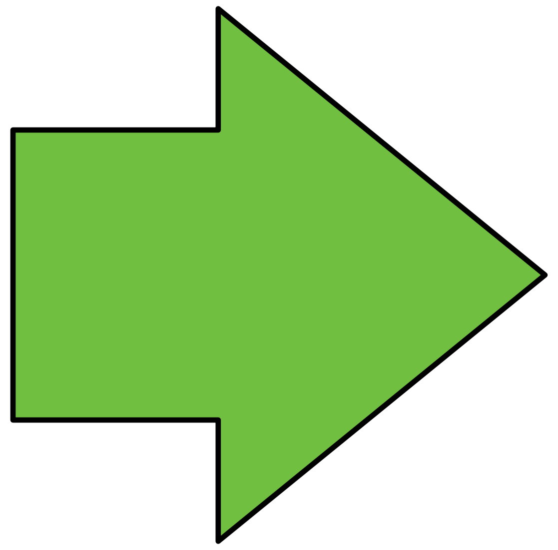
Challenge

"Smoothly" transform a face into another

Related: slow motion interpolation

- interpolate between key frames





Averaging images

Cross-fading

- output(x, y) = t * im1(x, y) + (1-t) * im2(x, y)

Problem with cross fading

Features (eyes, mouth, etc) are not aligned

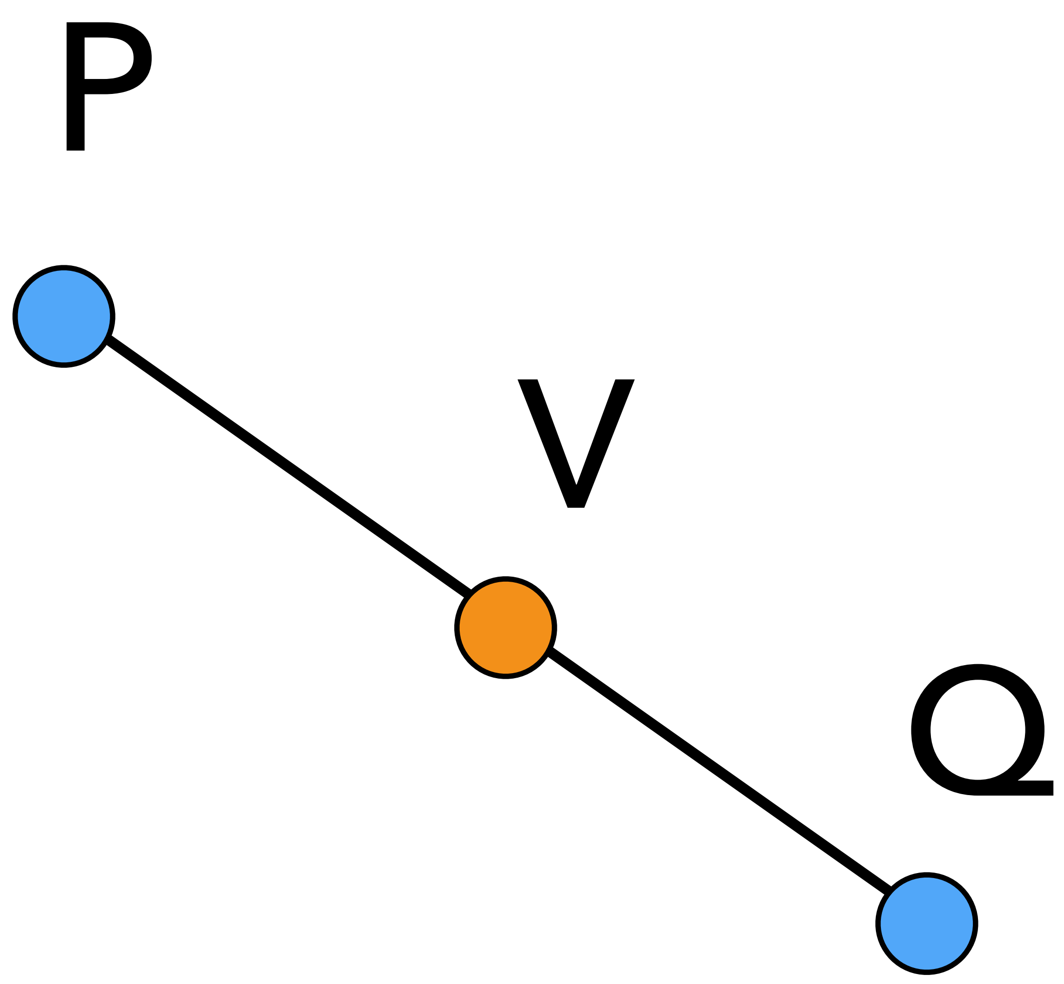
It is probably not possible to get a global alignment

We need to interpolate the LOCATION of features

Averaging points (location)

P & Q are two 2D points (in the "domain")

$V = t\,P + (1\text{-}t)\,Q$



Warping

Move pixel spatially: C′(x,y) = C(f(x,y))

Leave colors unchanged

Warping

Deform the domain of images (not range)

Central to morphing

Also useful for

- Optical aberration correction
- Video stabilization
- Slimming people down

original image

DxO Optics Pro correction

Recap & questions

Color (range) interpolation (lerp):

- output(x, y) = t * im1(x, y) + (1-t) * im2(x, y)

Location (domain) interpolation (lerp):

- V= t P + (1-t) Q

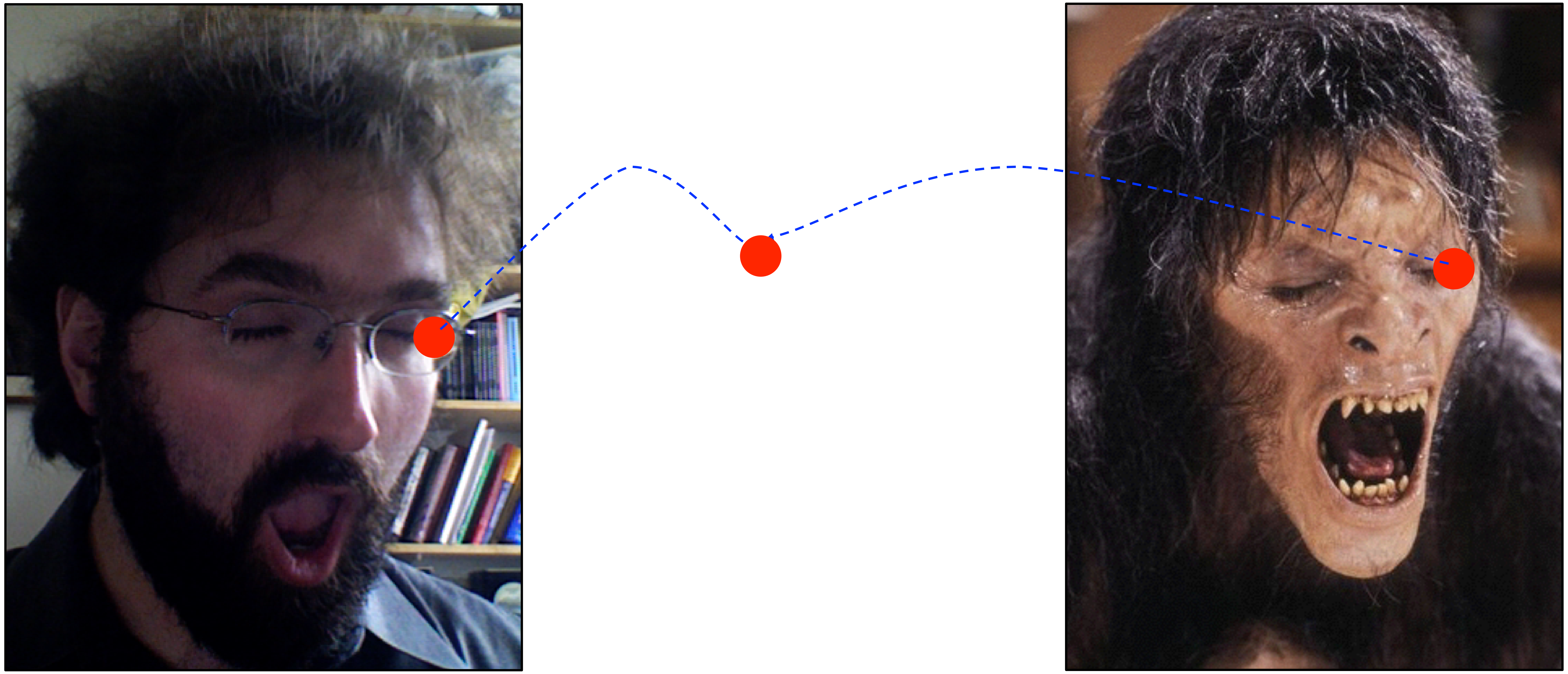
Warping: domain transform

- out(x,y)=im(f-1(x,y))

Morphing: combine both

For each pixel

- Transform its location like a vector (domain)
- Then linearly interpolate colors (range)



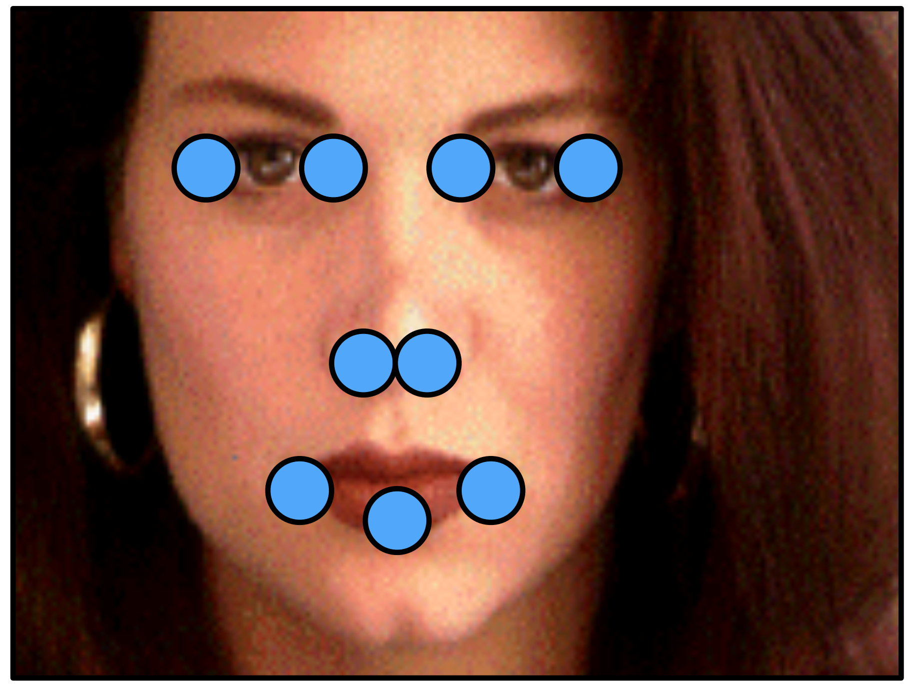
Morphing

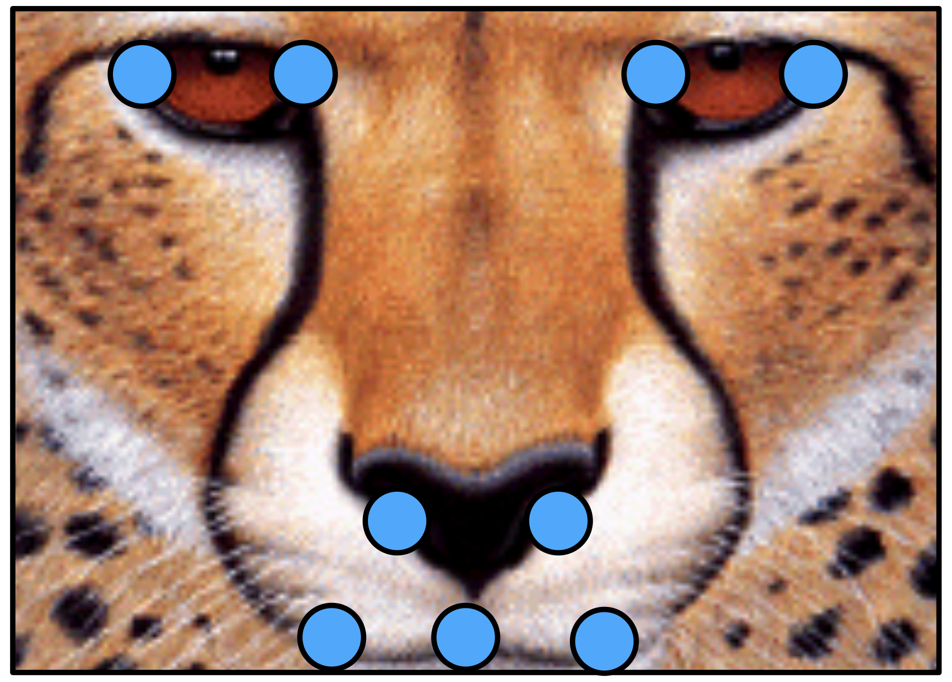
Input: two images I_0 and I_1

Expected output:

- image sequence I_t, with $t \in]0,1[$

User specifies sparse correspondences on the images



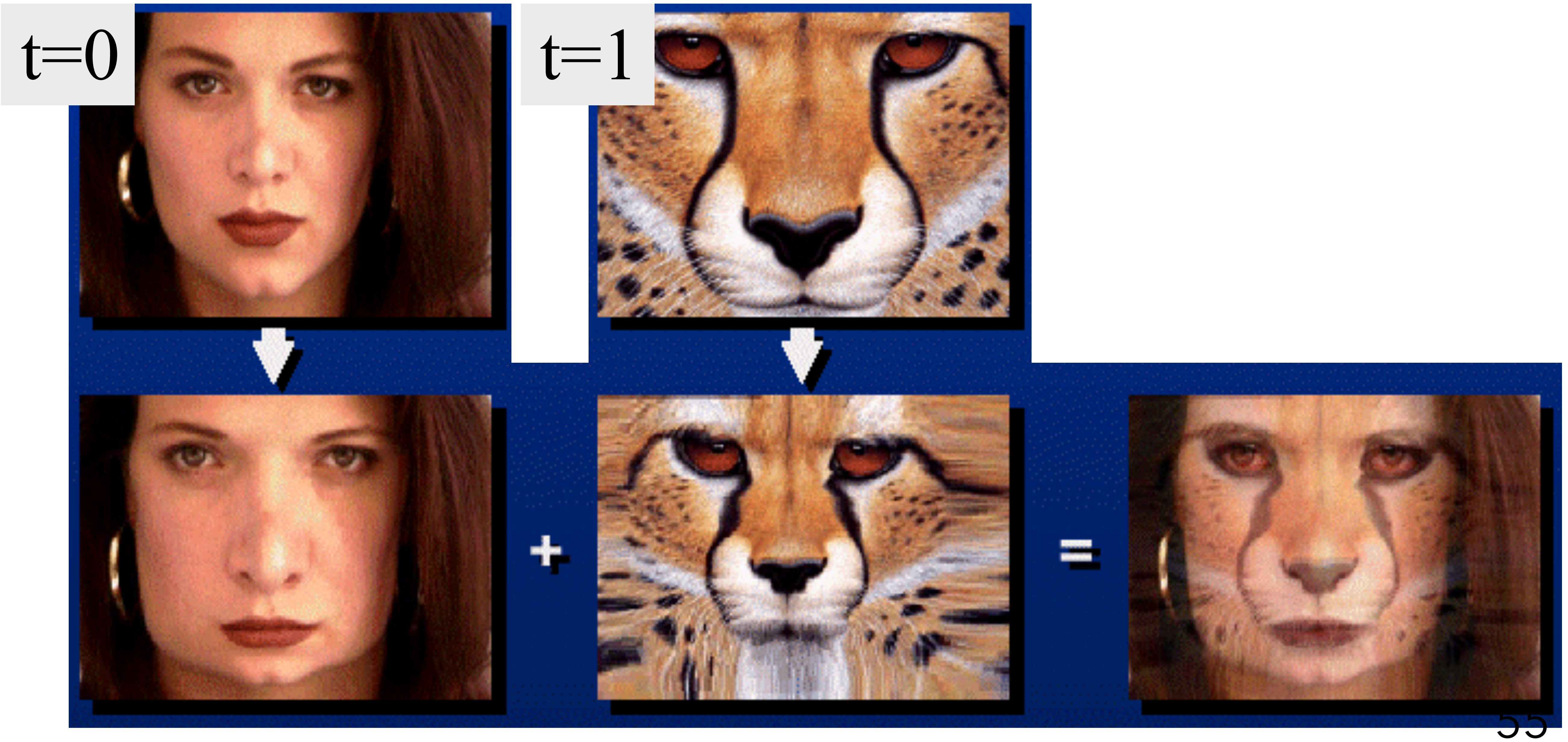


Morphing

For each intermediate frame I_t

- Interpolate feature locations $P^t_i = (1 - t)\, P^0_i + t\, P^1_i$
- Perform two warps: one for I_0, one for I_1
  - Deduce a dense warp field from the pairs of features
  - Warp the pixels
- Linearly interpolate the two warped images

t=0.5

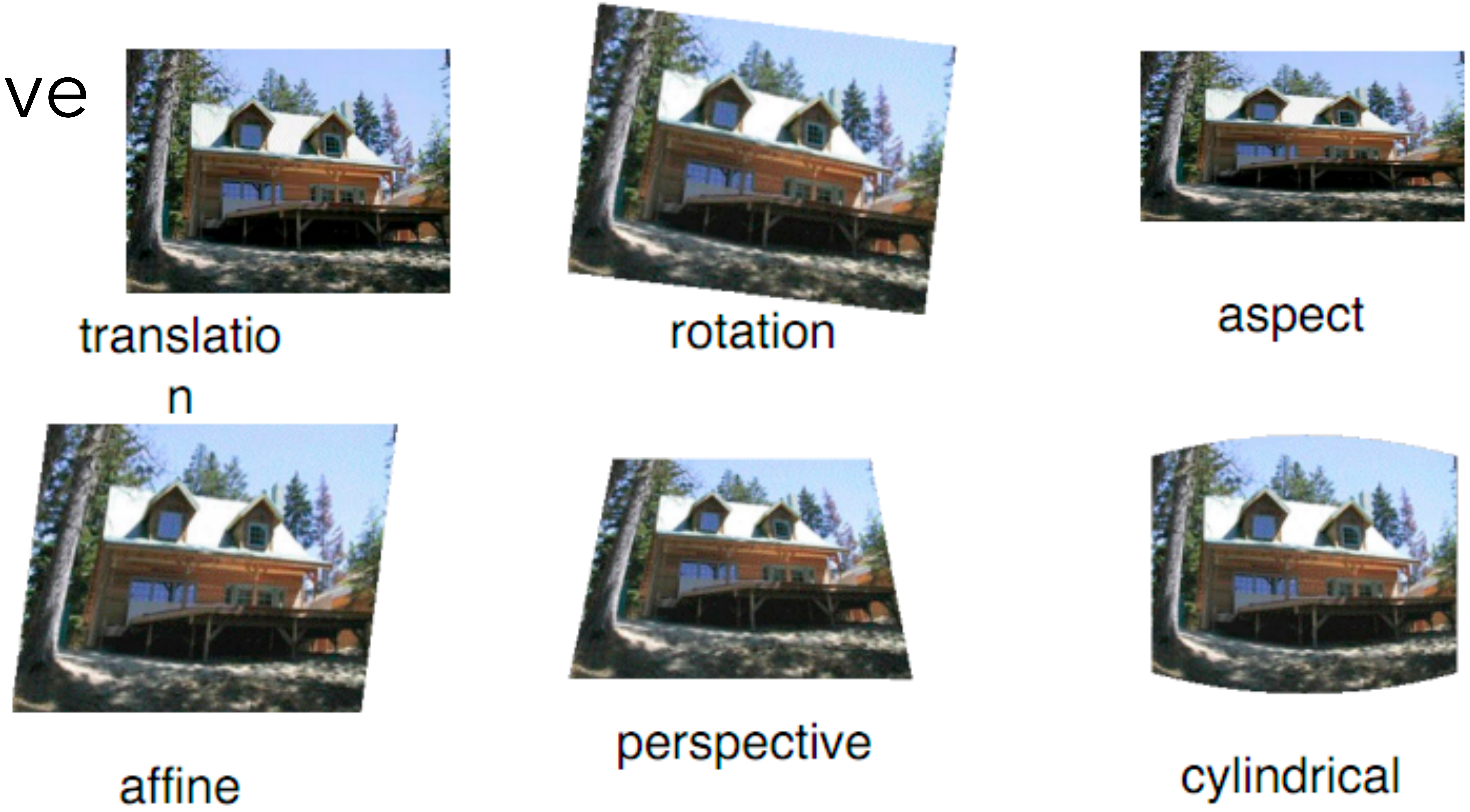


Warping

How do we specify the warp?

Before, we saw simple transformations

- linear, affine, perspective



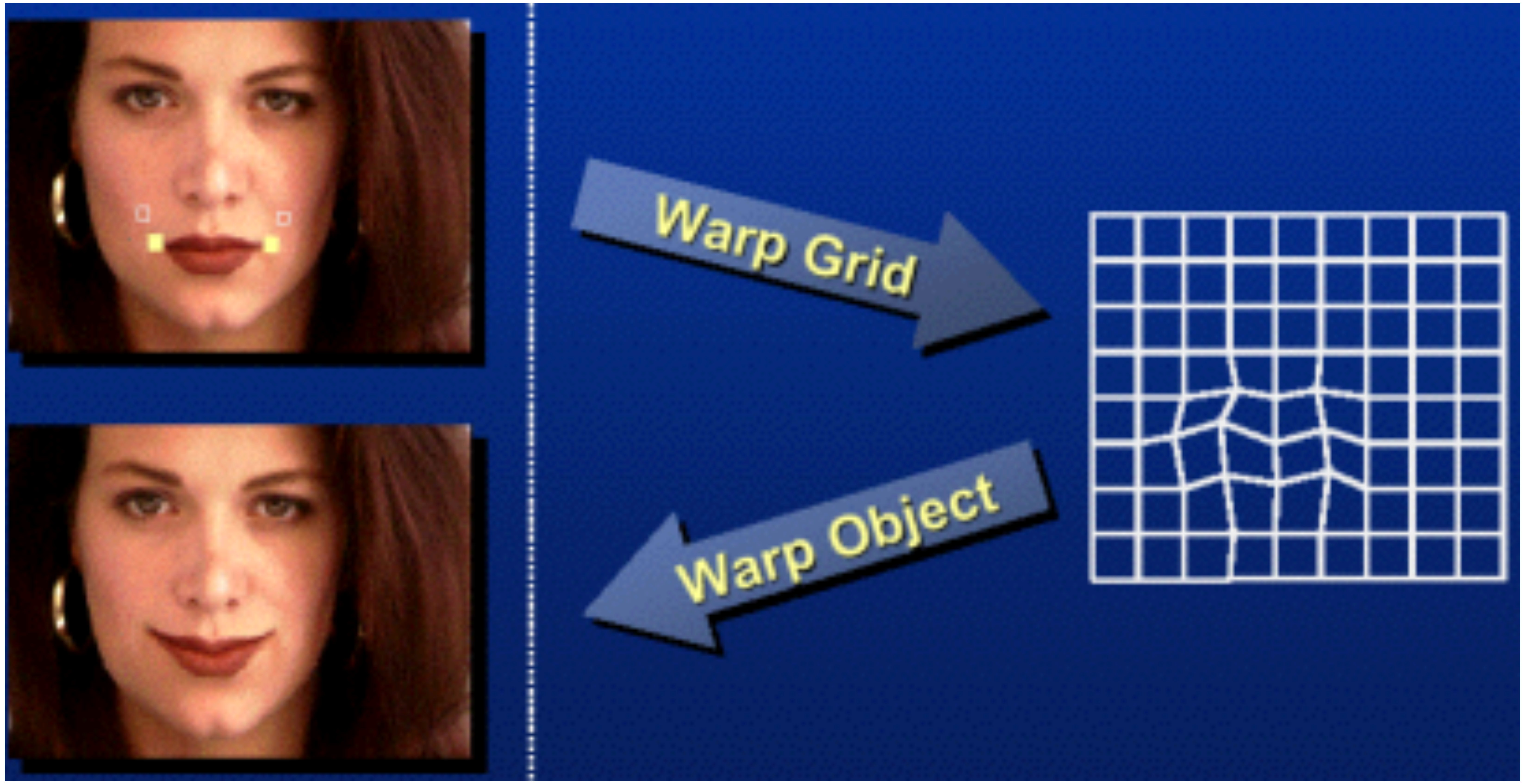
But we want more flexibility

illustration by Rick Szeliski

Image Warping – parametric

Move control points to specify a spline warp

Spline produces a smooth vector field



Warp specification - dense

How can we specify the warp?

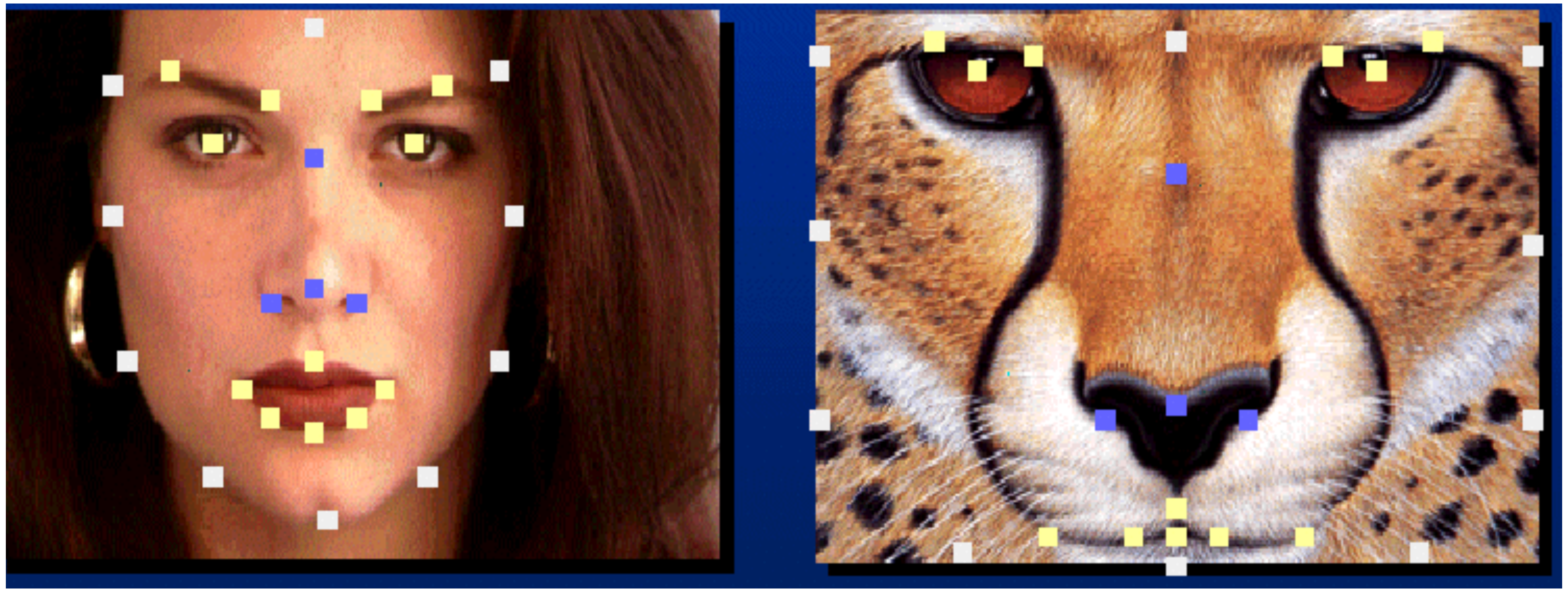
- Specify corresponding spline control points
    - interpolate to a complete warping function

But we want to specify only a few points, not a grid

Warp specification - sparse

How can we specify the warp?

- Specify corresponding points
  - interpolate to a complete warping function

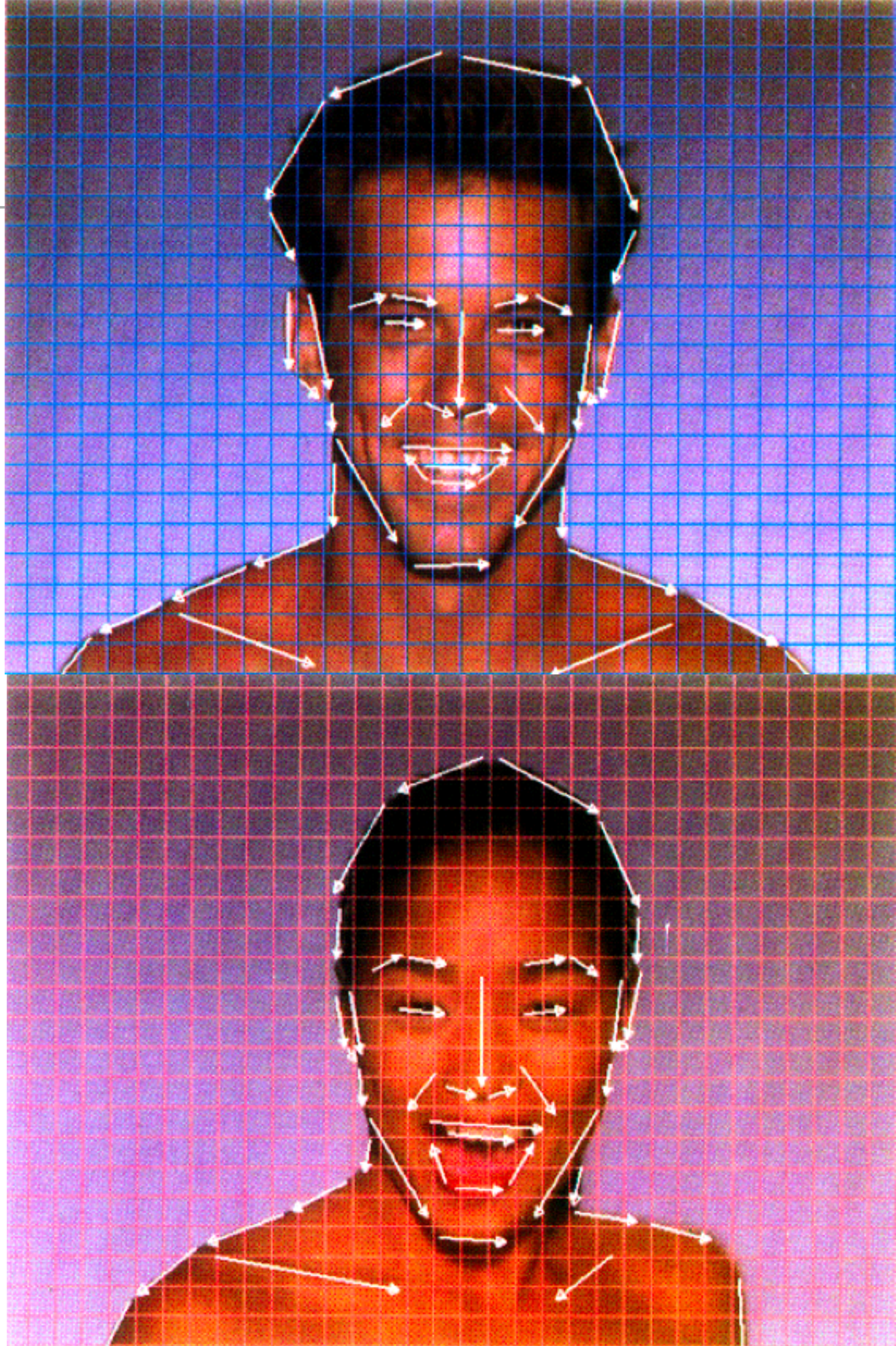


How do we go from feature points to pixels?

Beier and Neely

Specify warp based on pairs of segments

- "Feature-Based Metamorphosis", SIGGRAPH 1992
- Used in Michael Jackson's "Black and White" music video
- Assignment 6!!



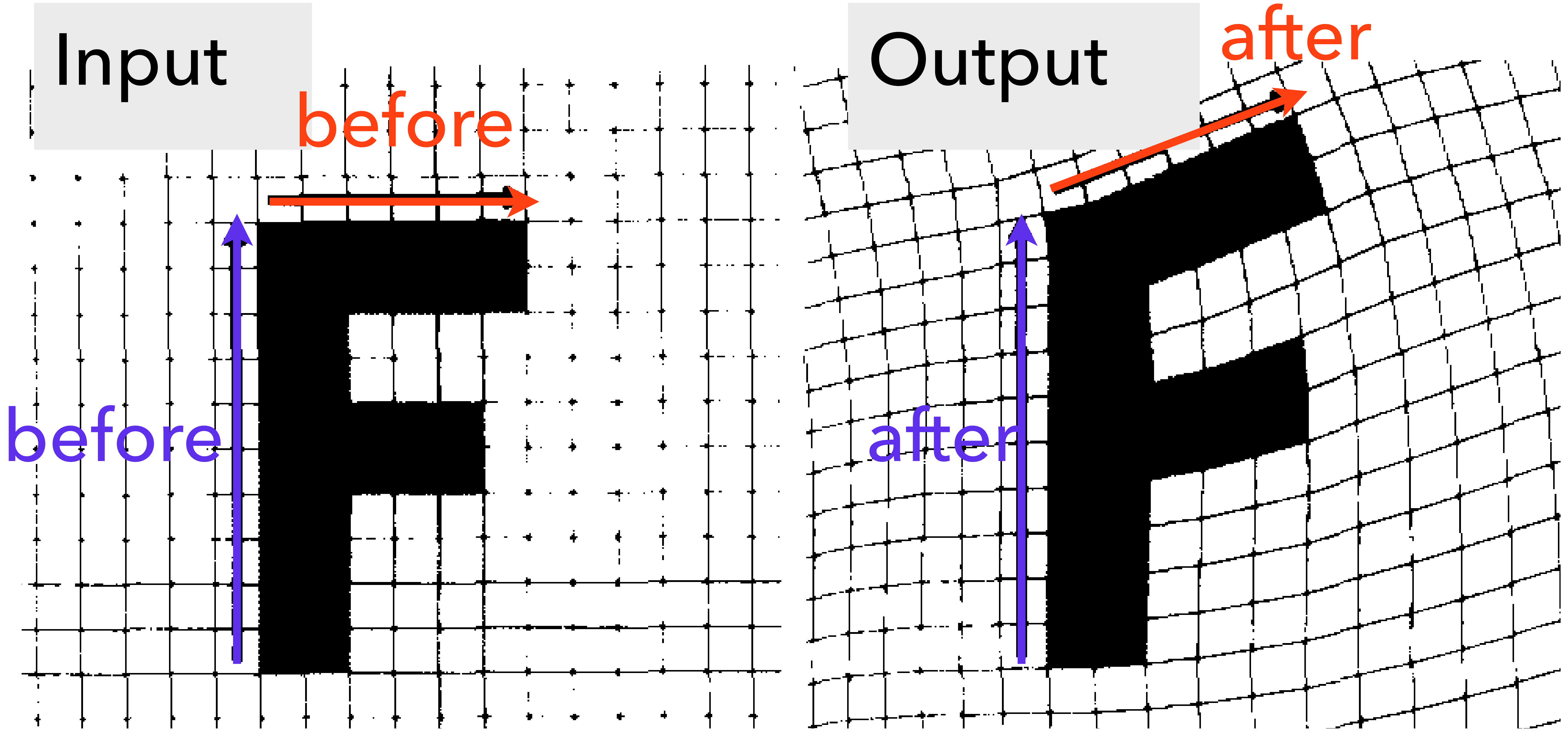
Questions?

Segment-based warping

Problem statement

Inputs: One image, two lists of segments before and after, in the image domain

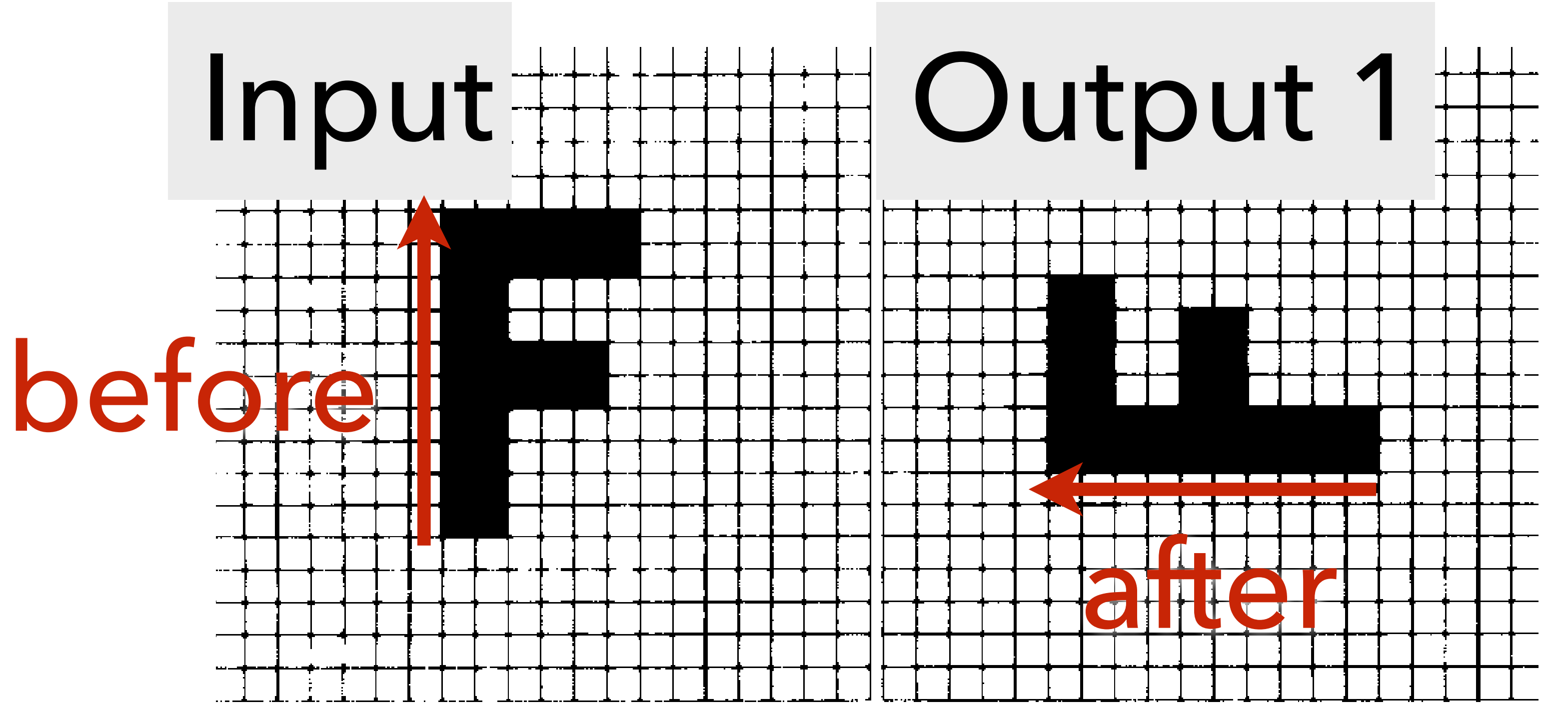
Goal: warp the image "following" the displacement of the segments



Idea

Each before/after pair of segment implies a planar transformation

- simple and linear

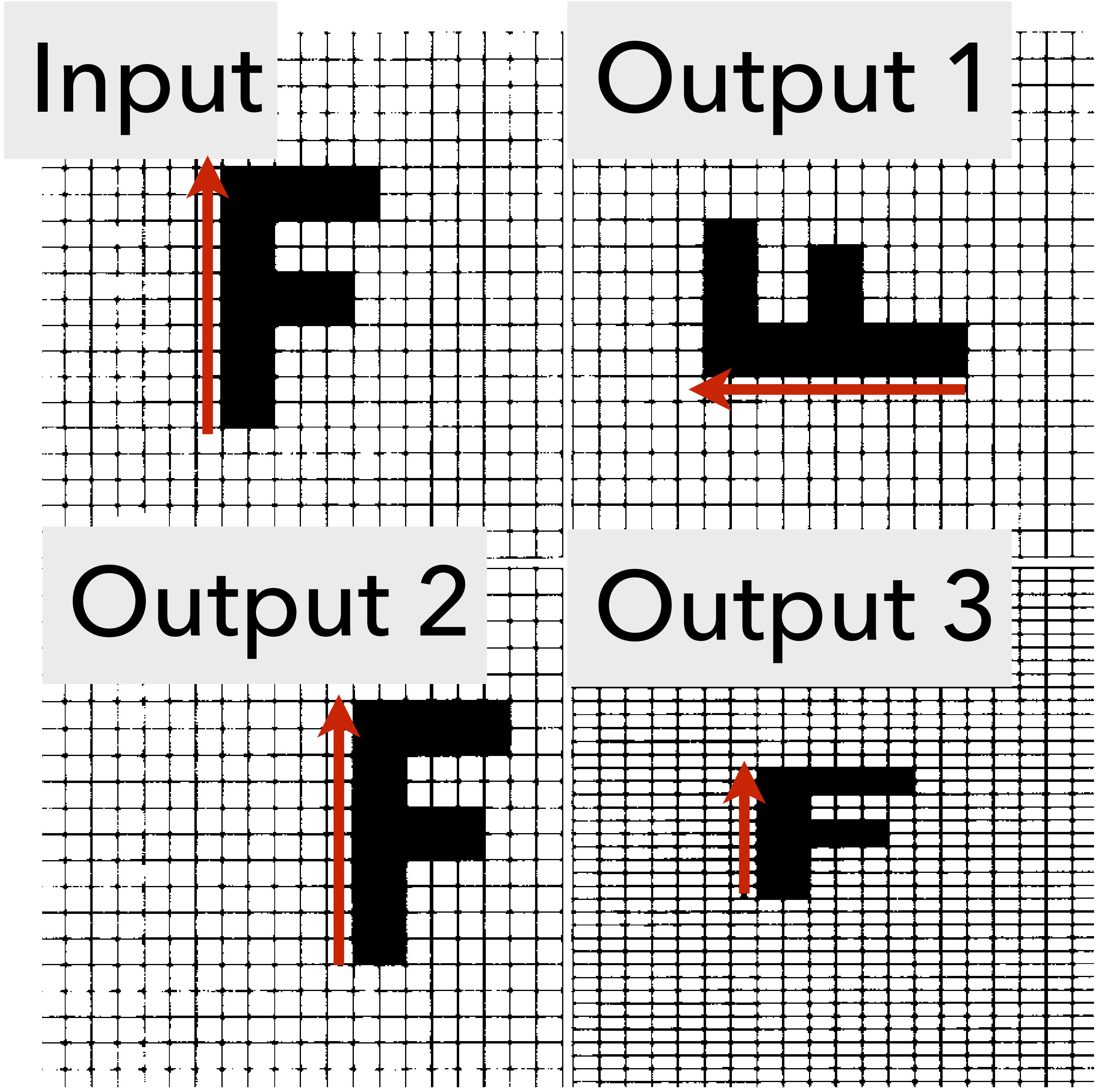


Single line transforms

Idea

Each before/after pair of segment implies a planar transformation

- simple and linear



Single line transforms

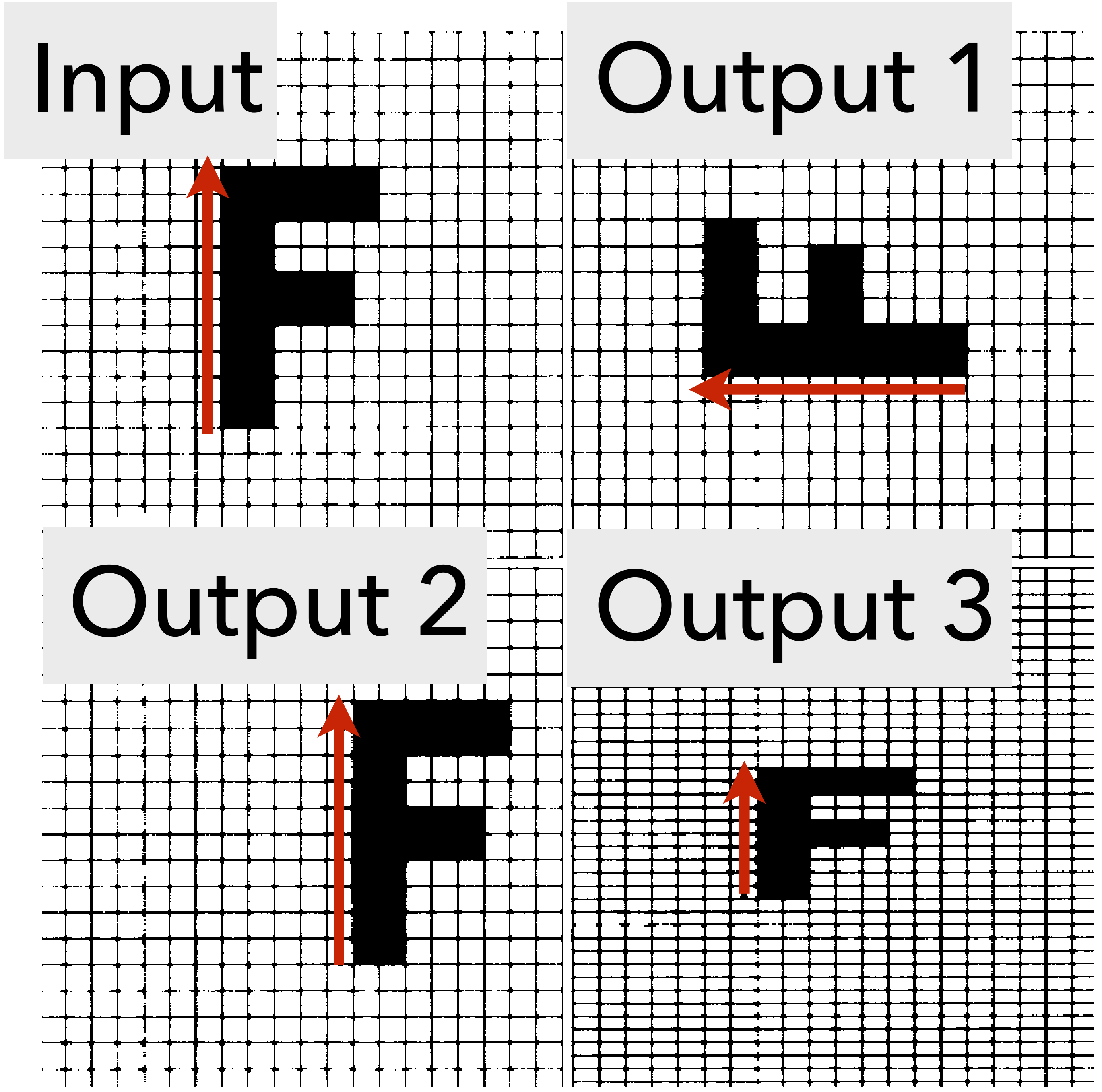
Test

$\rightarrow$ `warpBy1(im, segment(0,0, 10,0), segment(10, 10, 30, 15))` $\rightarrow$

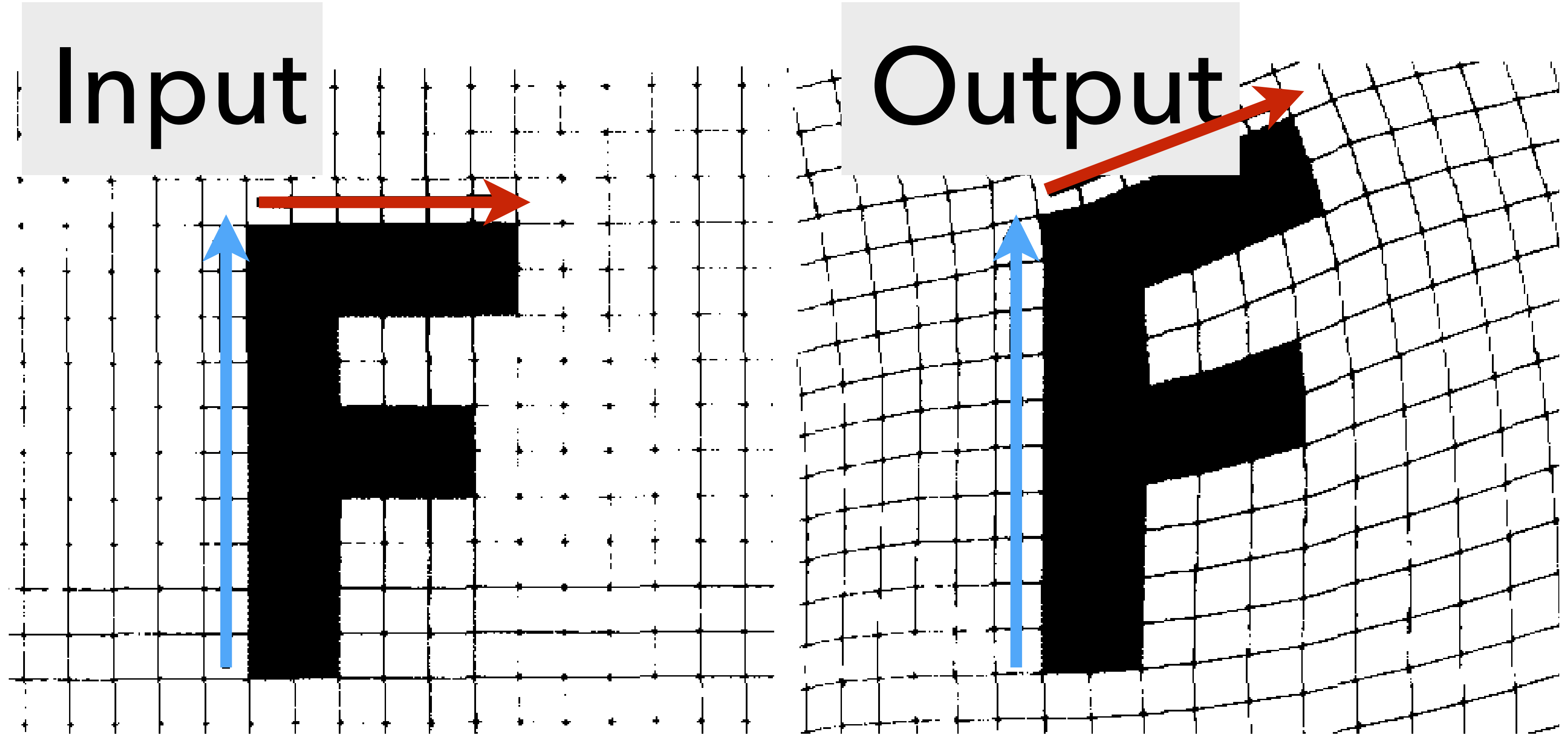
Idea

Each before/after pair of segment implies a planar transformation

Then take weighted average of transformations



Single line transforms



Transform wrt 2 lines

Transform wrt 1 segment

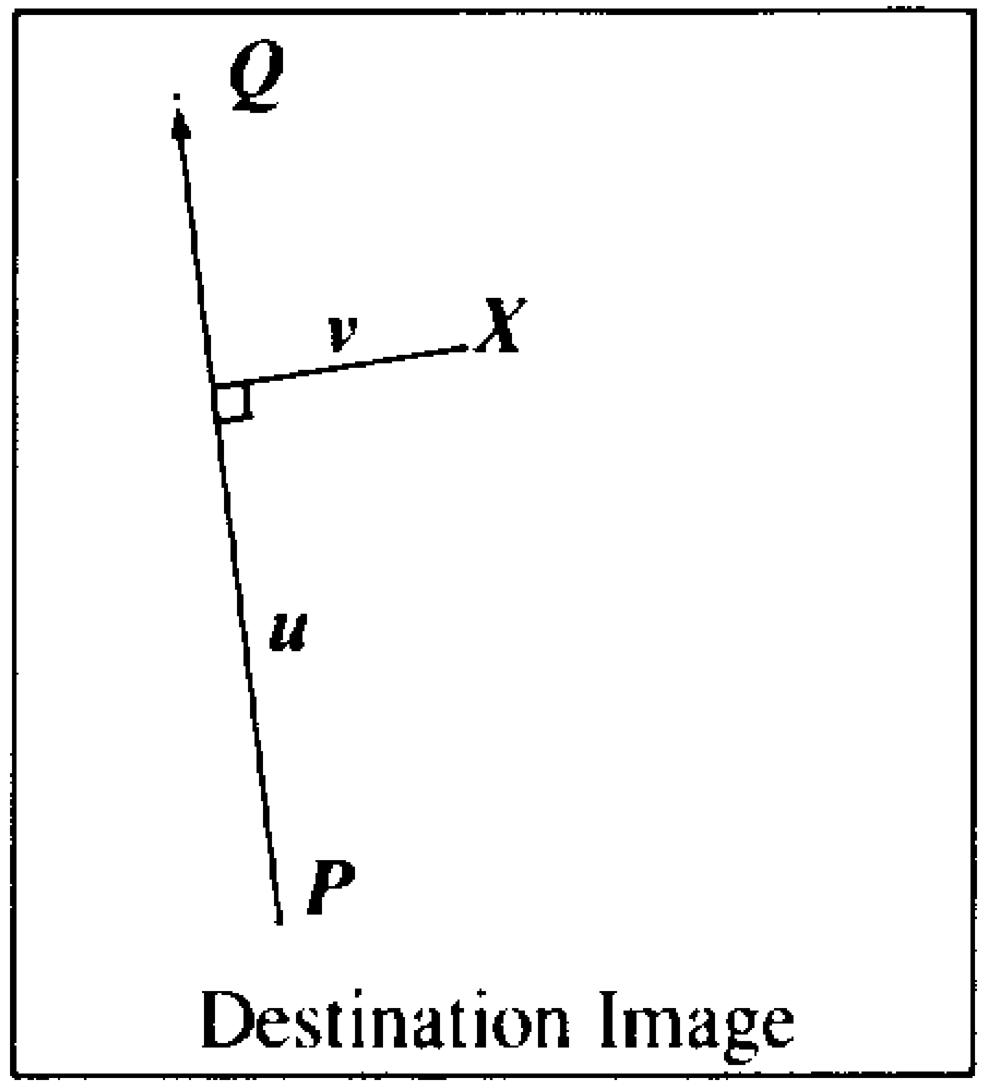
Define a coordinate system with respect to segment

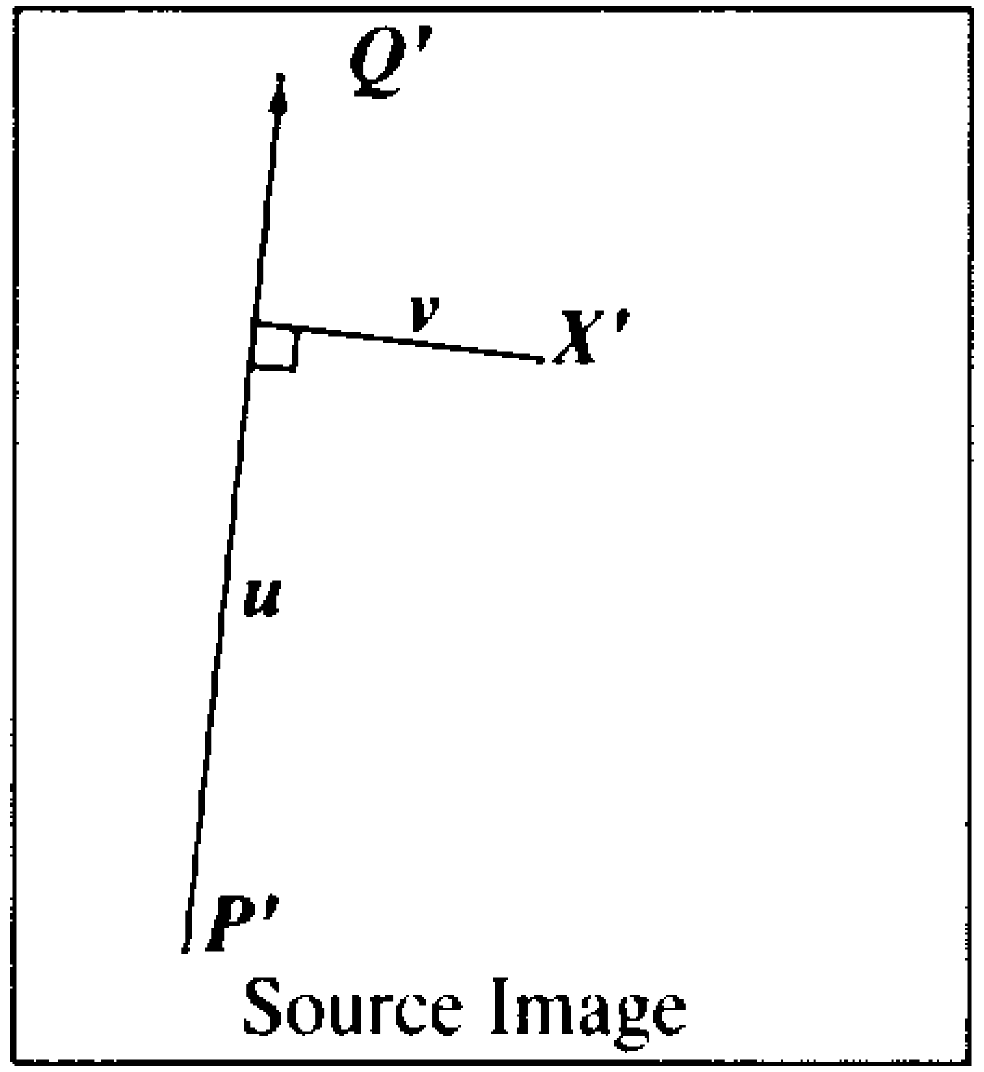
- 1 dimension, u, along segment
- 1 dimension, v, orthogonal to segment

Compute u, v in one image

- The after one, because we use the inverse transform

Compute point corresponding to u, v in second image





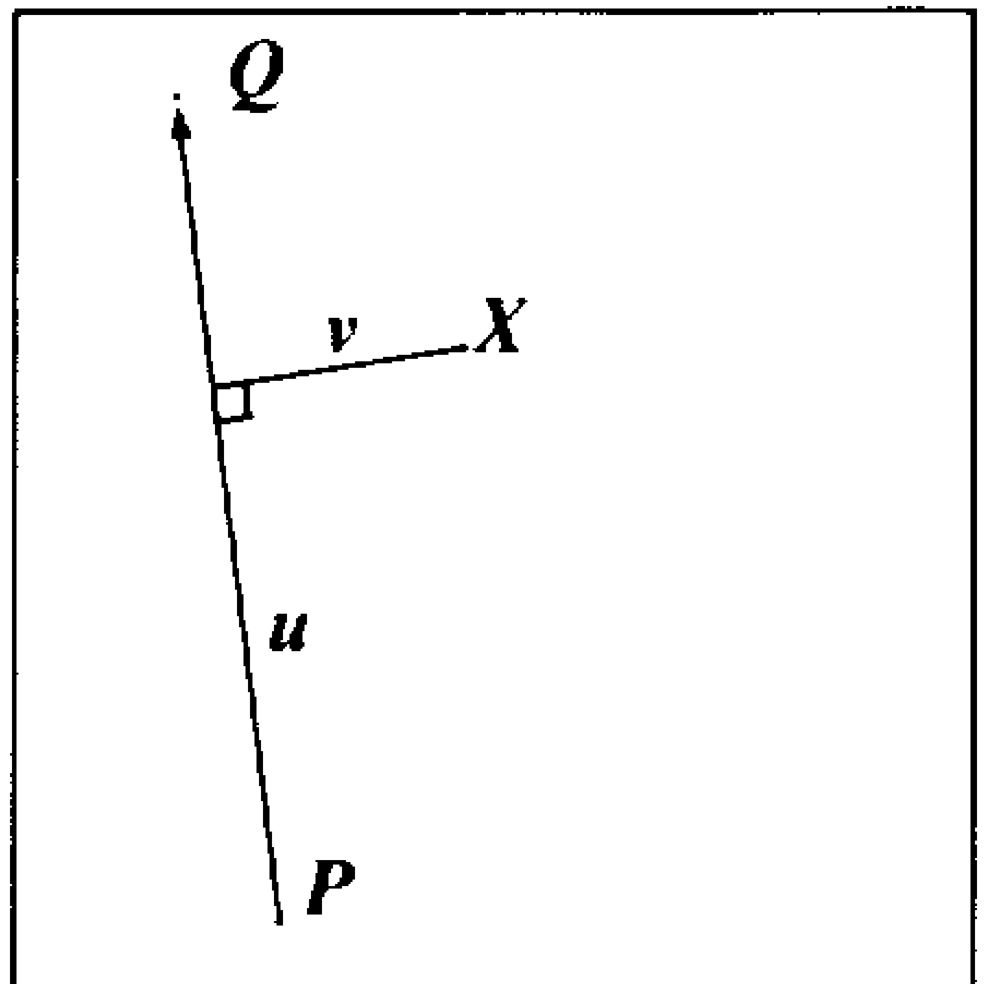
Computing u, v

$u = PX.PQ/||PQ||^2$

- this way u is 0 at P and 1 at Q

$v = PX.perpendicular(PQ)/||PQ||$

- where perpendicular(PQ) is PQ rotated by 90 degrees, and has length $||PQ||$
- unlike u which is normalized, v is in distance units



Destination Image

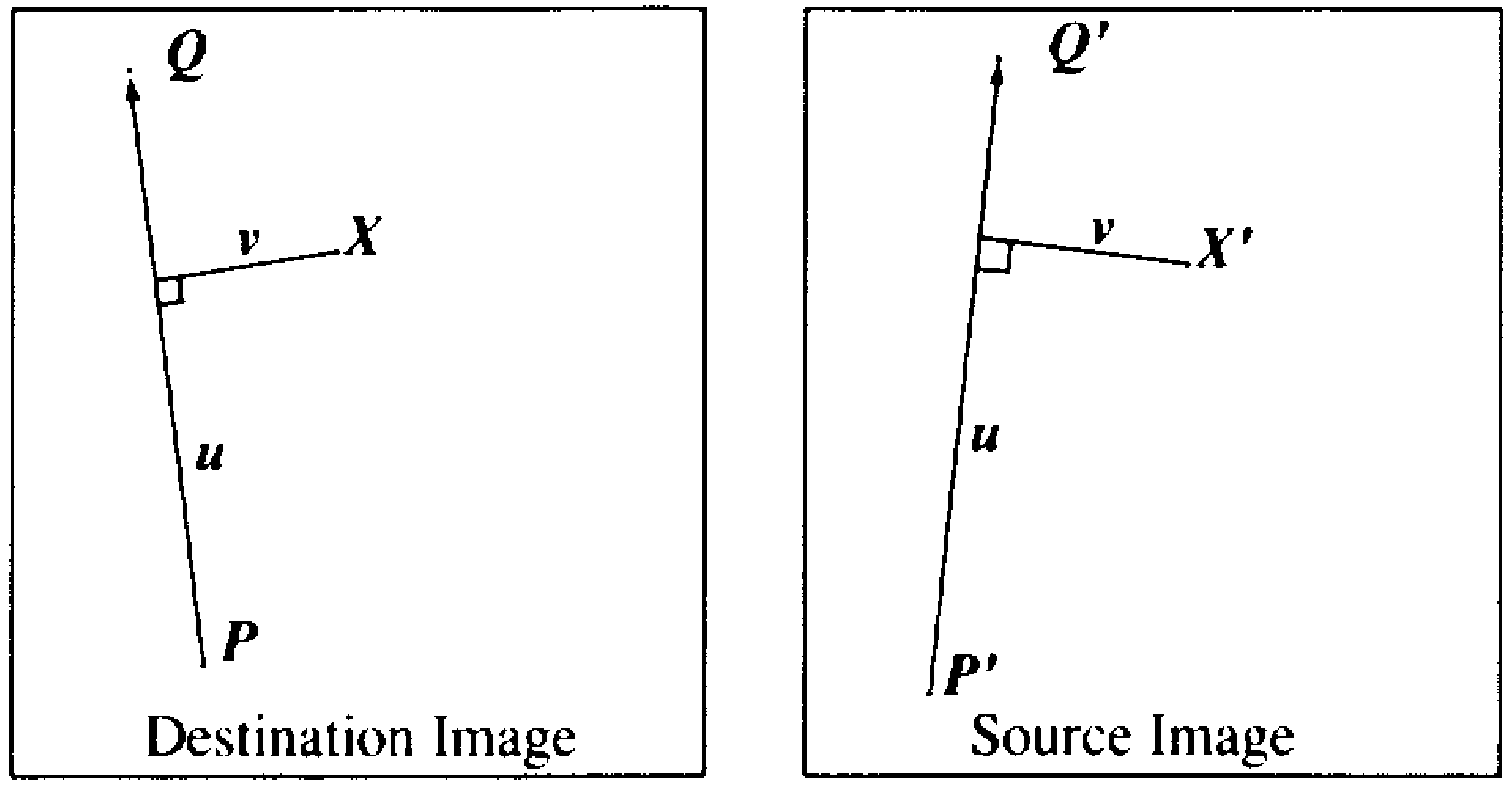
Transforming a point given u, v

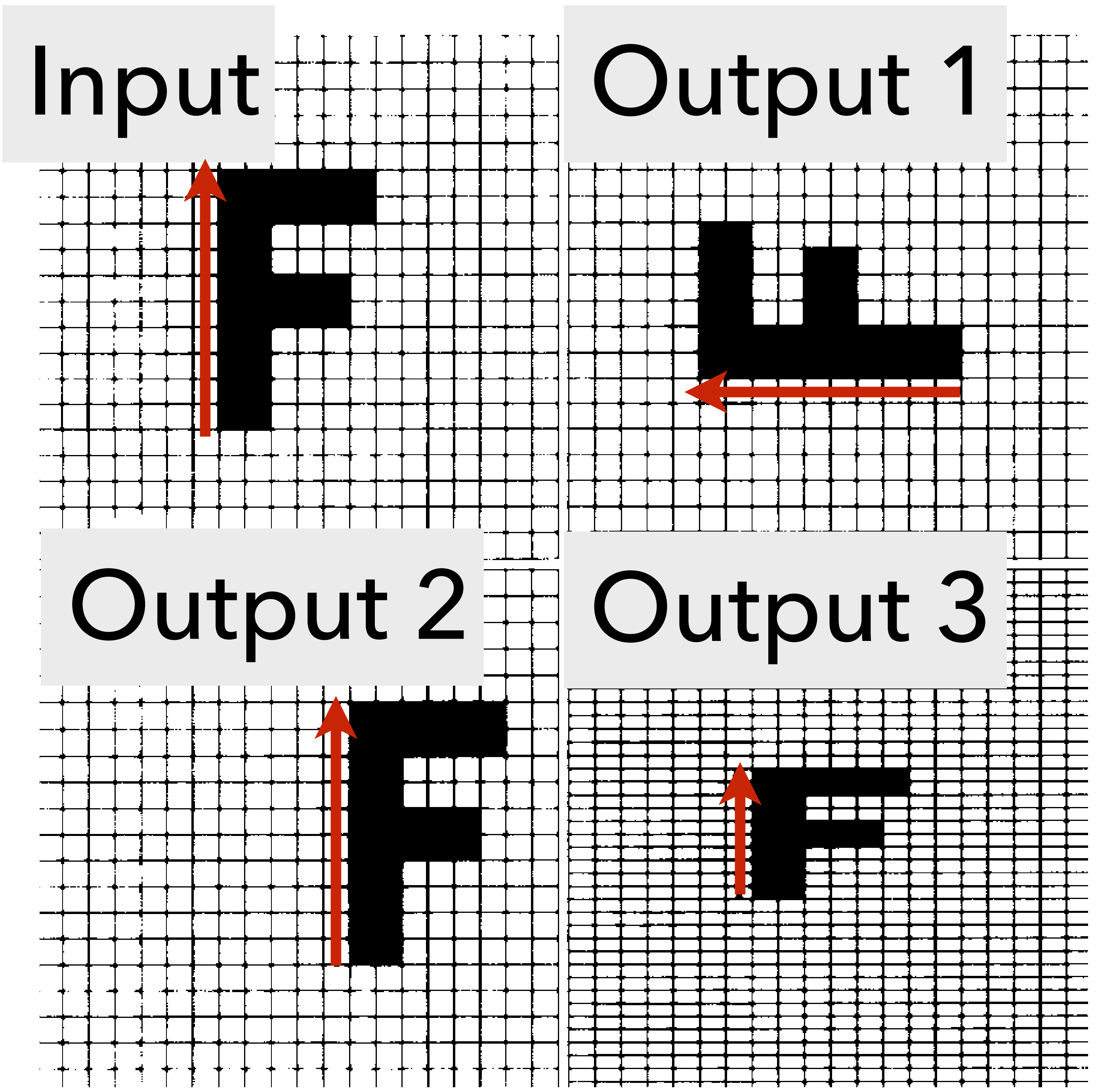
$X' = P' + u*P'Q + v*\text{perpendicular}(P'Q')/||P'Q'||$

The u component is scaled according to segment scaling

But v is absolute (see output3)

- They say they tried to scale v as well but it didn't work as well





Questions?

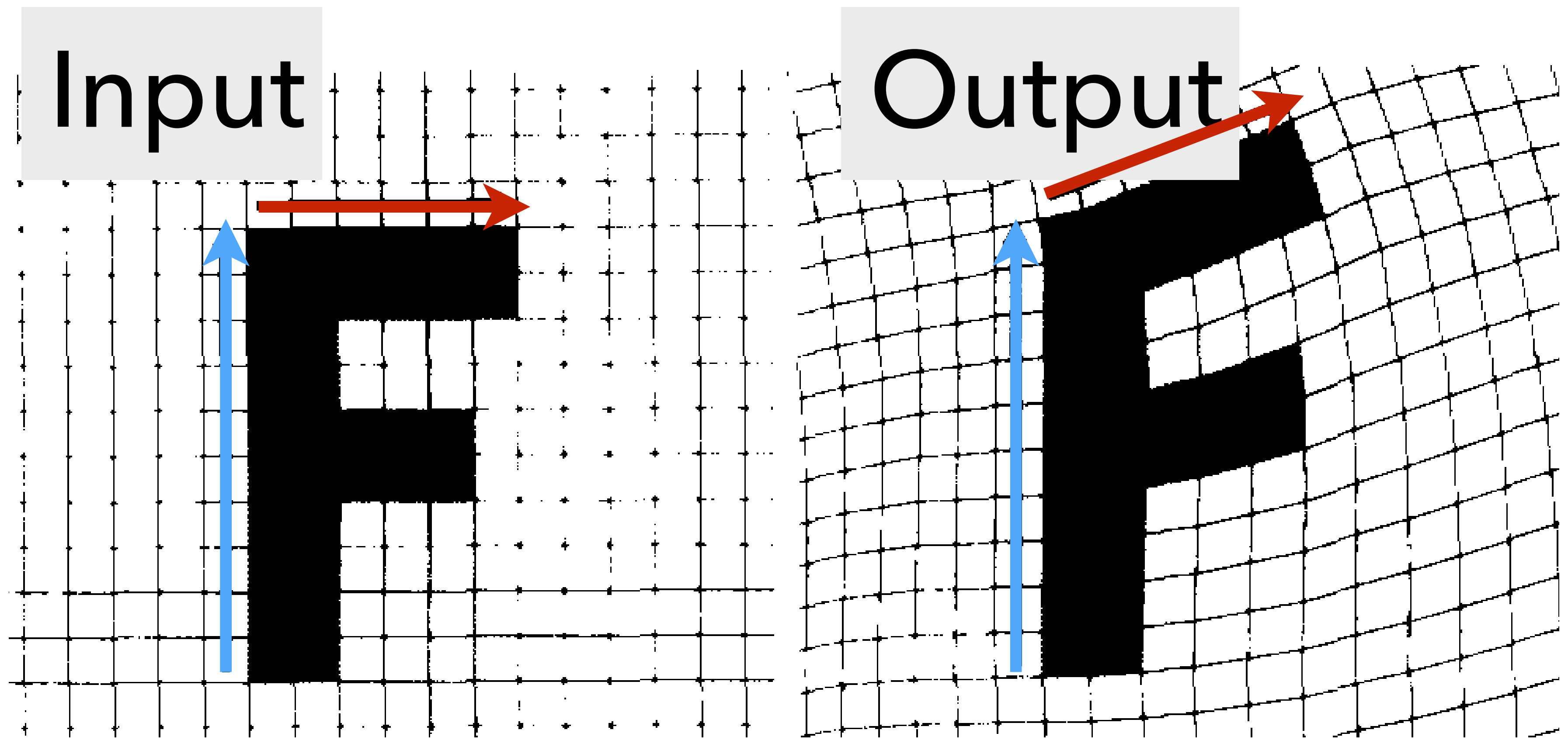
Multiple segments

For each point X

- For each segment pair sbefore[i], safter[i]
  - Transform X into X'i
- Compute weighted average of all transformed X'i
  - weight according to distance to segments

$$weight = \left(\frac{length^p}{a + dist}\right)^b$$

where a, b, p control the influence



Transform wrt 2 lines

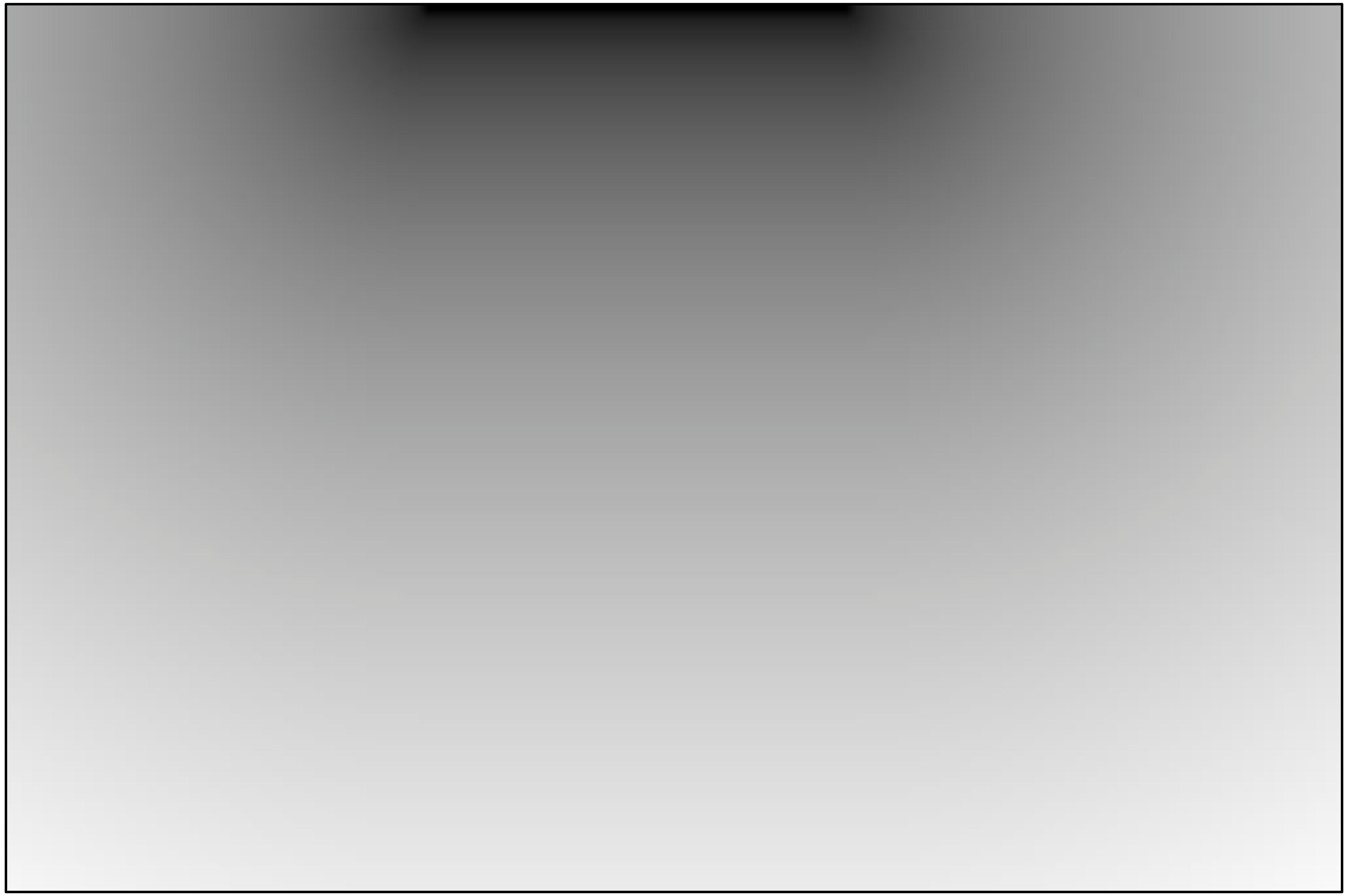
Distance to a segment

Multiple cases…

- dot product, test > 0, < 1

Debugging: example

Debugging my distance function

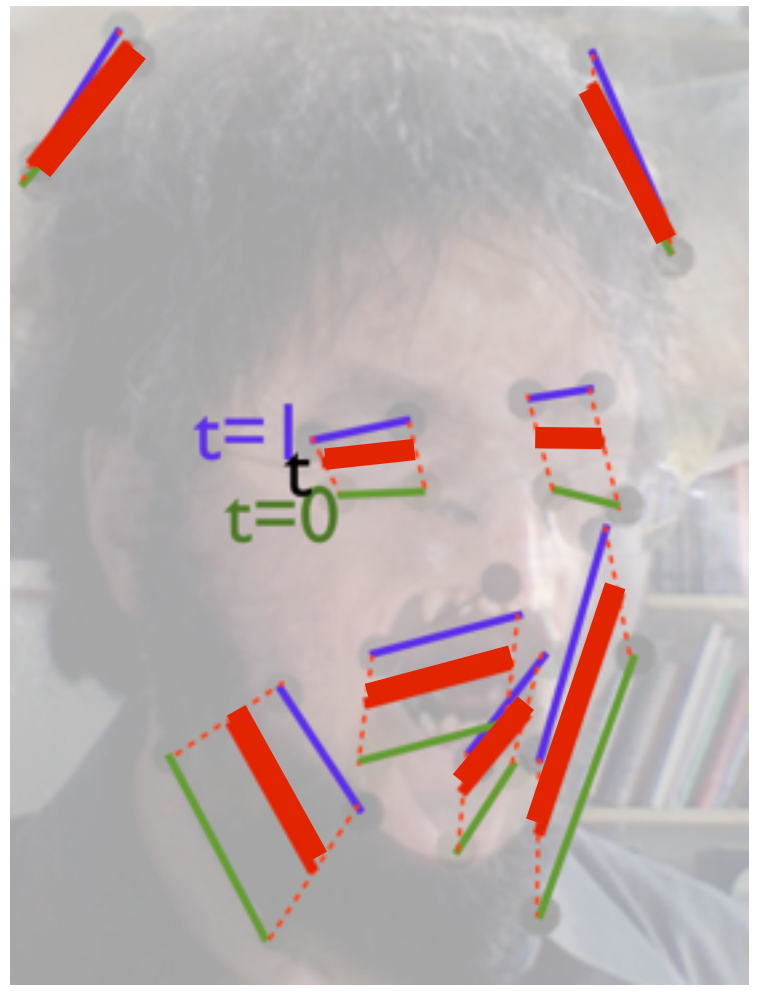


Morphing

Input images

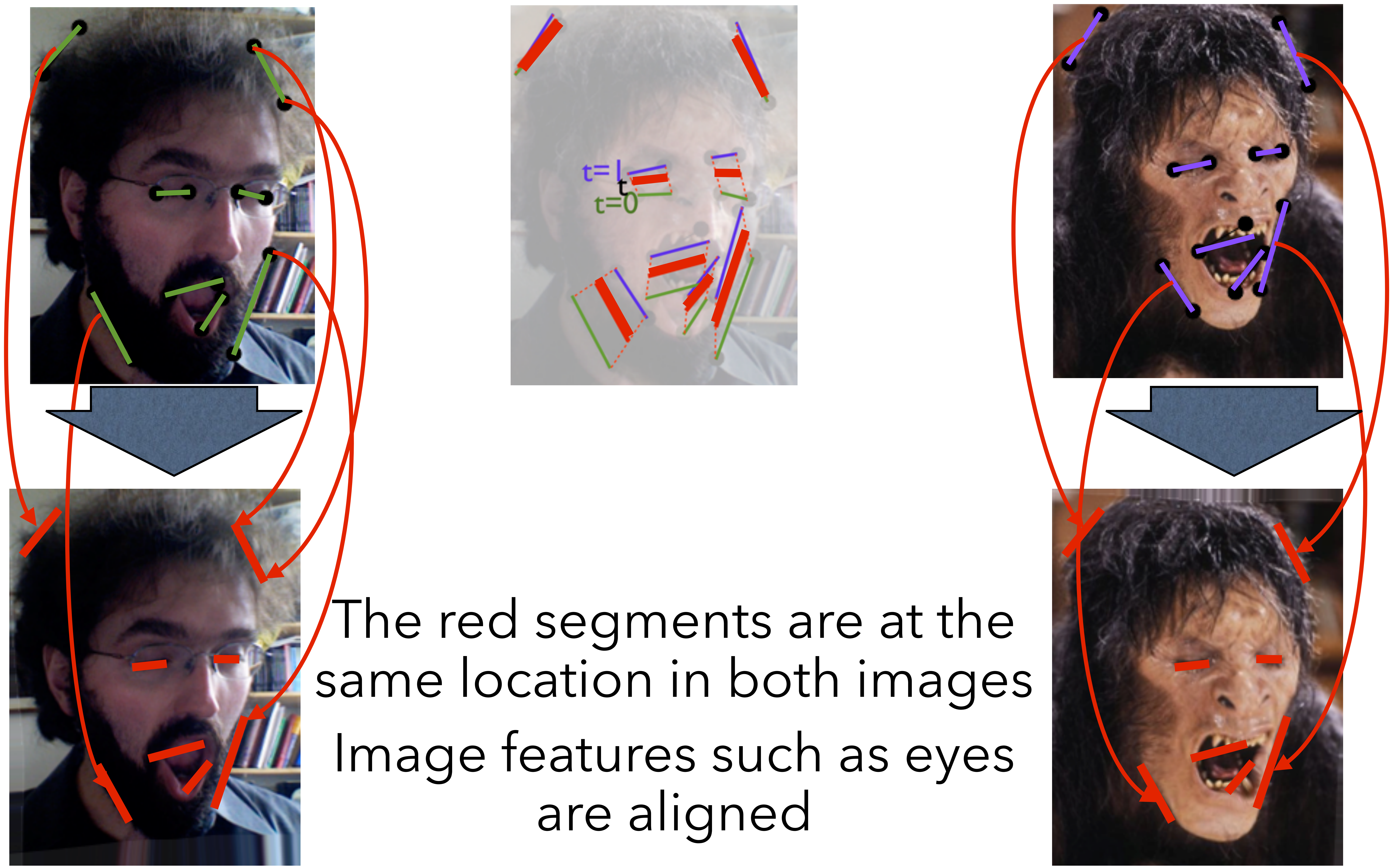
Segments

Interpolate segments



t=0.5

Warp images to segments[t]



The red segments are at the same location in both images

Image features such as eyes are aligned

Interpolate color

Result

Recap

For each intermediate frame I_t

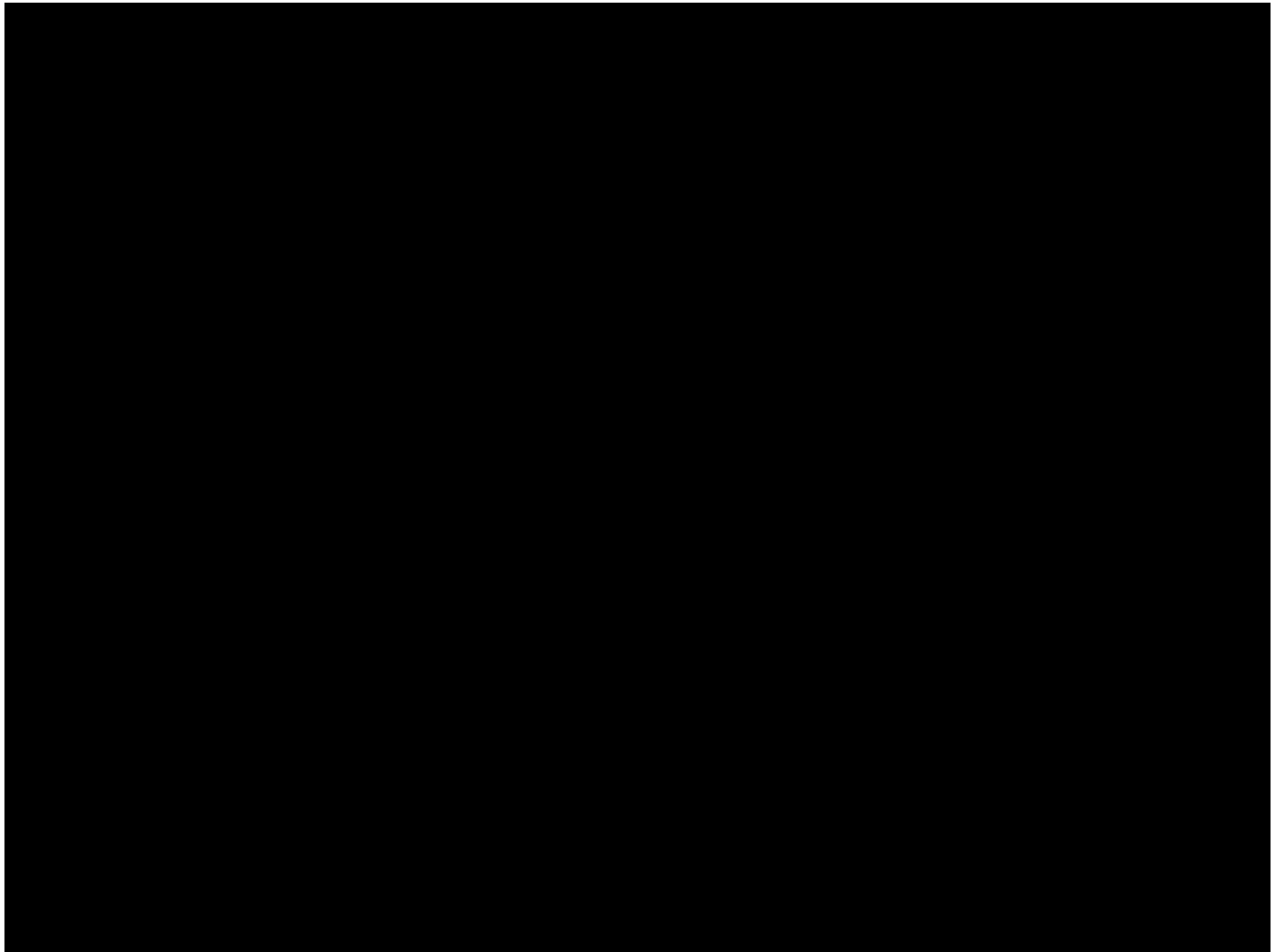
- Interpolate segment locations $y^t_i = (1 - t)\, x^0_i + t\, x^1_i$
- Perform two warps: one for I_0, one for I_1
  - Deduce a dense warp field from the pairs of features
  - Warp the pixels
- Linearly interpolate the two warped images

Michael Jackson' BW

Uses the very technique we just studied

More morphing madness

Gondry's Rolling Stones video



Women in Art video

http://youtube.com/watch?v=nUDIoN-_Hxs

Willow

1988, special effects by ILM (first use of morphing)

Slide credits

Frédo Durand

Marc Levoy