Morphing

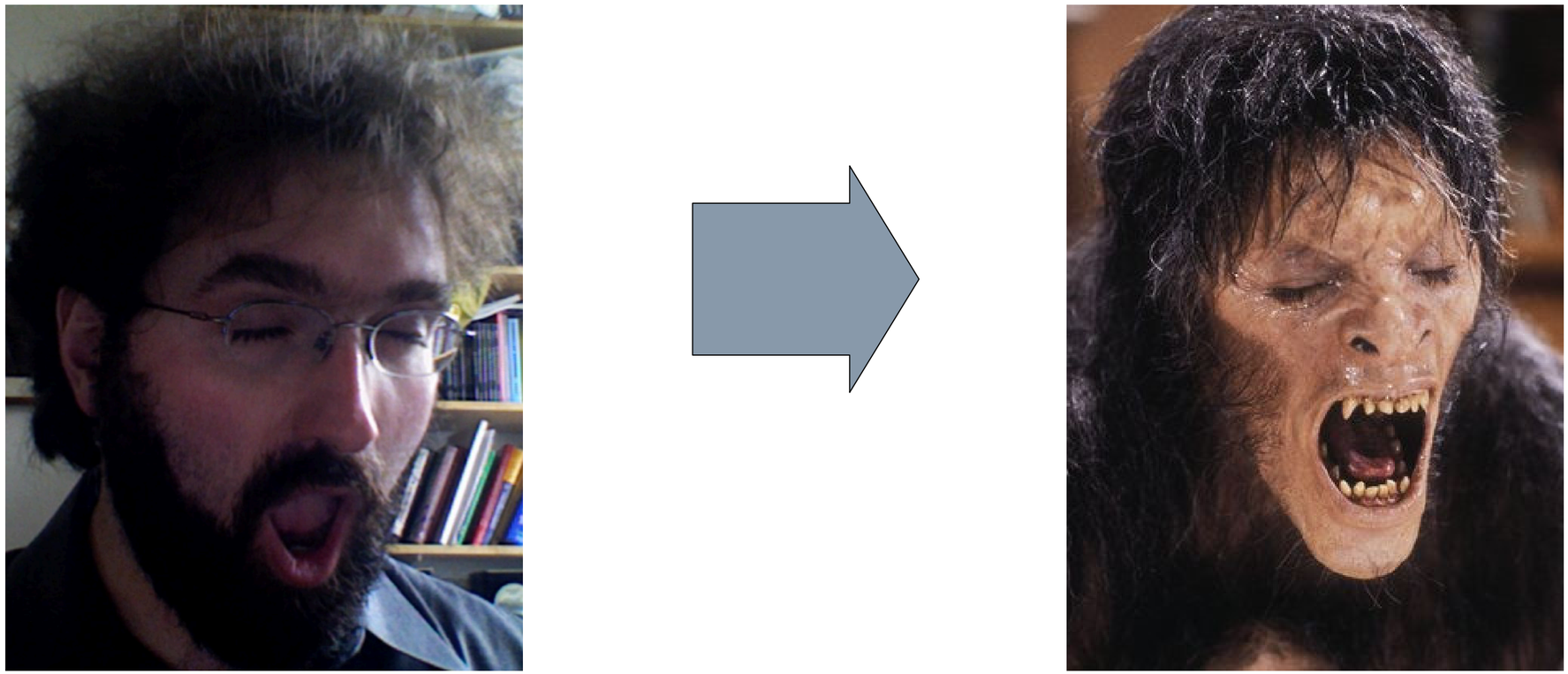
Fredo Durand
MIT EECS
6.8370/6.8371

No Frédo office hours tomorrow.

Sorry.

Challenge

- **“Smoothly” transform a face into another**
- **Related: slow motion interpolation interpolate between key frames**



Averaging images

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



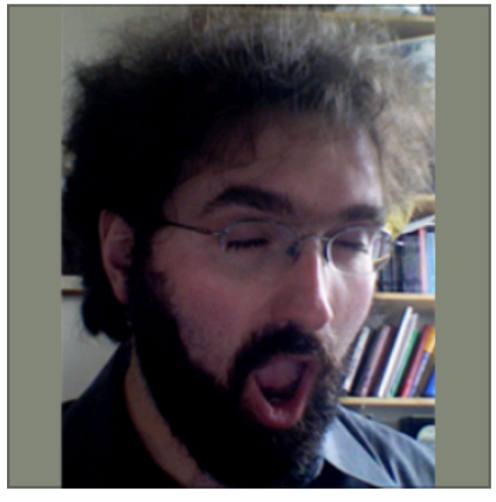
Problem with cross fading

- **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**
  - domain transform!

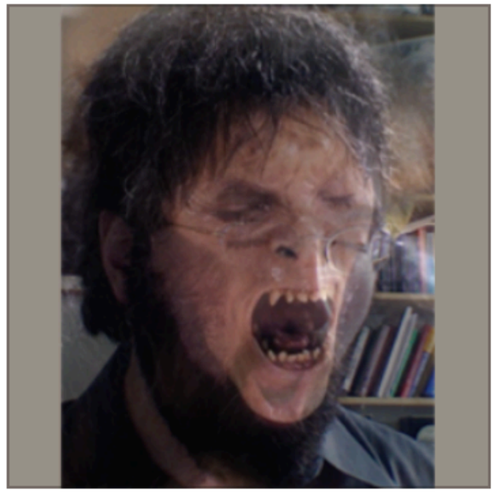


Demo: **Why a cross-dissolve ghosts, on a man-to-werewolf morph (interactive…**

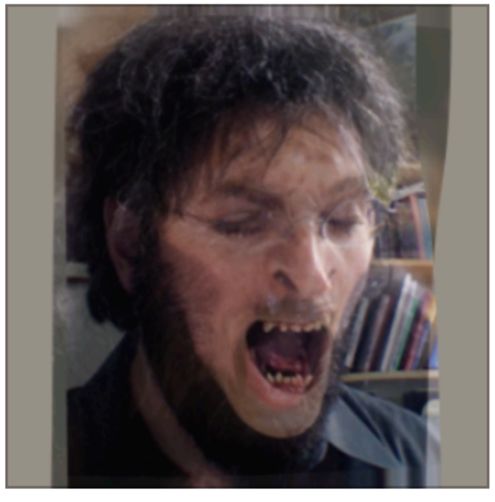
I_0



cross-dissolve, t = 0.50



morph, t = 0.50



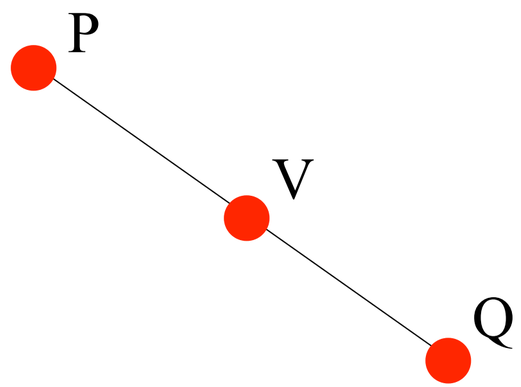
I_1



Beier and Neely field morph, 9 segment pairs; both middle panels use the same t

Averaging points (location)

- **P & Q are two 2D points (in the “domain”)**
- **V= (1-t) P + t Q**



Warping

- **Move pixel spatially: C’(x,y)=C(f(x,y))**
- **Leave colors unchanged**

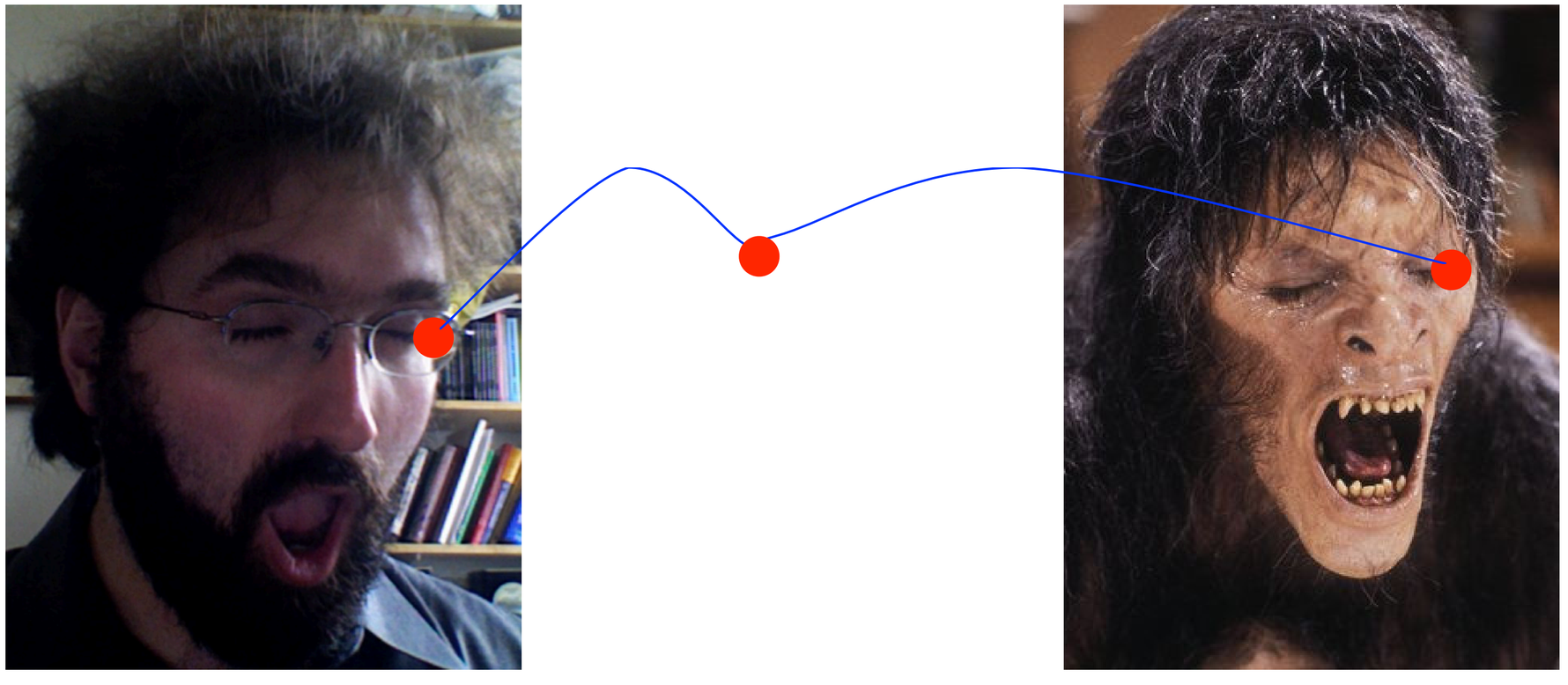


Recap & questions

- **Color (range) interpolation:**
  output(x, y) = (1-t) * im1(x, y) + t * im2(x, y)
- **Location (domain) interpolation: V= (1-t) P + t Q**
- **Warping: domain transform out(x,y)=im(f $^{-1}$(x,y))**

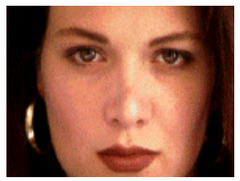
Morphing: combine both

- **For each pixel**
  - Transform its location like a vector (domain)
  - Then linearly interpolate colors (range)



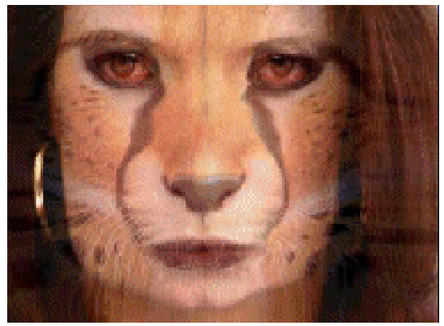
Morphing

- **Input: two images I_0 and I_{N+1}**

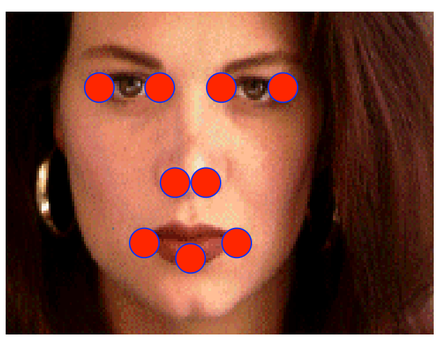


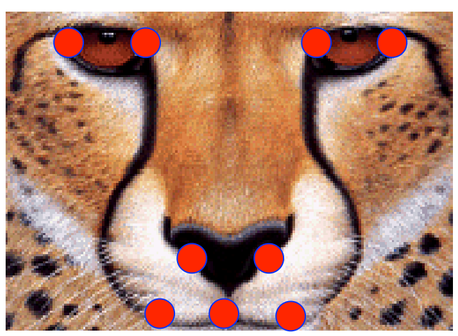


- **Expected output: image sequence I_i, with i∈ 1..N**



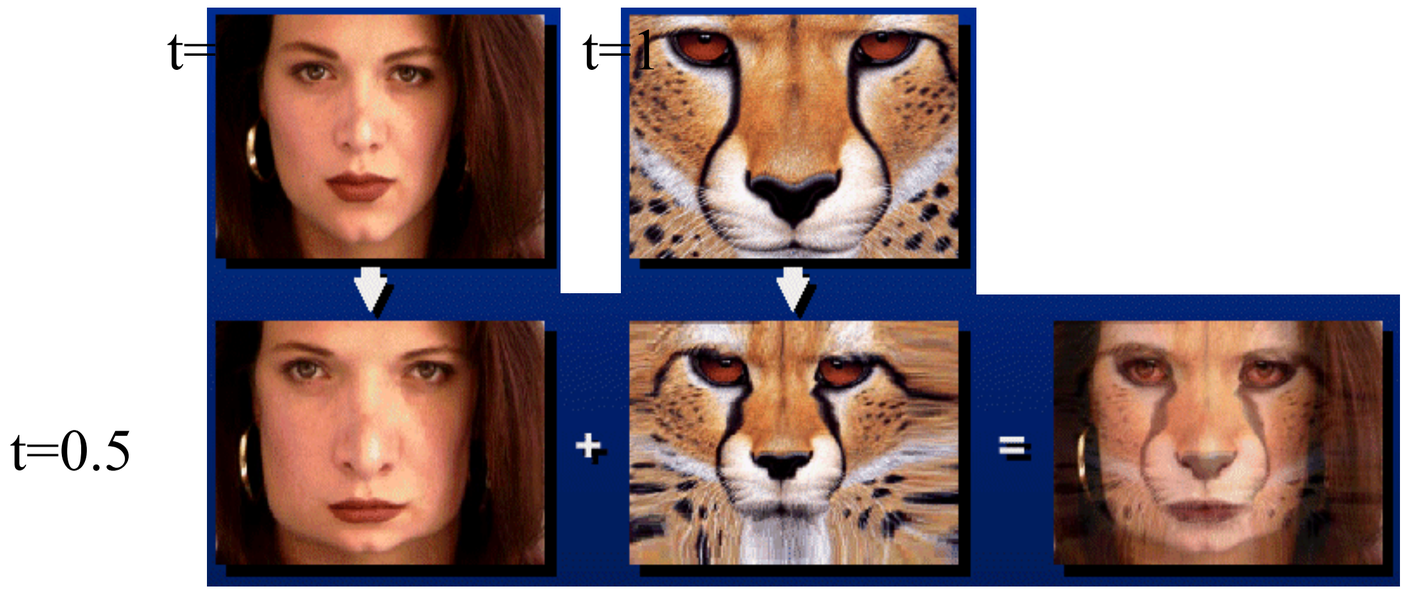
- **User specifies sparse correspondences on the images**





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



Demo: The three-step morph pipeline on the same man-to-werewolf pair, for…



t = 0.50: weights 0.50 and 0.50

Warping

How do we specify the warp?

- **Before, we saw simple transformations**
  - linear, affine, perspective



translatio
n



rotation



aspect



affine



perspective

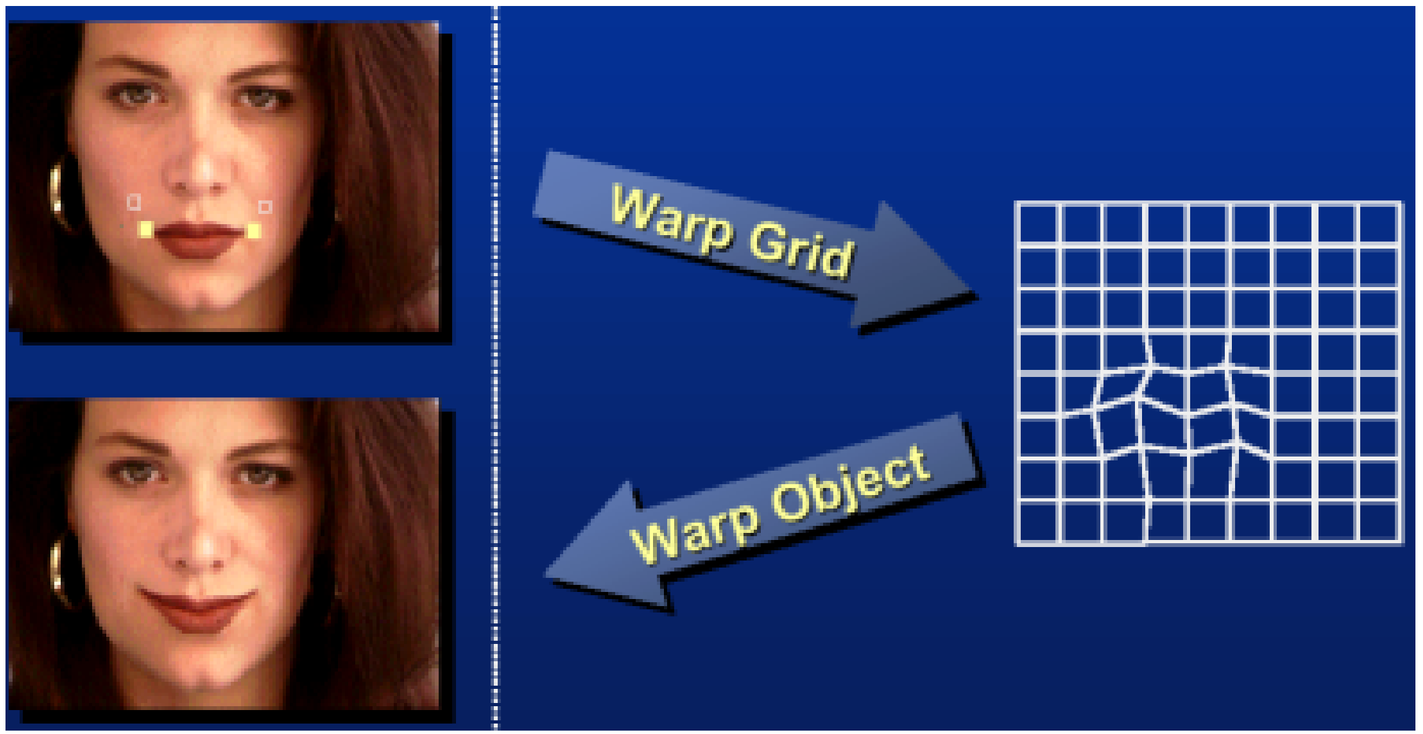


cylindrical

- **But we want more flexibility**

Image Warping – parametric

- **Move control points to specify a spline warp**
- **Spline produces a smooth vector field**



Slide Alyosha Efros

Warp specification - dense

- **How can we specify the warp?**

  Specify corresponding *spline control points*

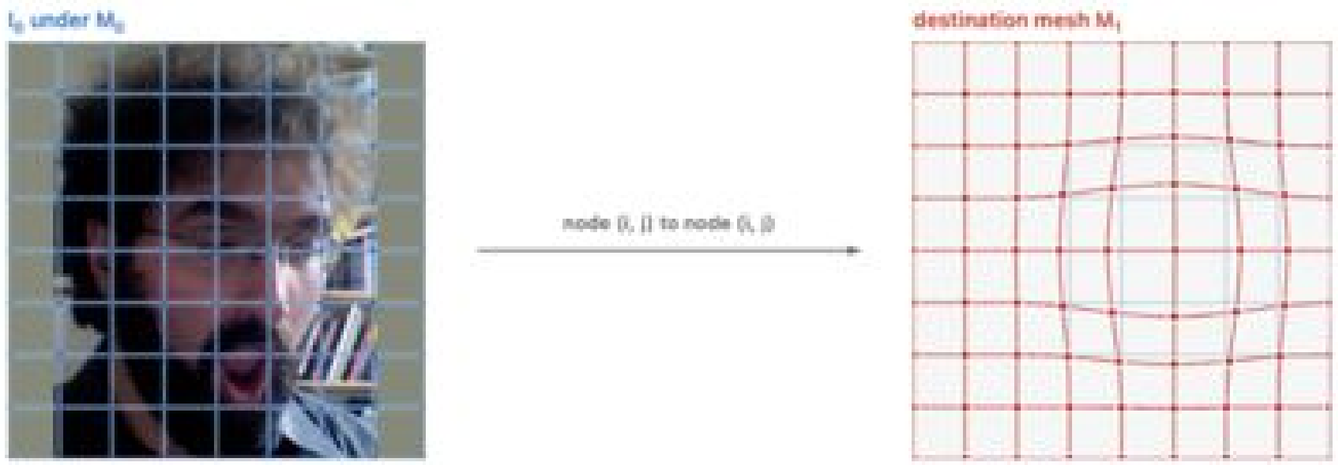
  - *interpolate* to a complete warping function



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

Slide Alyosha Efros

Demo: Two-pass mesh warp: M_0 and M_1

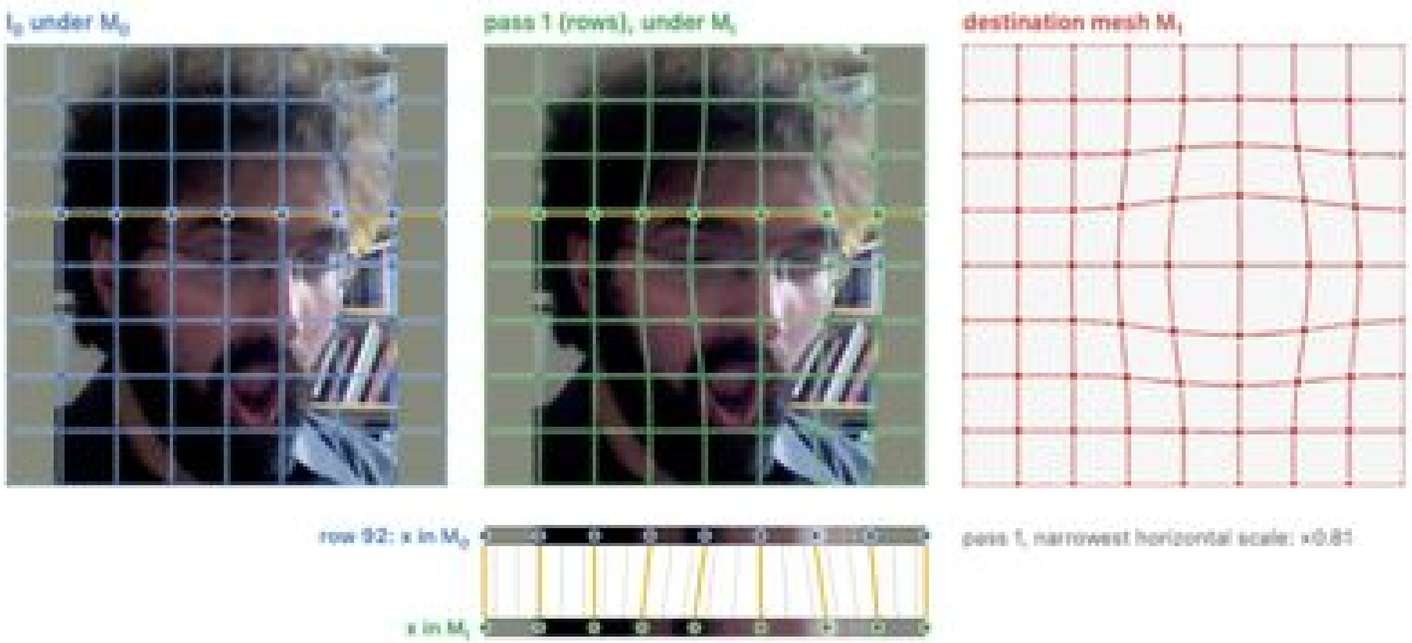


1 mesh warp: M_0 and M_1 | 2 mesh warp: pass 1, rows | 3 mesh warp: pass 2, columns | 4 against triangles and bilinear cells | 5 mesh morph: M(t)

grid cells 8 × 8 spline Catmull-Rom linear M₁ bulge grids reset

Demo courtesy of Frédo Durand, MIT.

Demo: Two-pass mesh warp: pass 1, rows



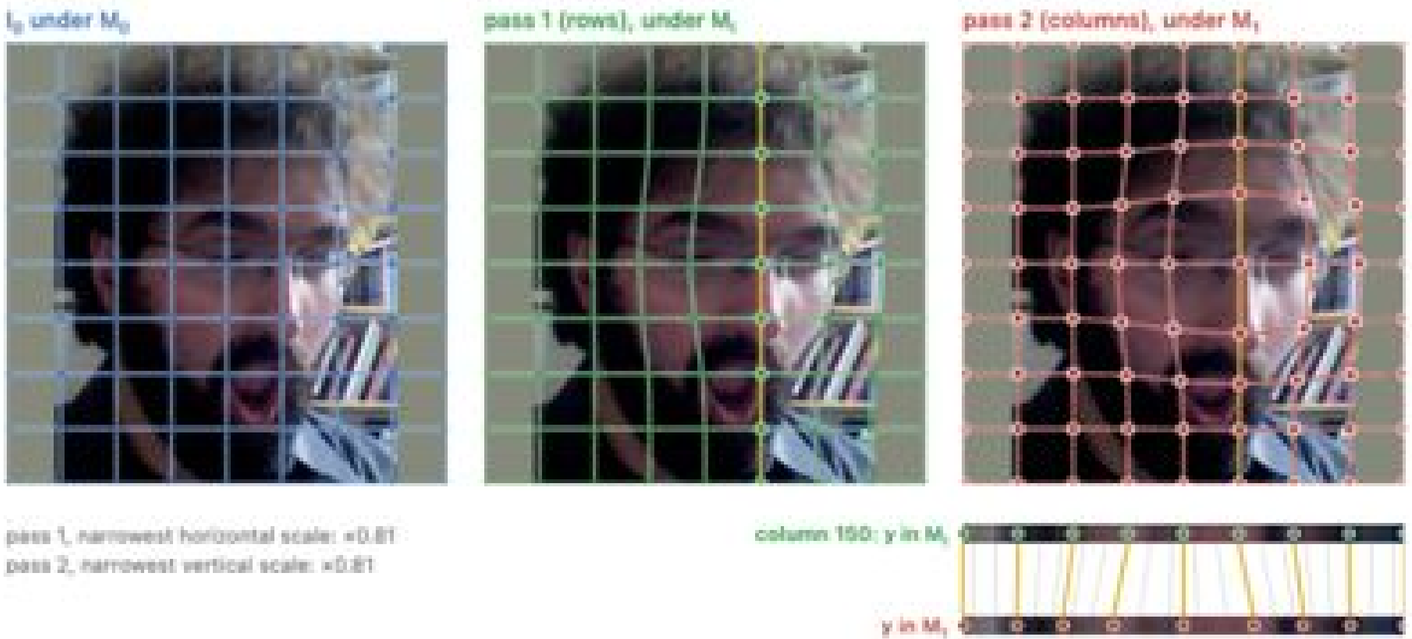
1 mesh warp: M_0 and M_1 | 2 mesh warp: pass 1, rows | 3 mesh warp: pass 2, columns | 4 against triangles and bilinear cells | 5 mesh morph: M(t)

grid cells 8 × 8 spline Catmull-Rom linear M_t bulge

row 92 grids reset

Demo courtesy of Frédo Durand, MIT.

Demo: Two-pass mesh warp: pass 2, columns



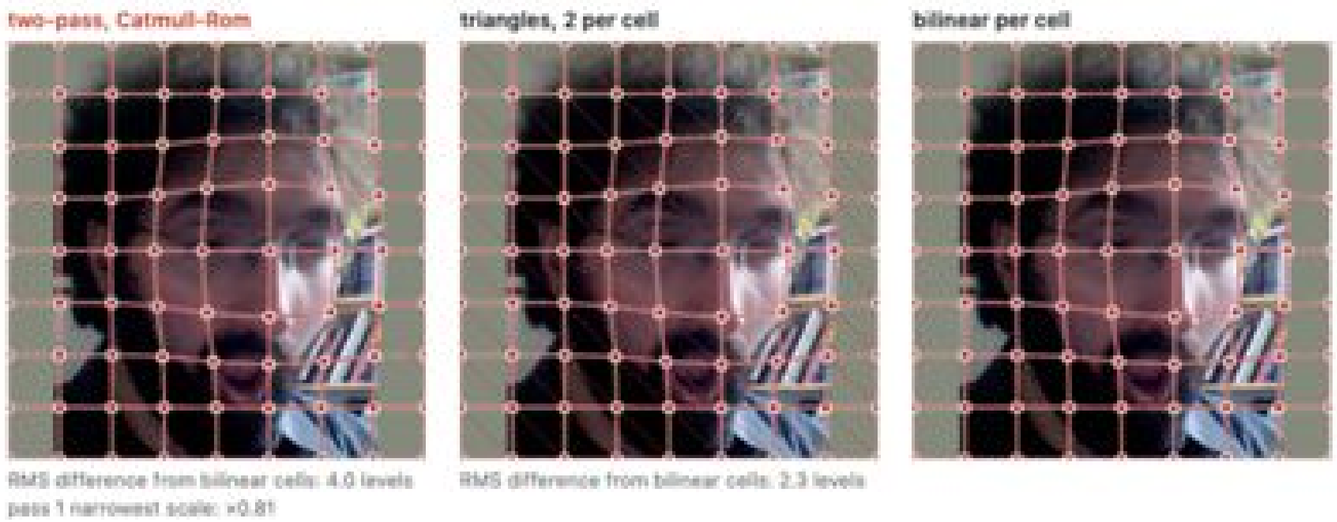
1 mesh warp: M_0 and M_1 | 2 mesh warp: pass 1, rows | 3 mesh warp: pass 2, columns | 4 against triangles and bilinear cells | 5 mesh morph: M(t)

grid cells 8 × 8 spline Catmull-Rom linear M_t bulge

column 150 grids reset

Demo courtesy of Frédo Durand, MIT.

Demo: Two-pass mesh warp: against triangles and bilinear cells



1 mesh warp: M_x and M_y | 2 mesh warp: pass 1, rows | 3 mesh warp: pass 2, columns | **4 against triangles and bilinear cells** | 5 mesh morph: M(t)

grid cells 8 × 8 spline **Catmull-Rom** linear M bulge grids reset

Demo courtesy of Frédo Durand, MIT.

Demo: Two-pass mesh warp: mesh morph: M(t)



1 mesh warp: M_0 and M_1 | 2 mesh warp: pass 1, rows | 3 mesh warp: pass 2, columns | 4 against triangles and bilinear cells | 5 mesh morph: M(t)

grid cells 8 × 8 spline Catmull-Rom linear t 0.50 grids

reset

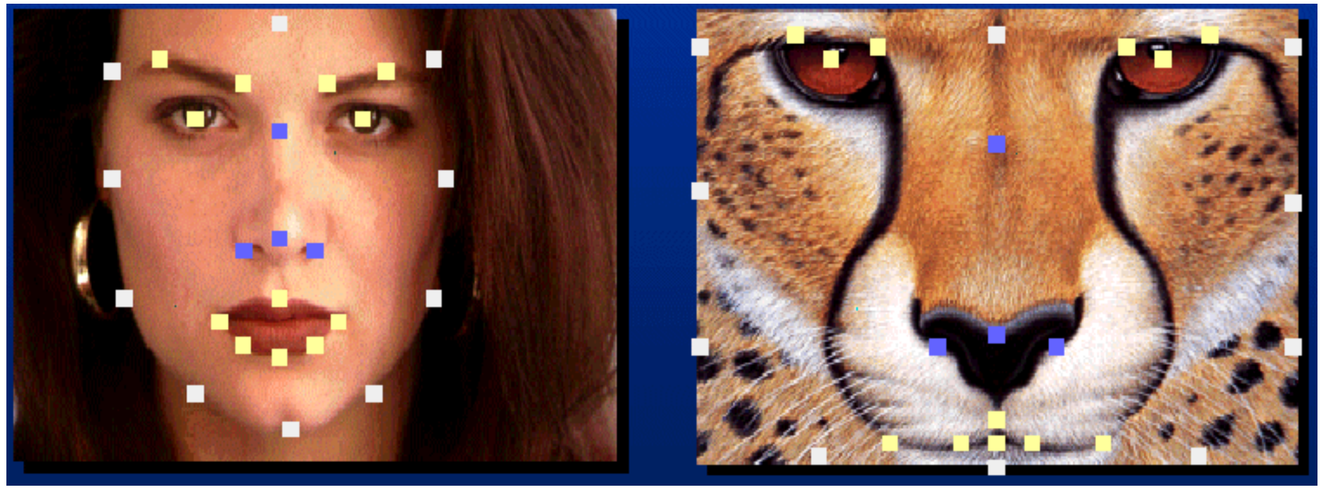
Demo courtesy of Frédo Durand, MIT.

Warp specification - sparse

- **How can we specify the warp?**

  Specify corresponding *points*

  - *interpolate* to a complete warping function
  - How do we do it?



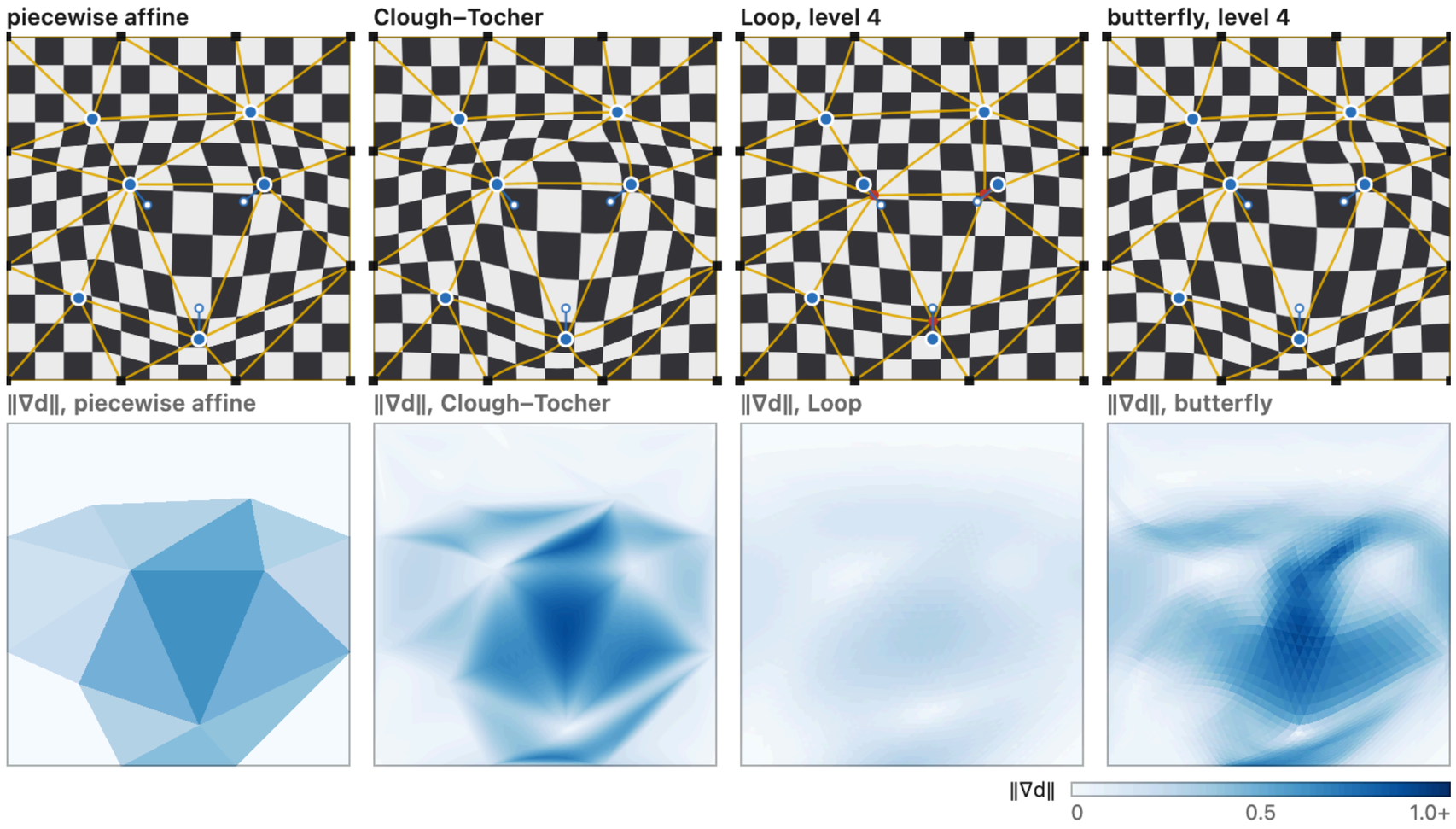
How do we go from feature points to pixels?

Slide Alyosha Efros

Demo: Mesh warp

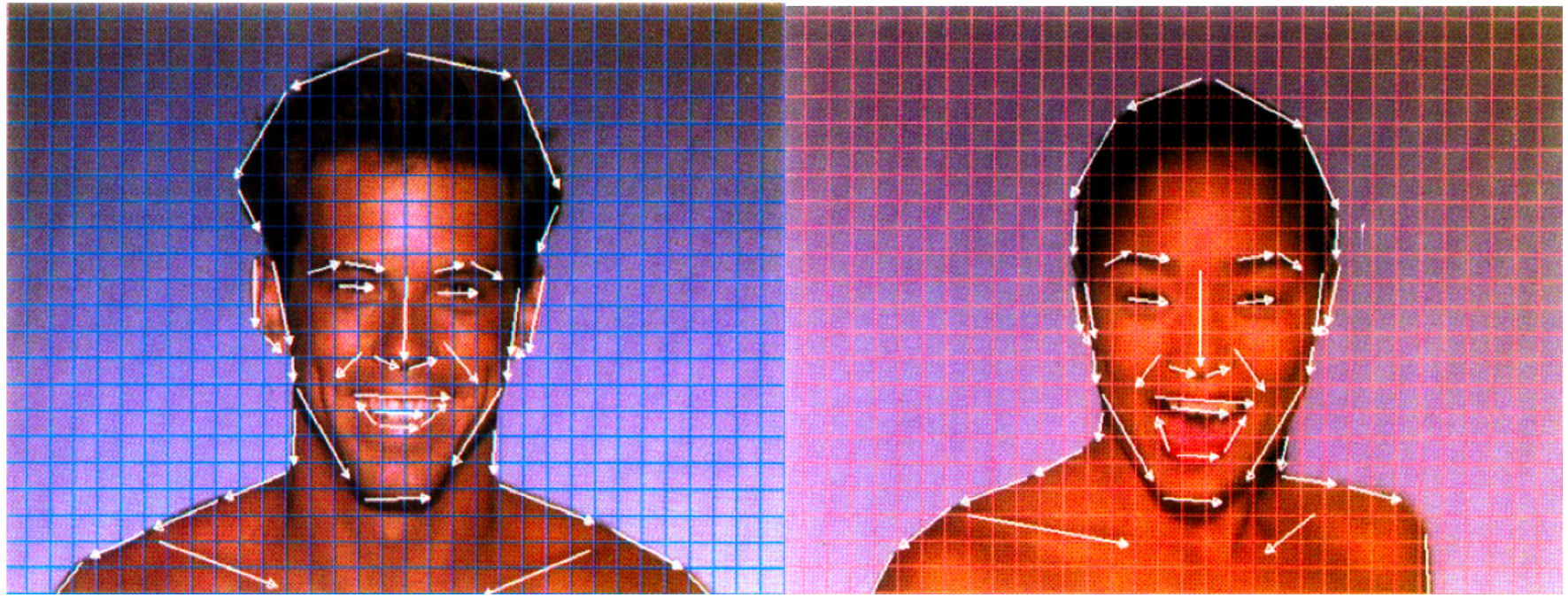


Demo: Smooth triangulation warps



Beier and Neely

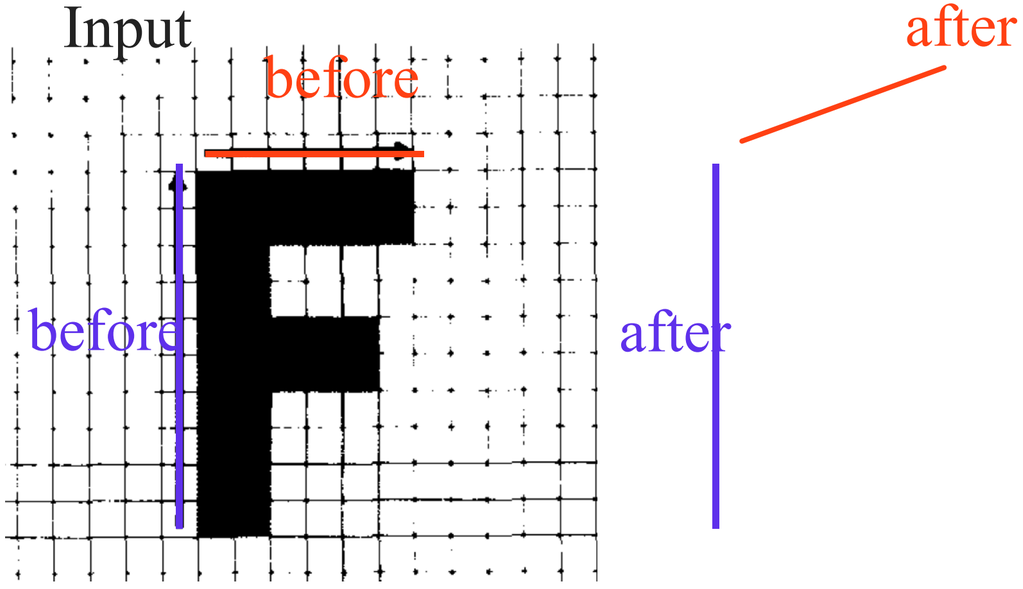
- **Specify warp based on pairs of segments**
  - **http://dl.acm.org/citation.cfm?id=134003**
  - Feature-Based Metamorphosis, SIGGRAPH 1992
  - Used in Michael Jackson' Black and White Music Video
  - Pset 5!!

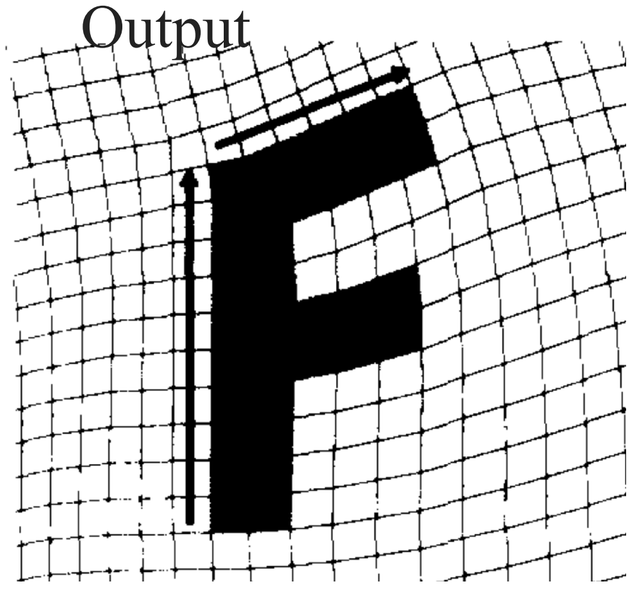


Segment-based warping

Problem statement

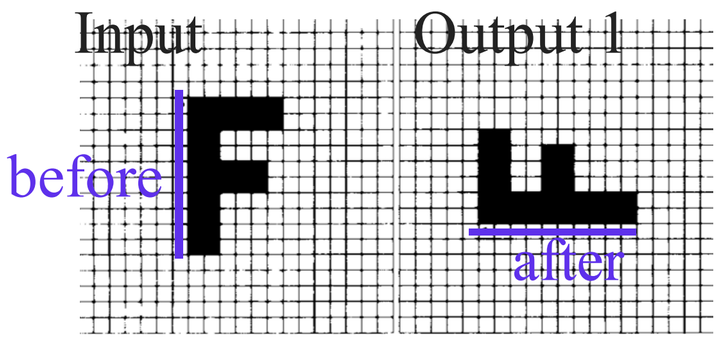
- **Inputs: One image, two lists of segments before and after, in the image domain**
- **Goal: warp the image "following" the displacement of the segments**





Idea

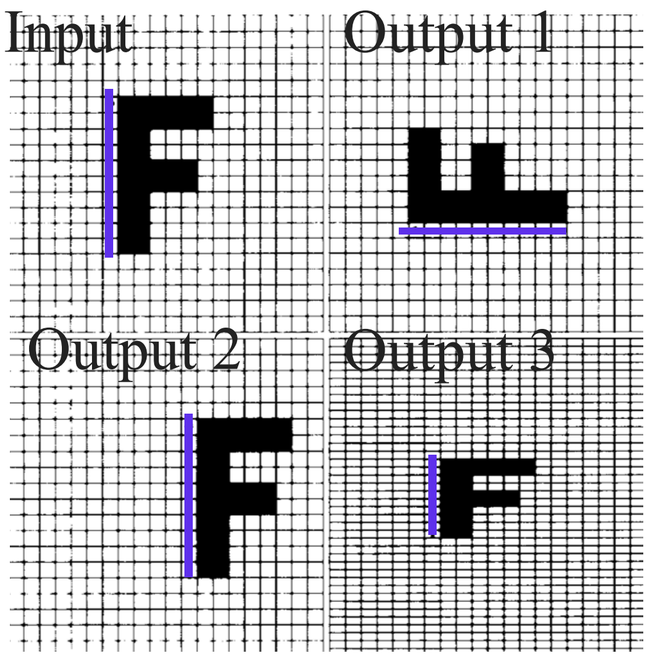
- **Each before/after pair of segment implies a planar transformation**
  - simple and affine



Single line transforms

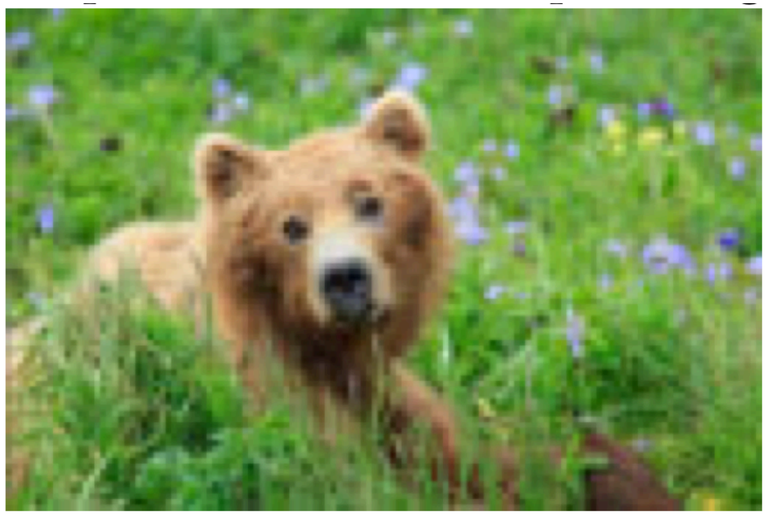
Idea

- **Each before/after pair of segment implies a planar transformation**
  - simple and affine

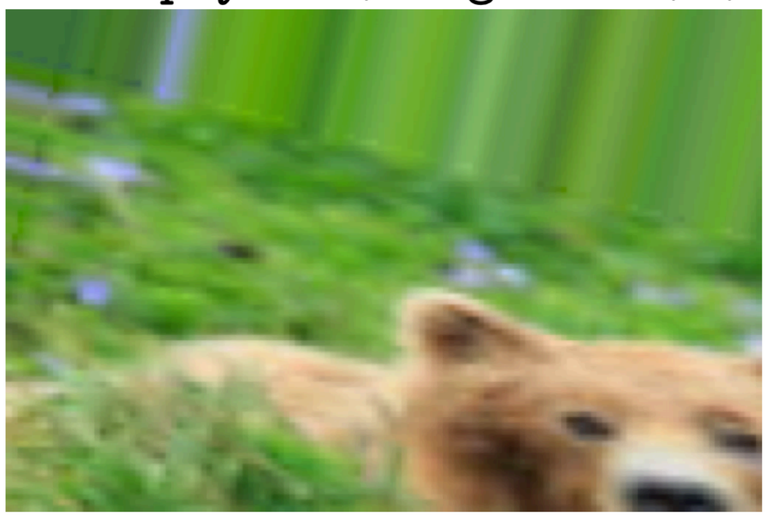


Single line transforms

Test

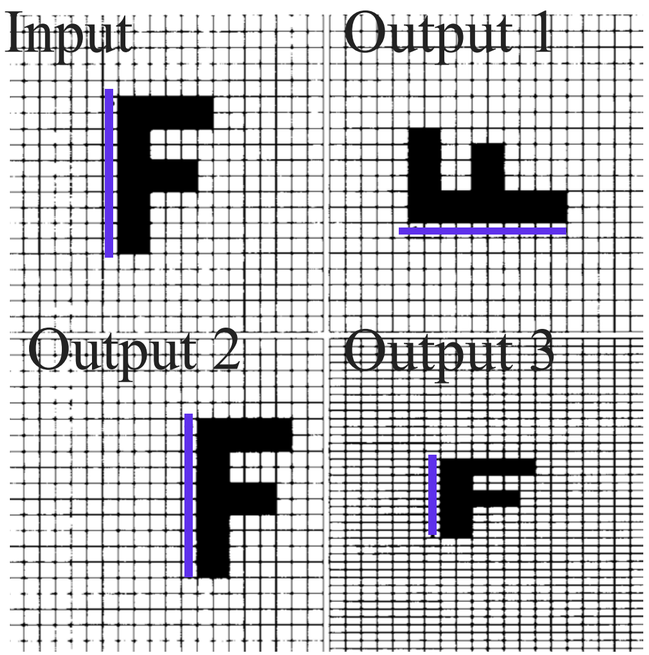


→ `warpBy1(im, segment(0,0, 10,0), segment(10, 10, 30, 15))` →

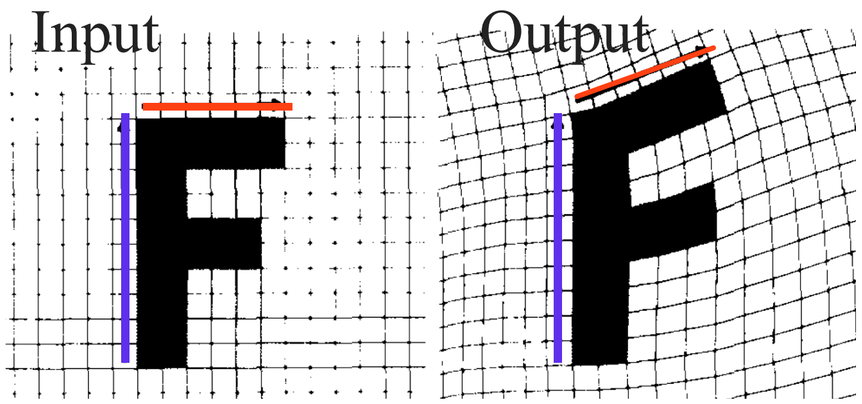


Idea

- **Each before/after pair of segment implies a planar transformation**
- **Then take weighted average of transformations**



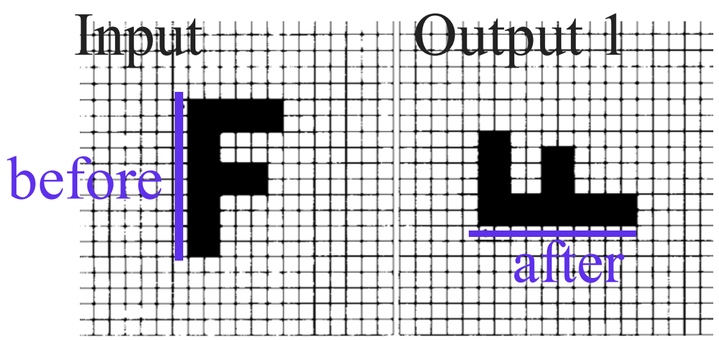
Single line transforms



Transform wrt 2 lines

Back to single segment

- **Each before/after pair of segment implies a planar transformation**
  - simple and affine

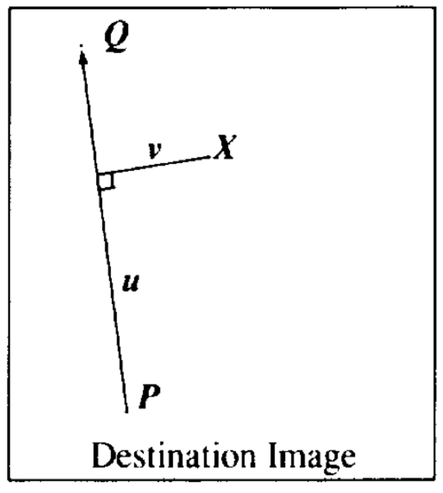


Single line transforms

Linear algebra refresher

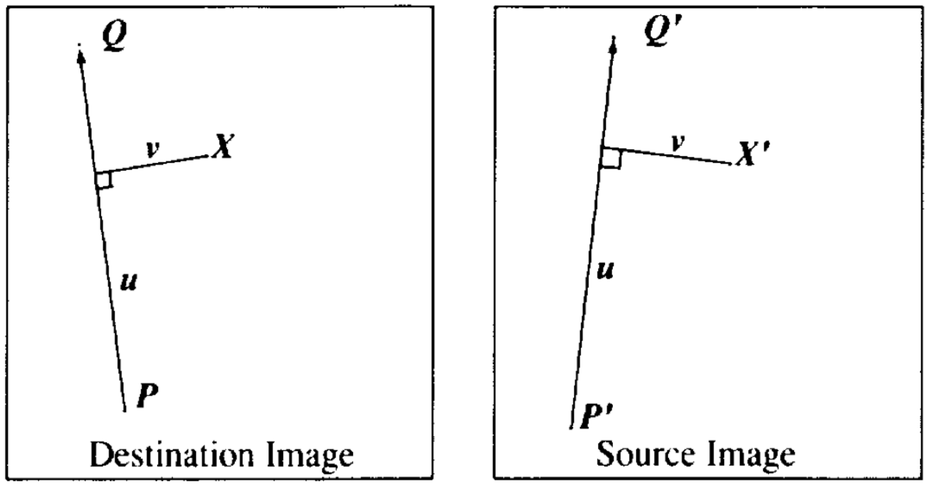
Transform wrt 1 segment

- **Define a coordinate system with respect to segment**
  - 1 dimension, *u*, along segment
  - 1 dimension, *v*, orthogonal to segment

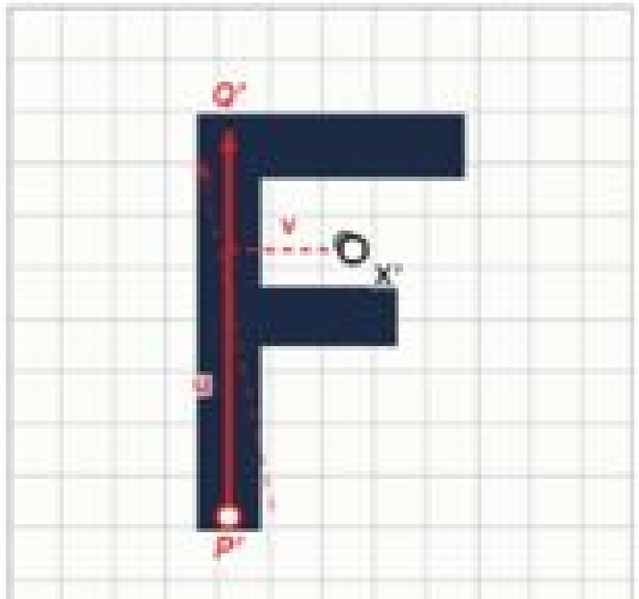


Transform wrt 1 segment

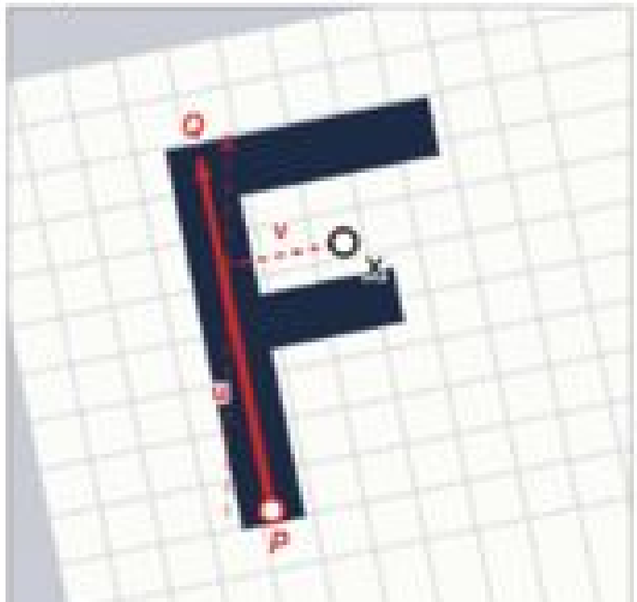
- **Define a coordinate system with respect to segment**
  - 1 dimension, *u*, along segment
  - 1 dimension, *v*, orthogonal to segment
- **Compute u, v in one image**
  - The after one, because we use the inverse transform
- **Compute point corresponding to u, v in second image**



Input: source **segments P'Q'**



output: destination segments PQ



X = (168, 118) in the output; in each segment's frame:
segment 1: u = 0.69, v = 62.1 px → X'1 = (172, 122)
X' = P' + u (Q' − P') + v ⊥(Q' − P') / ‖Q' − P'‖ = (172, 122)

◀ | 1 one segment | 2 two segments | 3 the weight field | ▶

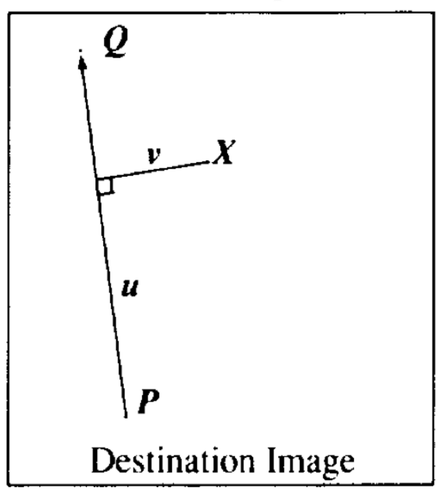
input | the letter F, as in the paper | upload a photograph | ☐ scale v like u

reset

Demo courtesy of Frédo Durand, MIT.

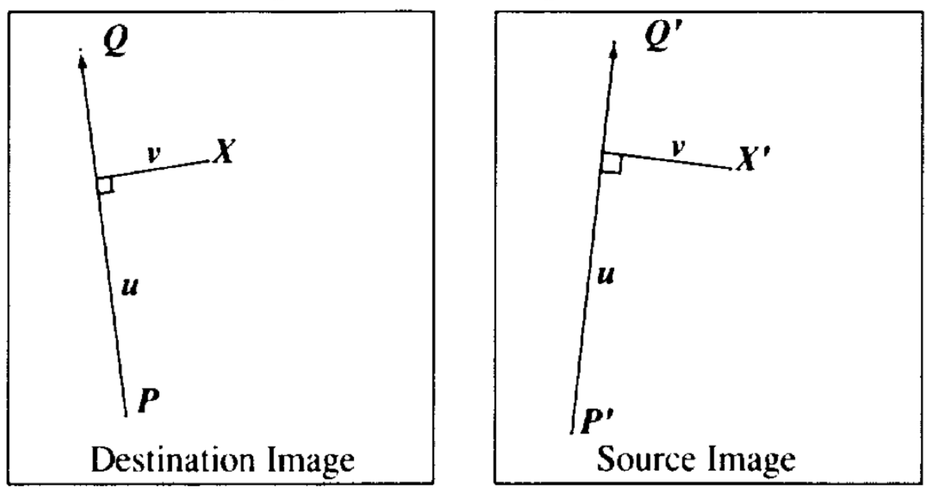
Computing u, v

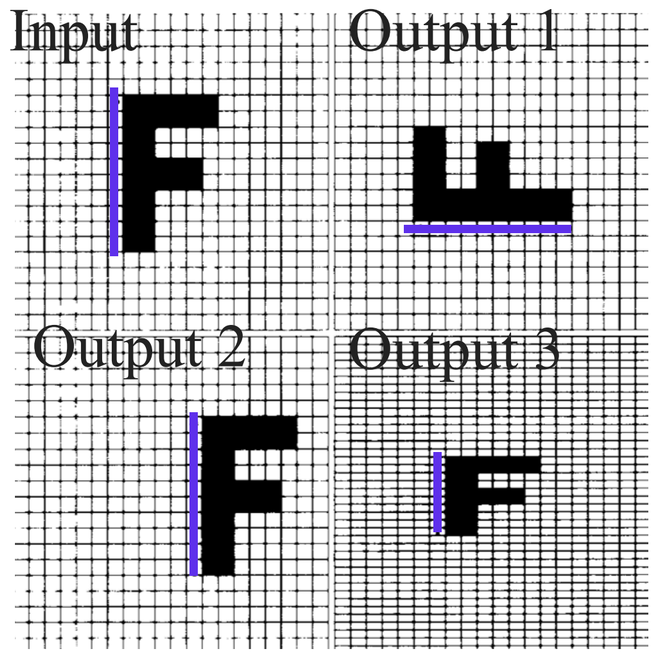
- **u = PX.PQ/||PQ||²**
  - this way u is 0 at P and 1 at Q
- **v=PX.perpendicular(PQ)/||PQ||**
  - where perpendicular(PQ) is PQ rotated by 90 degrees, and has length ||PQ||
  - unlike u which is normalized, v is in distance units



Transforming a point given u, v

- **X'=P'+uP'Q+v 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





Single line transforms

Questions?

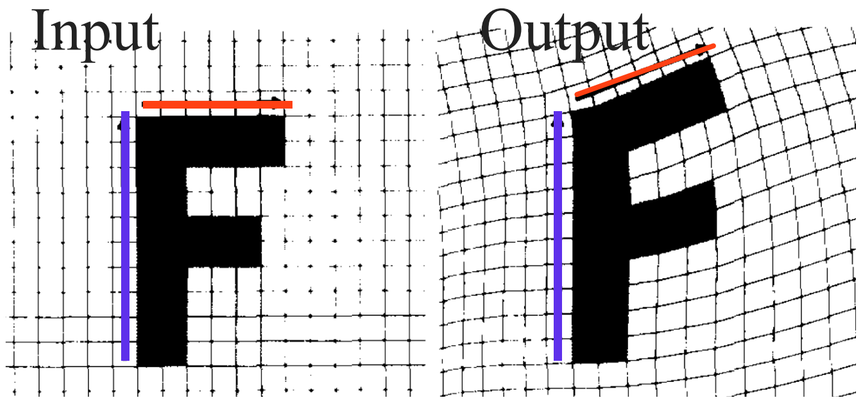


Multiple segments

- **For each point X**
  - For each segment pair s_before[i], s_after[i]
    - Transform X into X'i according to this pair
  - Compute weighted average of all transformed X'i
    - weight according to distance to segments

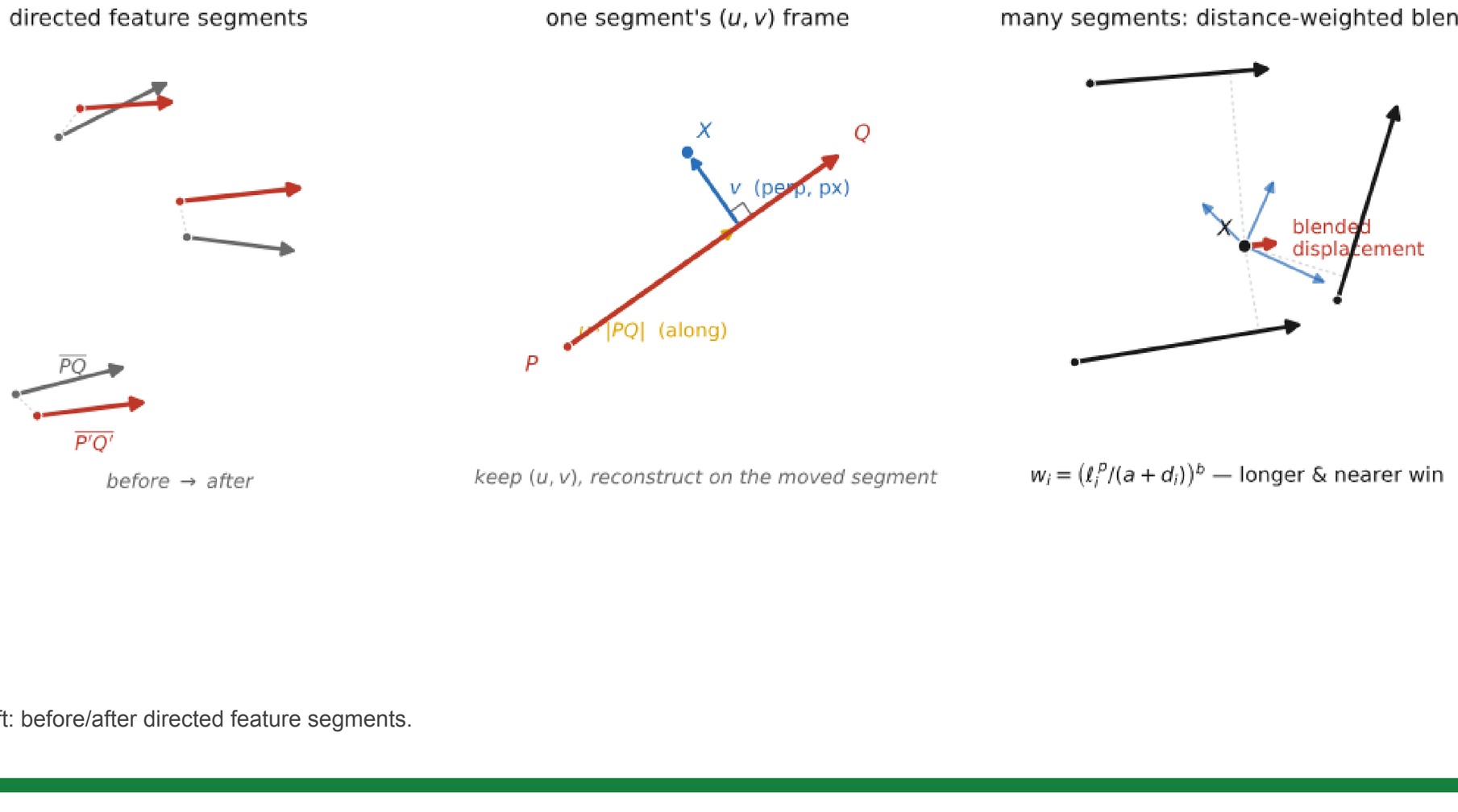
$$weight = \left(\frac{length^p}{a + dist} \right)^b$$

where a, b, p
control the influence



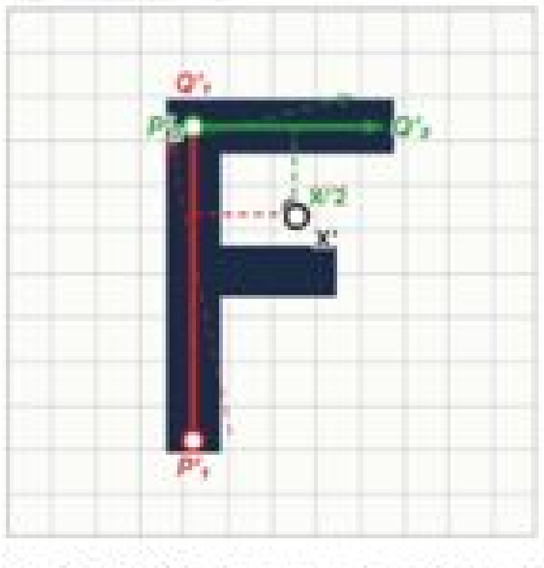
Transform wrt 2 lines

Figure 11.1.9: Beier–Neely feature-line warp

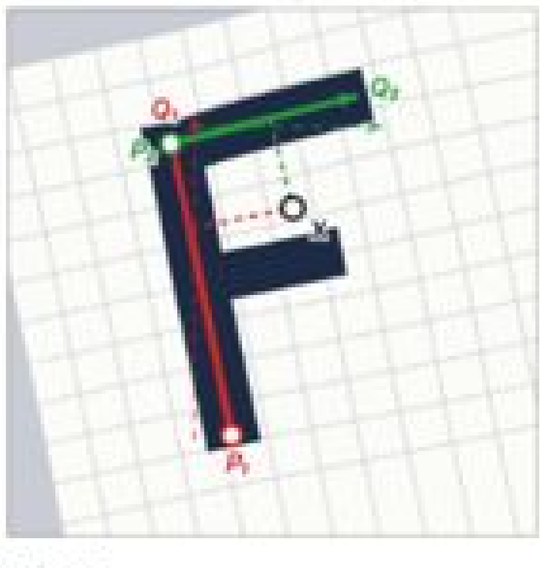


Left: before/after directed feature segments.

input: source **segments P'Q'**



output: destination segments PQ



X = (168, 118) in the output; in each segment's frame:
segment 1: u = 0.69, v = 62.1 px → X'1 = (172, 122); d = 62.1 px, ℓ = 180 px, w = 0.045, 55% of the total
segment 2: u = 0.53, v = 54.6 px → X'2 = (170, 125); d = 54.6 px, ℓ = 114 px, w = 0.037, 45% of the total
X' = Σ w_i X'_i / Σ w_i = (171, 123), $w = (\ell^p / (a + d))^b$

◀ 1 one segment | 2 two segments | 3 the weight field ▶

● segment 1 ● segment 2 input the letter F, as in the paper upload a photograph

☐ scale v like u

a 1.0 b 2.0 p 0.50 reset

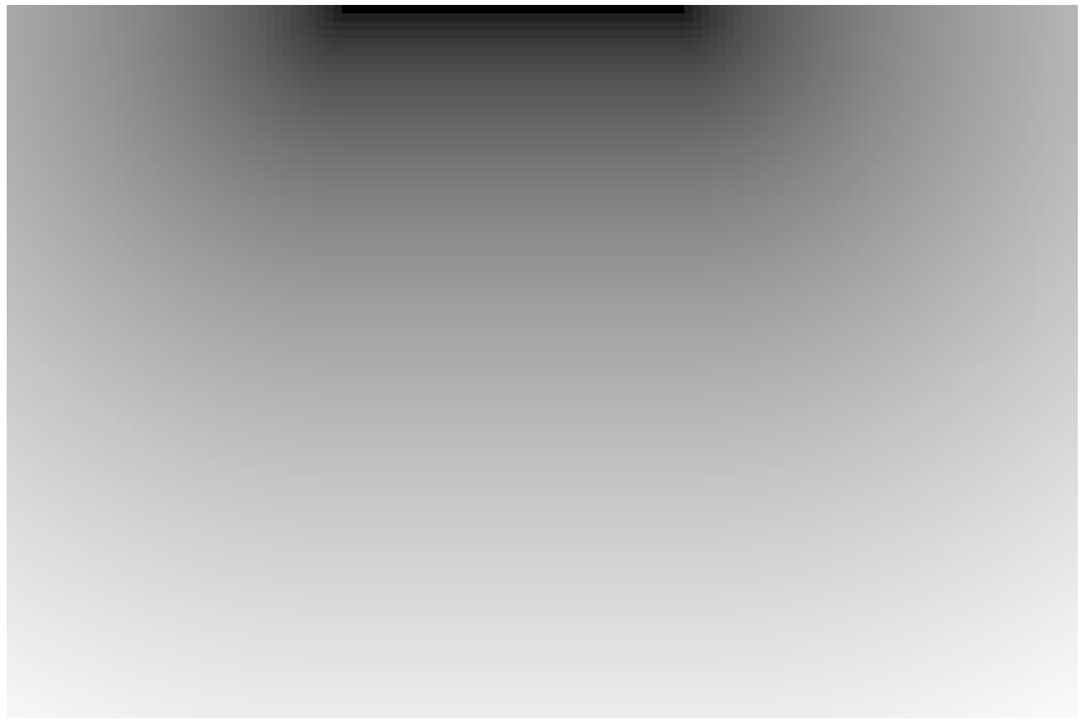
Demo courtesy of Frédo Durand, MIT.

Distance to a segment

- **Multiple cases...**
  - dot product, test >0, <1

Debugging: example

- **Debugging my distance function**



Questions?

- **More debugging**



The whole warp, one output pixel at a time

```
for each output pixel X = (x, y):
    sum = (0, 0);    wsum = 0
    for each segment pair i:          P_iQ_i in the output, P'_iQ'_i in the source
        u = (X − P_i)·(Q_i − P_i) / ‖Q_i − P_i‖²
        v = (X − P_i)·⊥(Q_i − P_i) / ‖Q_i − P_i‖
        X'_i = P'_i + u (Q'_i − P'_i) + v ⊥(Q'_i − P'_i) / ‖Q'_i − P'_i‖
        d = |v| if 0 ≤ u ≤ 1, else the distance to the nearer endpoint
        w = (length_i^p / (a + d))^b
        sum += w (X'_i − X);    wsum += w
    X' = X + sum / wsum
    out(x, y) = source(X')          bilinear: X' falls between pixels
```

Inverse warping: the loop runs over the output and looks each pixel up in the source, so no output pixel is left empty. With one segment pair, w drops out and this is the single-segment transform.

Demo: Beier–Neely warp

Beier–Neely field warp

A single-image **warp** (no second image, no morph, no blend): the reader draws pairs of directed **feature line segments** — a **source** segment (blue, where a feature *is*) and a **destination** segment (orange, where it should move *to*). The picture is **inverse-warped** by the dense Beier–Neely field so that, at each destination segment, it shows the warped-in source content. Drag the **orange** ends to deform the face; drag the **blue** ends to retarget a segment. With every destination on top of its source the warp is the **identity** (Beier & Neely, SIGGRAPH 1992).

+ add feature | − delete selected | ≡ snap to identity | ↺ reset

a 1.0 b 2.0 p 0.50 ☐ original

WARPED RESULT — DRAG THE HANDLES



Click near an endpoint to grab it; drag to move. Each ▣ blue source segment is paired with the ▣ orange destination

Morphing

Input images





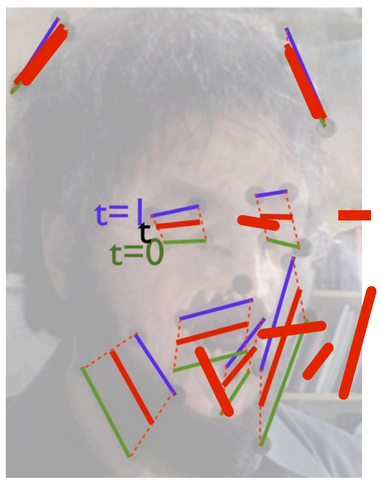
Segments





Interpolate segments

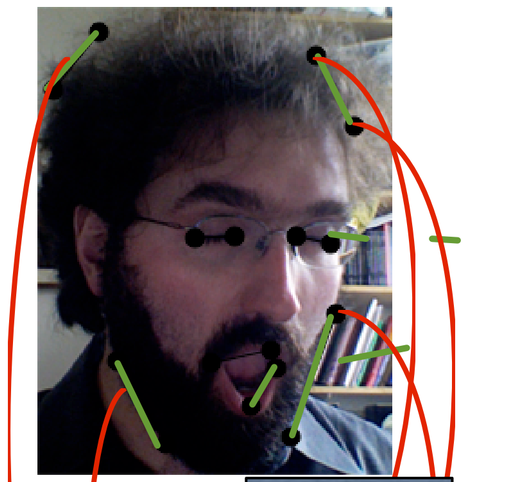


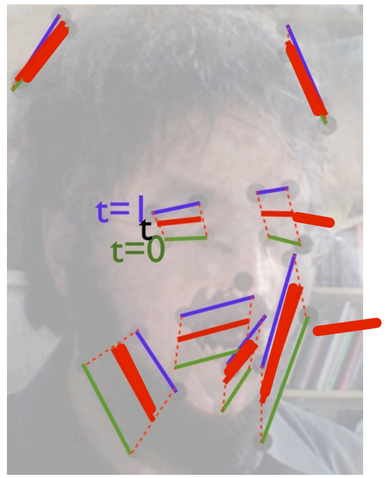




t=0.5

Warp images to segments[t]



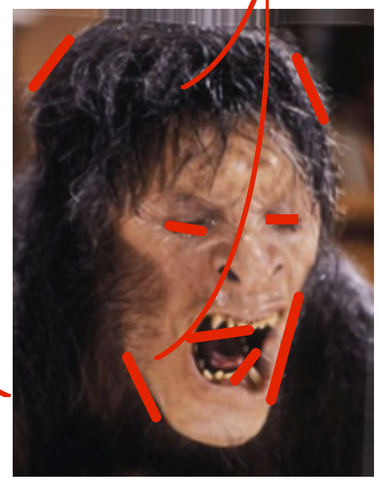




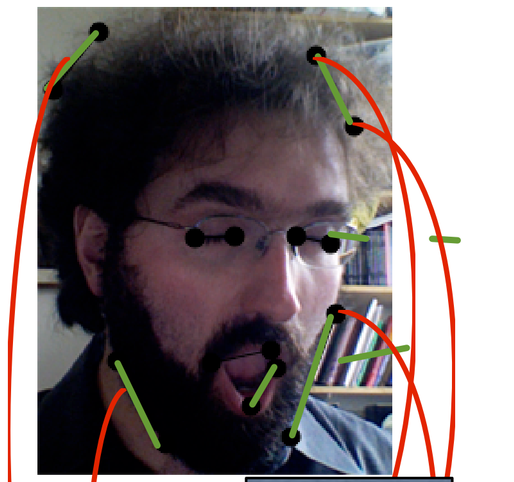


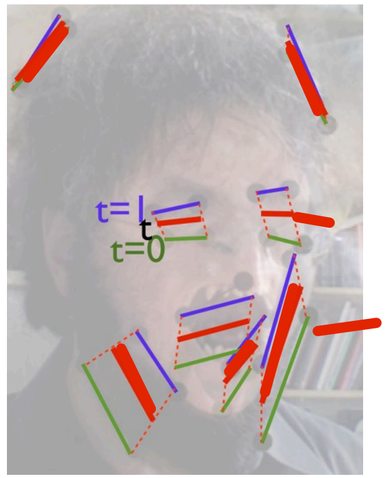
The red segments are at the same location in both images

Image features such as eyes are aligned



Warp images to segments[t]



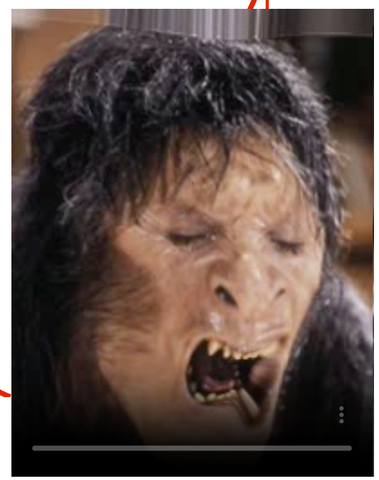






The red segments are at the same location in both images

Image features such as eyes are aligned

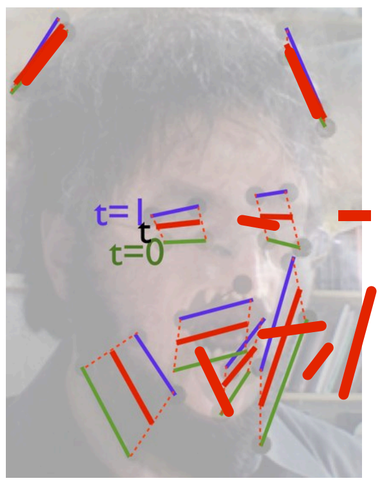


Interpolate color



Interpolate color

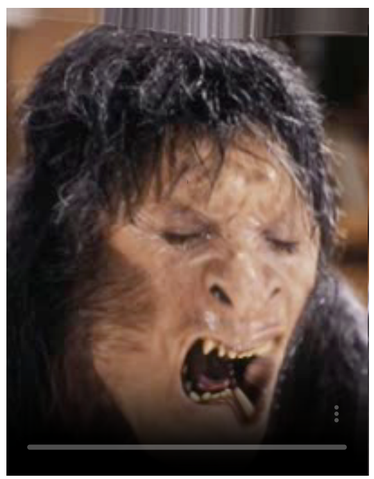








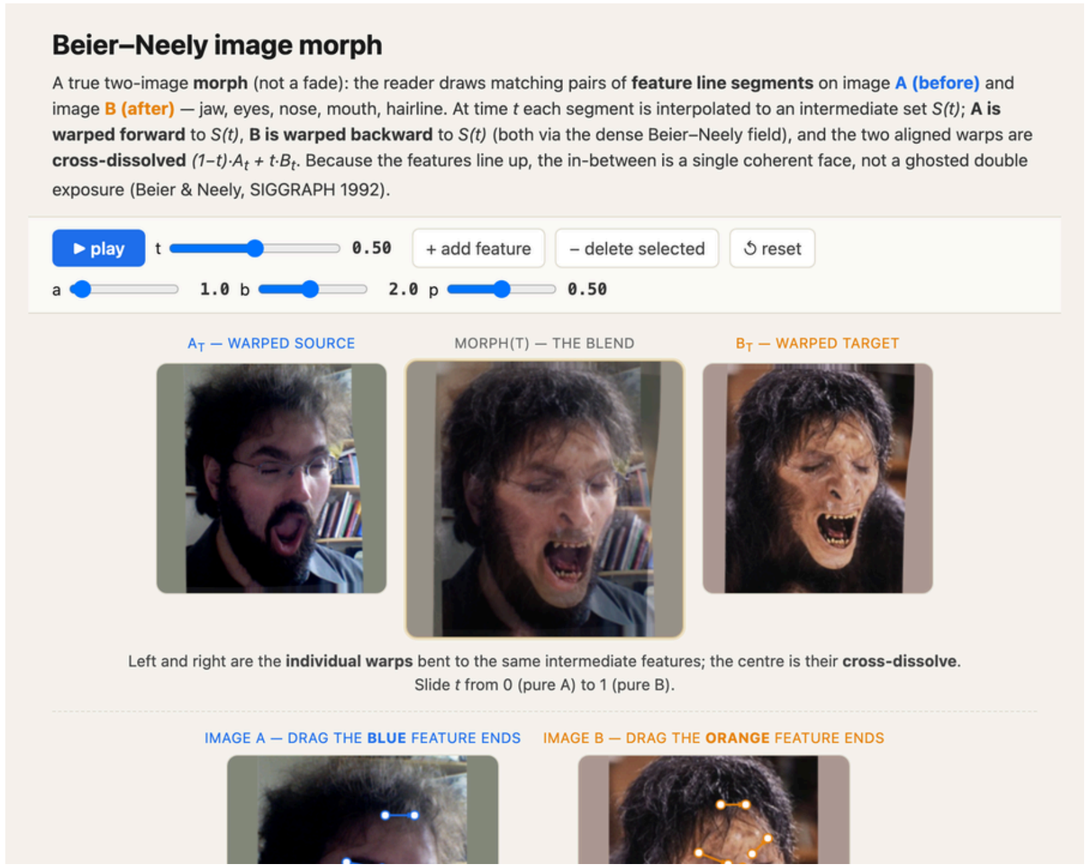




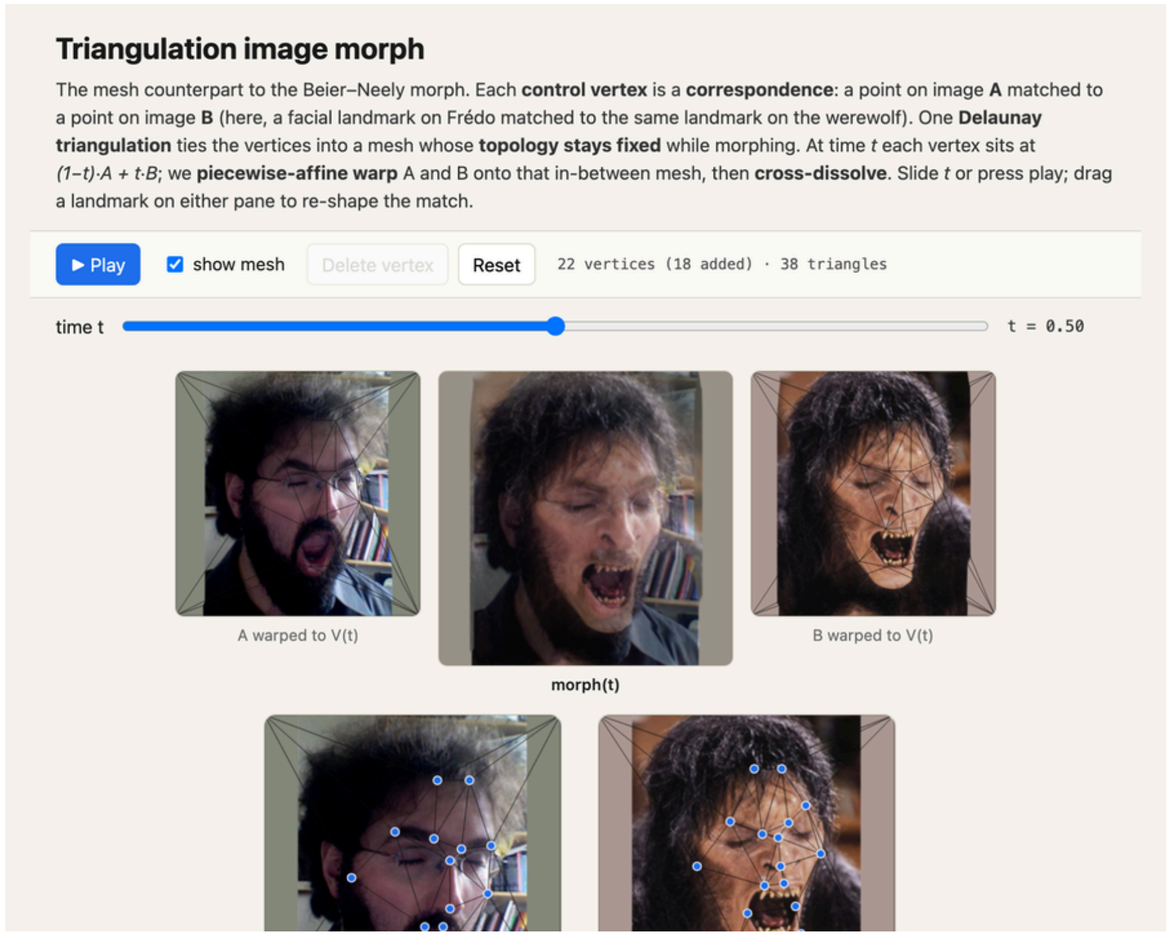
Result



Demo: Beier–Neely morph



Demo: Mesh morph



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



Separating the art and the artist

- Michael Jackson has been credibly accused of sexual abuse of children
- See for example the documentary “Leaving Neverland” https://en.wikipedia.org/wiki/Leaving_Neverland

Conversation about sexual assault

- Sexual assault is bad
- Consent is paramount
- Children are not able to consent
- See something, say something
- Victims’ stories are complicated. It doesn’t make it any less bad.
- There can be love or feelings involved, yet sexual assault is still bad
- It can take a long time to talk. It doesn’t make it less true.
- Men are more likely to be victims than falsely accused
- https://www.rainn.org/
- (617) 253-2300

Willow

Willow

- 1988, special effects by ILM
- First use of morphing



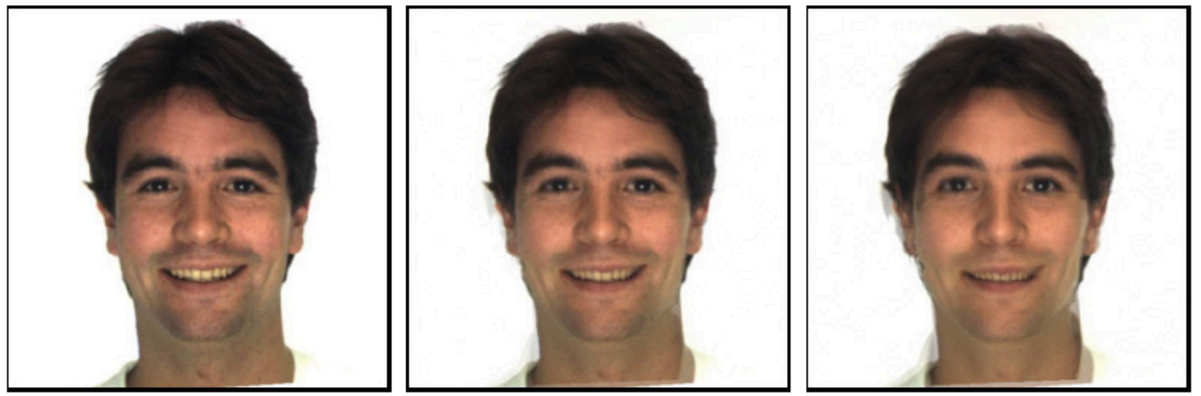
Featurette about morphing in Willow



Bells and whistles

Morphing & matting

- **Extract foreground first to avoid artifacts in the background**



(c) $\alpha = 0.0$ (d) $\alpha = 0.2$ (e) $\alpha = 0.4$



(f) $\alpha = 0.6$ (g) $\alpha = 0.8$ (h) $\alpha = 1.0$

Uniform morphing

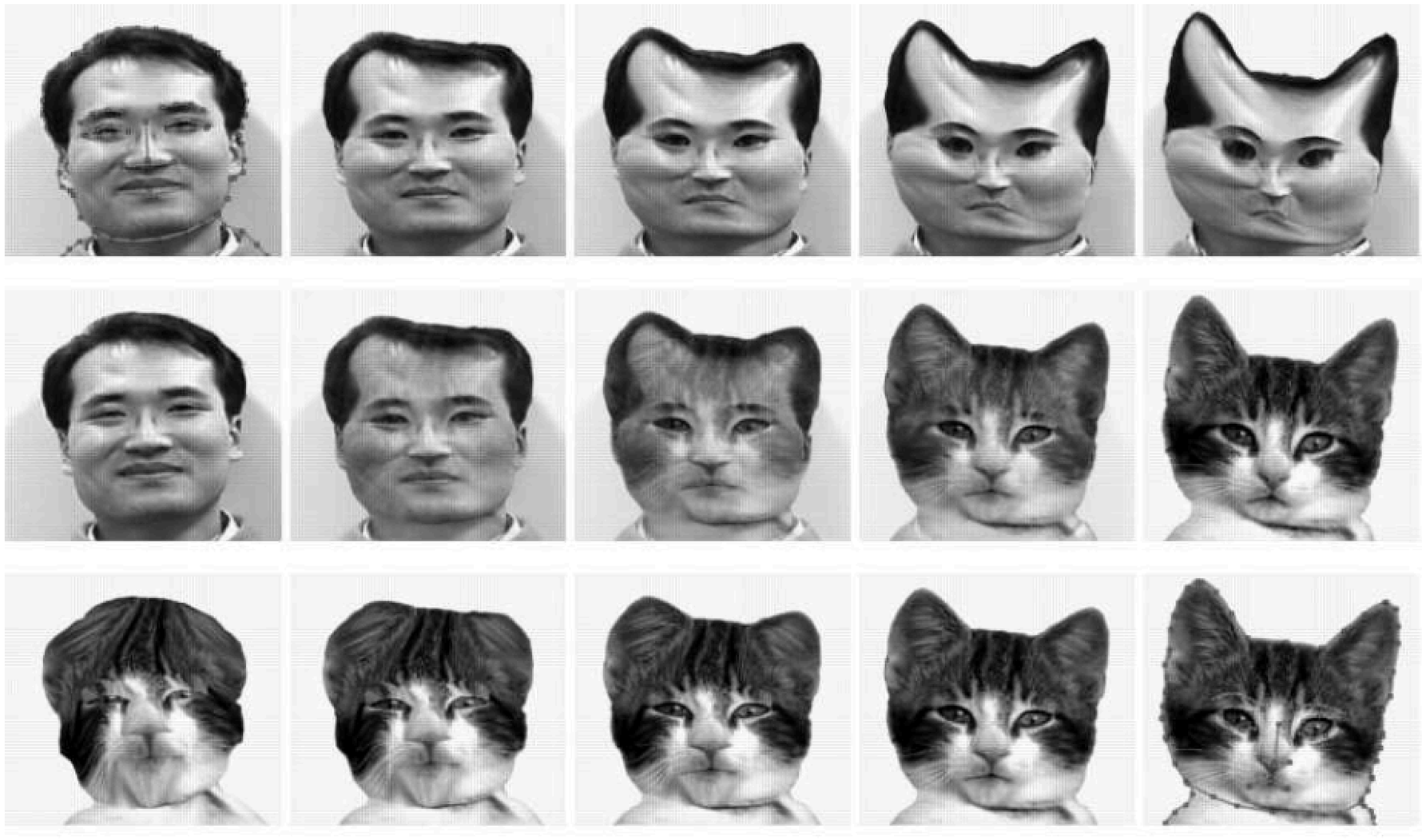


Figure 4. Uniform metamorphosis

Non-uniform morphing

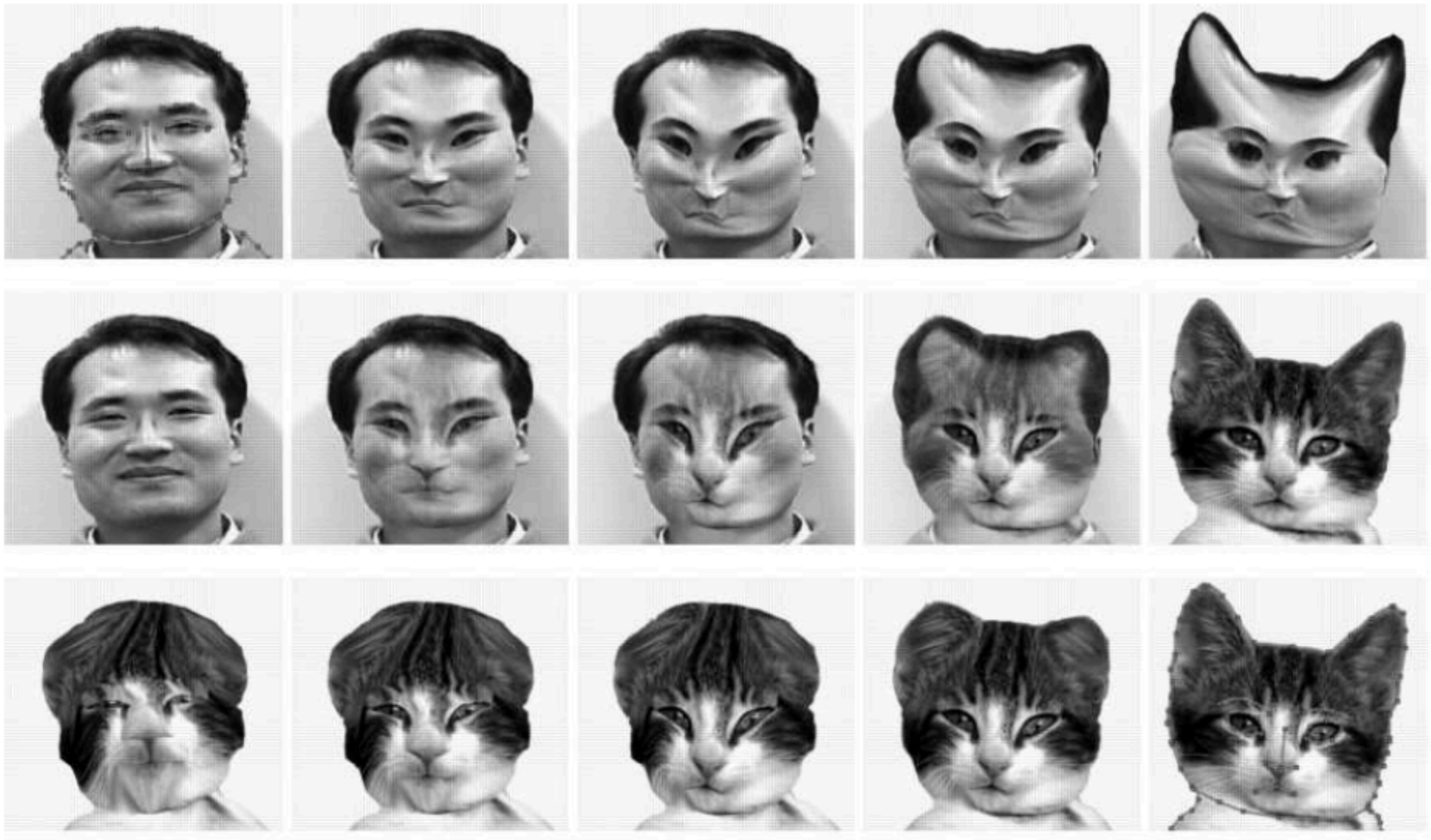
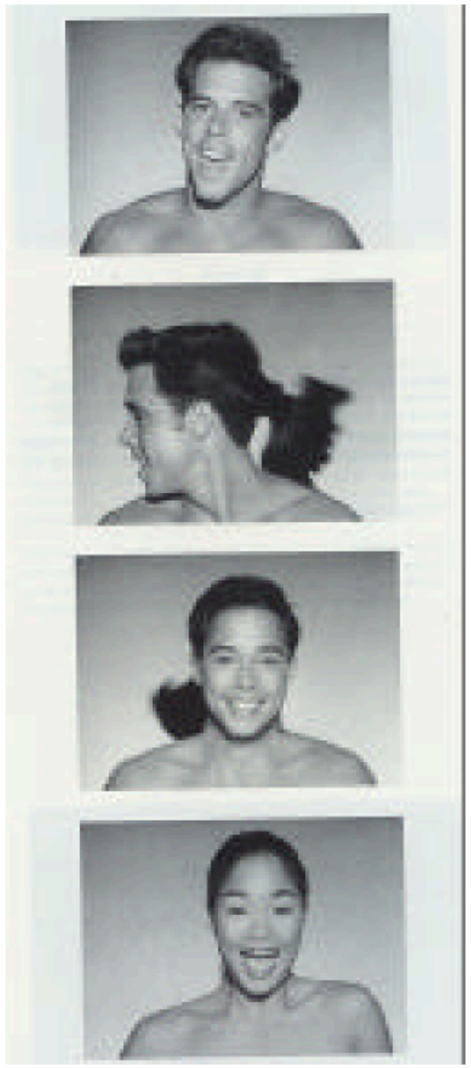


Figure 5. Nonuniform metamorphosis

Video

- **Lots of manual work**



Video: View Morphing, interpolating between two photographs along a physically valid…



Recap & Significance

Recap

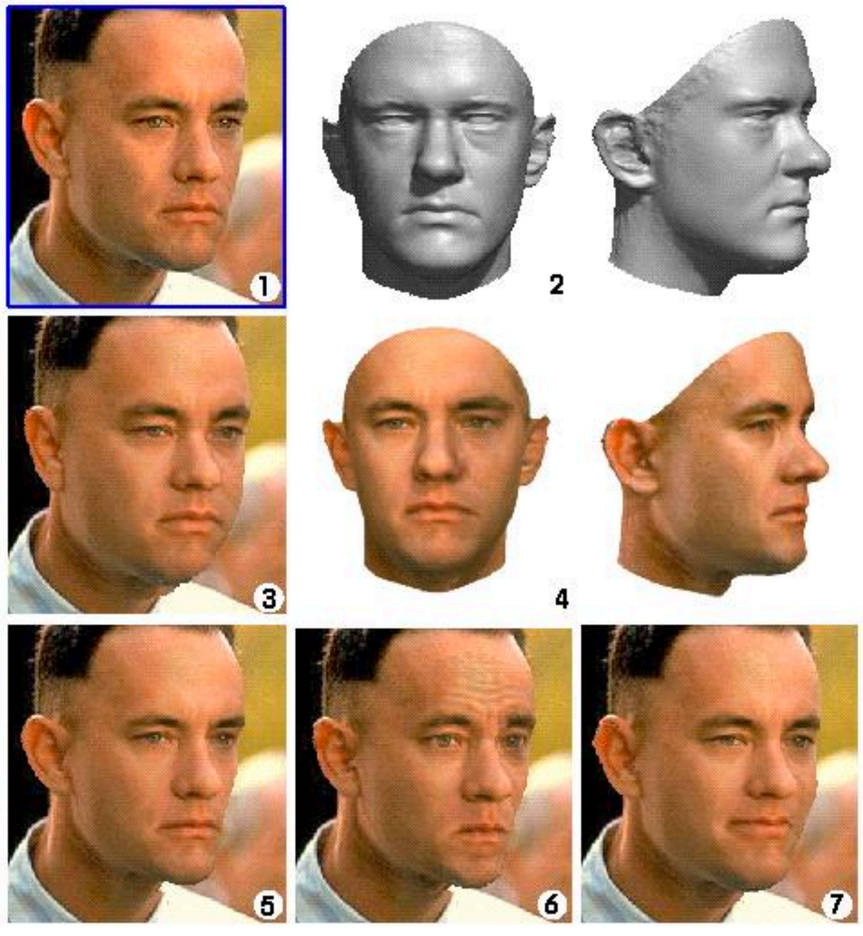
- **Idea that linear interpolation introduces blur**
- **Separation of shape and color**
- **Idea of non-rigid alignment of different images**
  - Applications to medical data
- **Applications, related to**
  - Special effects
  - Face recognition
  - Video frame interpolation
  - MPEG

More morphing madness

- **Gondry's Rolling Stones Video**

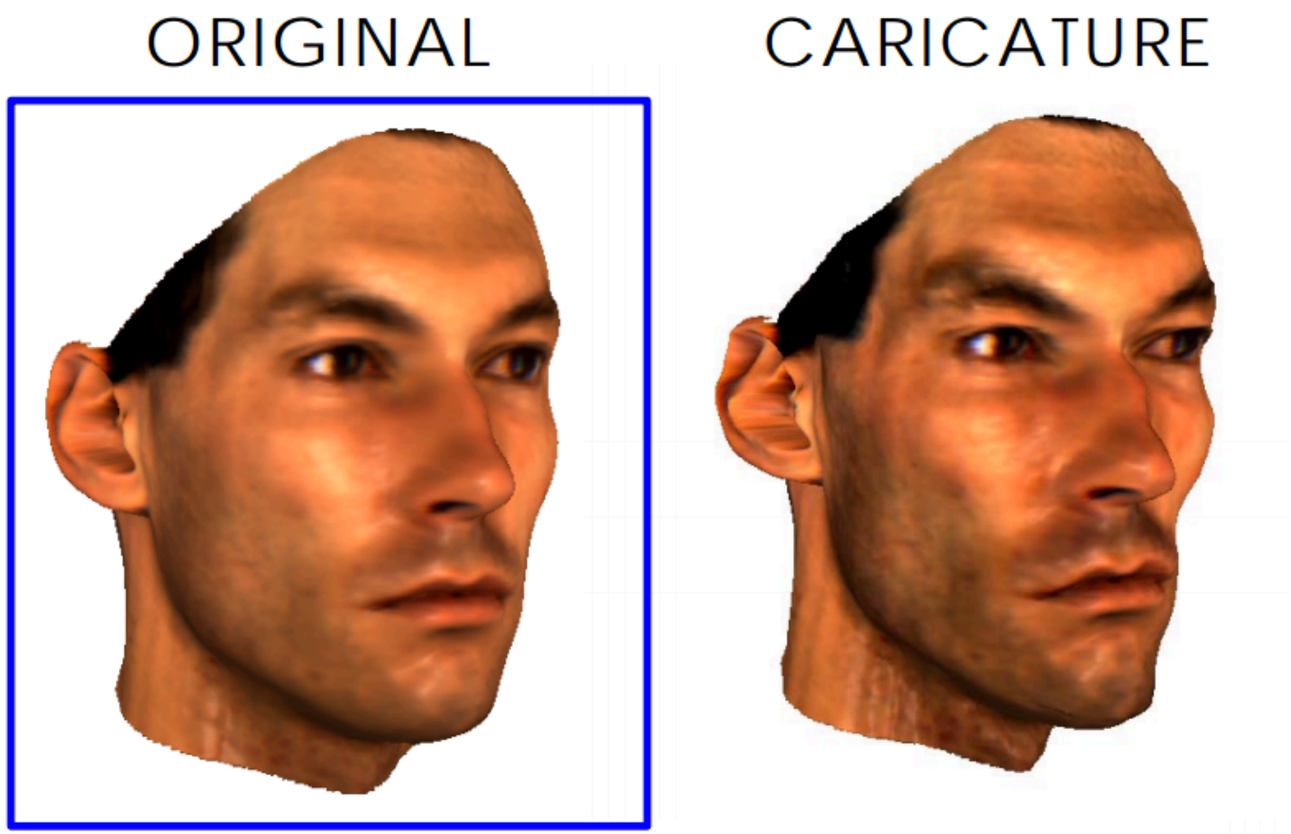


Morphable models



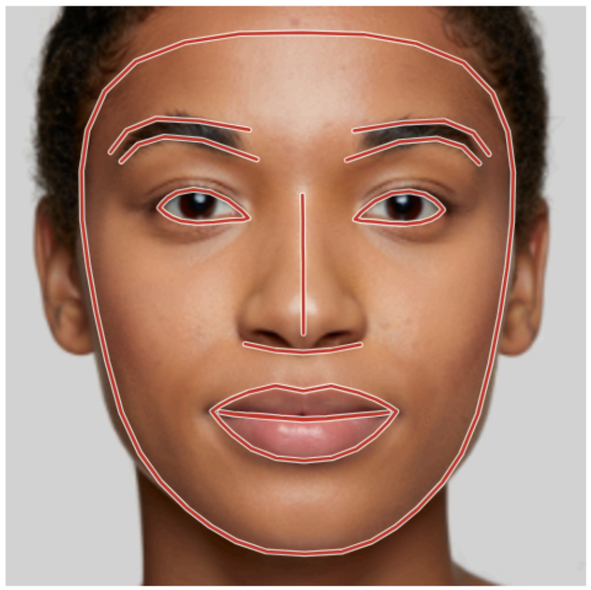
http://gravis.cs.unibas.ch/Sigg99.html

Morphable models

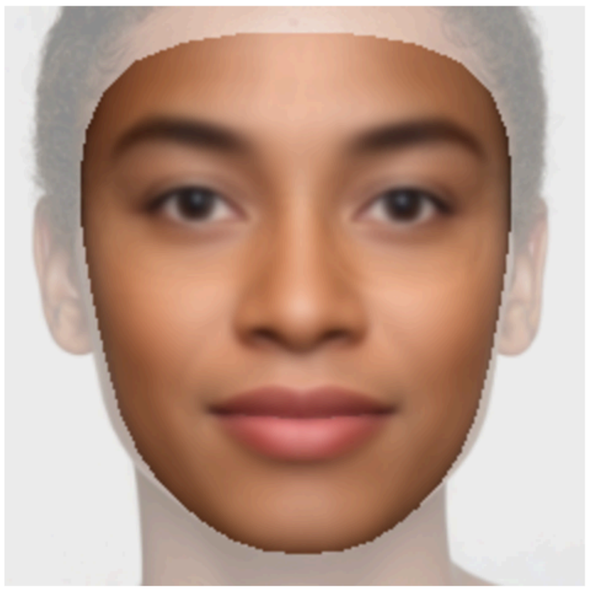


Demo: 2-D morphable face model

the photograph, model outline



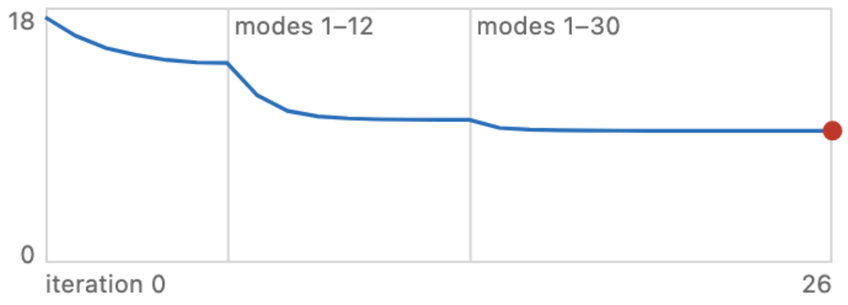
the model at iteration 26



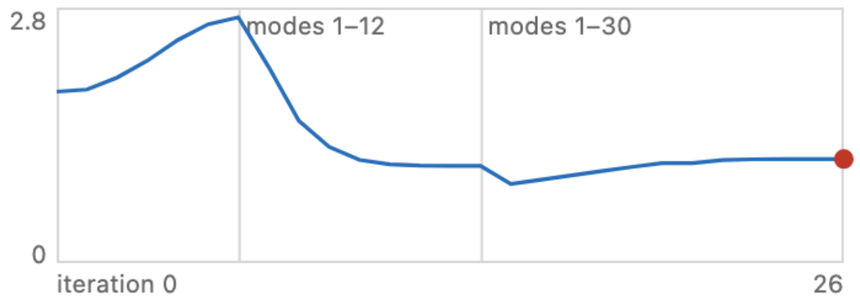
residual, photograph – model (× 2)

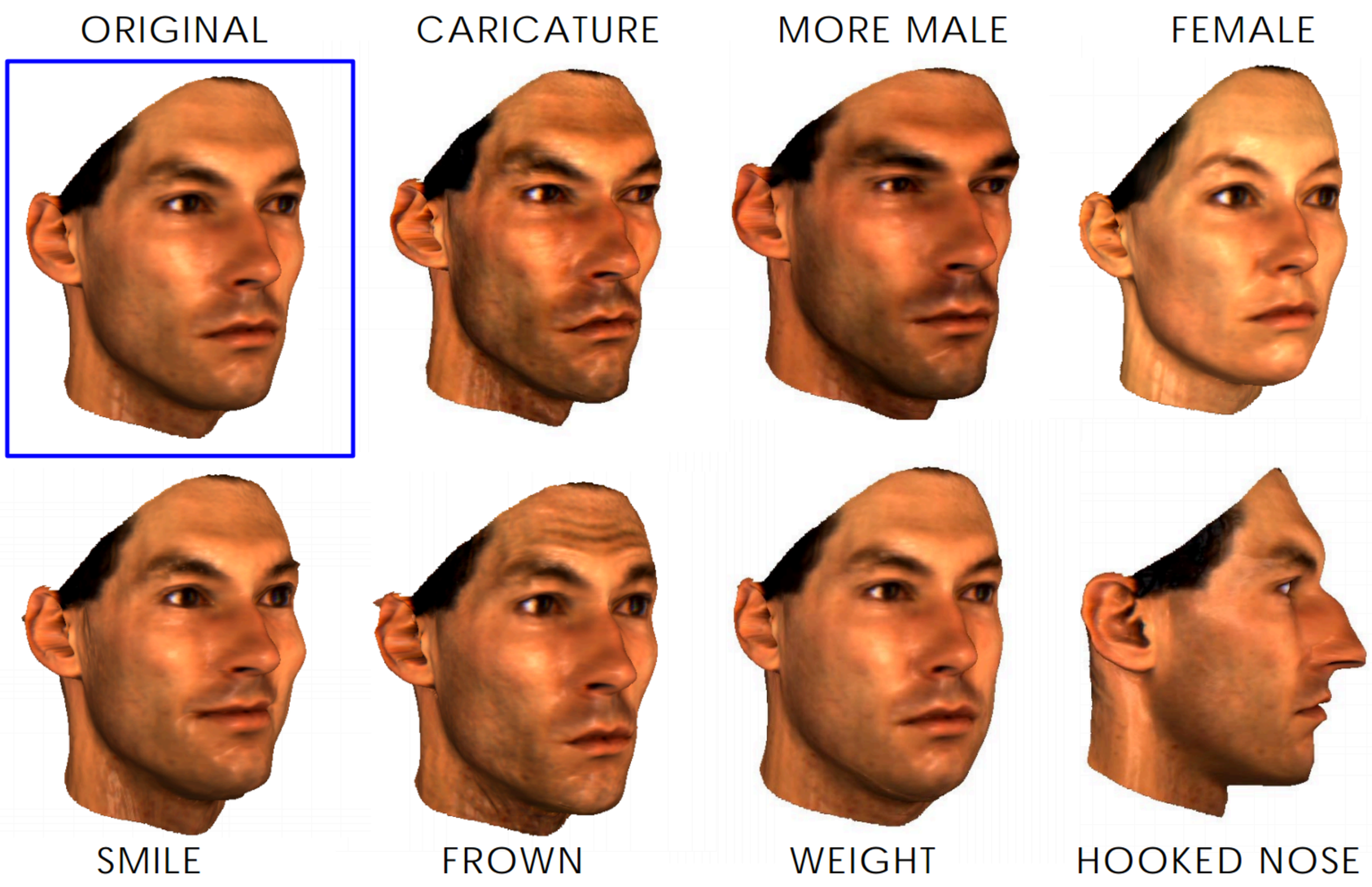


RMS residual 9.9 code values



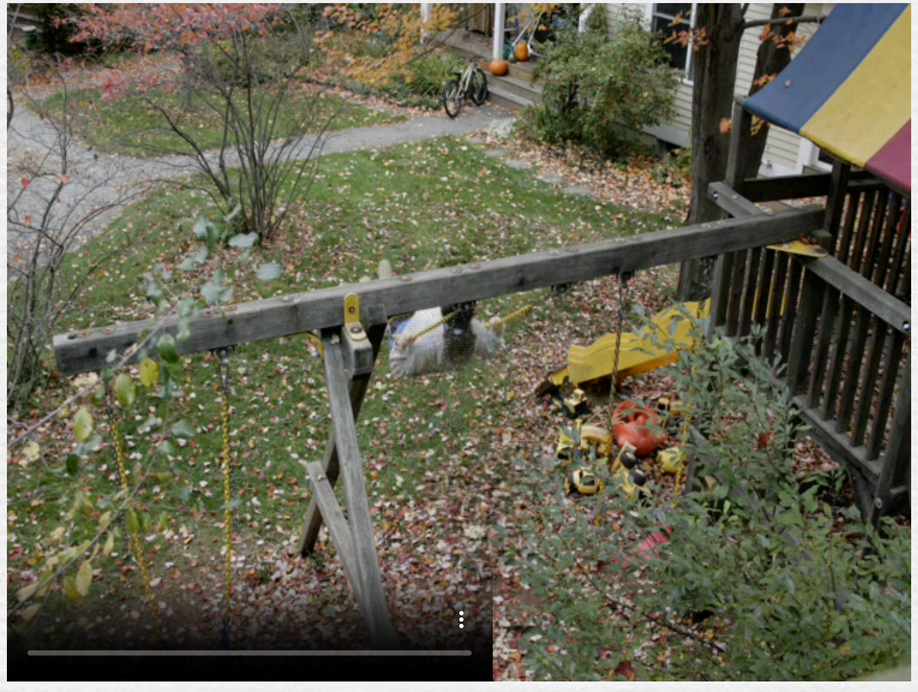
landmark distance (a check) 1.2% of eye width




ORIGINAL
CARICATURE
MORE MALE
FEMALE
SMILE
FROWN
WEIGHT
HOOKED NOSE

Application: Motion magnification

- *with Liu, Torralba, Freeman & Adelson [Siggraph 2005]*
- Analyze motion in video (robust to occlusion)
- Warp according to magnified motion



Motion magnification

- *with Liu, Torralba, Freeman & Adelson [Siggraph 2005]*
- Analyze motion in video (robust to occlusion)
- Magnify motion that is hard to see



Demo: Video magnification demo

Video magnification

Tiny temporal changes hidden in a video — the flush of a pulse, an imperceptible sway — are made visible by **amplifying** them. Pick a spatial pyramid, apply a **temporal band-pass** to each band's per-pixel time series, multiply the band-passed part by a gain **α**, and add it back. **Eulerian** (Wu et al., SIGGRAPH 2012) amplifies the band's *intensity*; **phase-based** (Wadhwa et al., SIGGRAPH 2013) amplifies its *local phase*, magnifying motion with far less noise. Everything runs in your browser; uploaded video never leaves your device.

Face | Face 2 | Baby | Baby 2 | Baby blanket | Crane | Guitar | Balance | Camera | Subway | **synth: pulse** | **synth: motion** | **load a video**

frame rate 30 fps ▶ play done in 319 ms · 60 frames · 160×120 · 30 fps

ORIGINAL



MAGNIFIED — EULERIAN



☐ **phase-based** (unchecked = Eulerian / linear) α amplification 18

temporal band-pass low 0.80 Hz high 2.5 Hz (log, 0.3–10 Hz · causal Butterworth IIR, no wrap-around)

pyramid levels 4 color boost 2.2 spatial blur σ 0.0

Motion Magnification

Ce Liu
Antonio Torralba
William T. Freeman
Fredo Durand
Edward H. Adelson

Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory

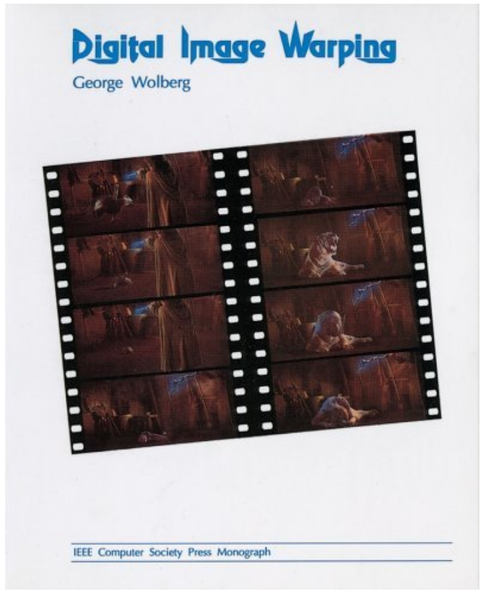


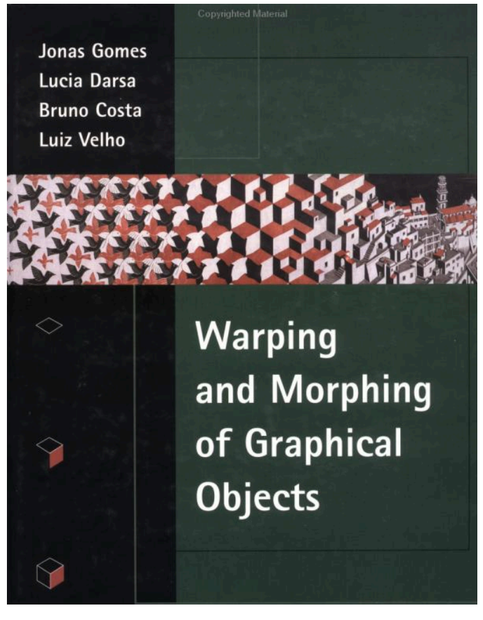
SIGGRAPH2005
ternational Conference on Computer Graphics and Interactive Techniques

References

Refs

- http://portal.acm.org/citation.cfm?id=134003&coll=GUIDE&dl=GUIDE&CFID=72901489&CFTOKEN=24335444
- http://www.visgraf.impa.br/cgi-bin/morphQuery.cgi?output=html
- http://www.cg.tuwien.ac.at/research/ca/mrm/
- http://w3.impa.br/~morph/sites.html
- http://w3.impa.br/~morph/sig-course/slides.html
- http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/morph.html
- http://www.fmrib.ox.ac.uk/~yongyue/thinplate.html
- http://www.uoguelph.ca/~mwirth/PHD_Chapter4.pdf





Software

- **http://www.morpheussoftware.net/**
- **http://www.debugmode.com/winmorph/**
- **http://www.stoik.com/products/morphman/mm1_main.htm**
- **http://www.creativecow.net/articles/zwar_chris/morph/index.html**
- **http://meesoft.logicnet.dk/SmartMorph/**
- **http://www.asahi-net.or.jp/~FX6M-FJMY/mop00e.html**
- **http://morphing-software-review.toptenreviews.com/**
- **http://www.freedownloadscenter.com/Search/morphing.html**

Advanced use of warping

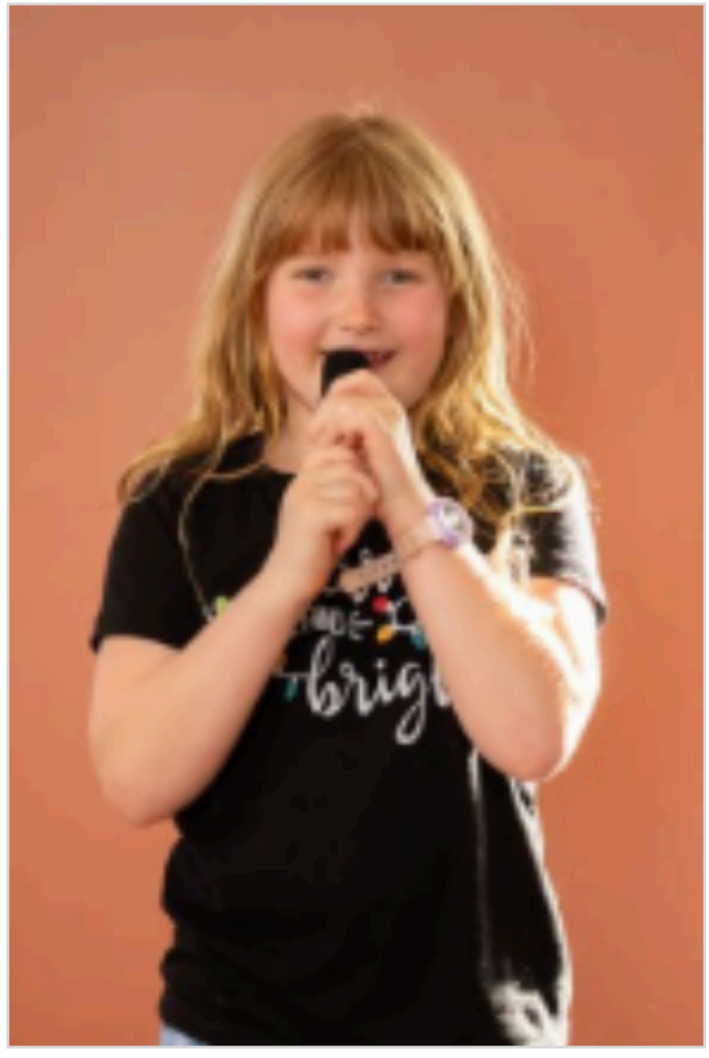
- http://www.collegehumor.com/video/6705603/north-korean-photoshop-tutorial

Demo: Liquify demo

original



liquify: drag to warp

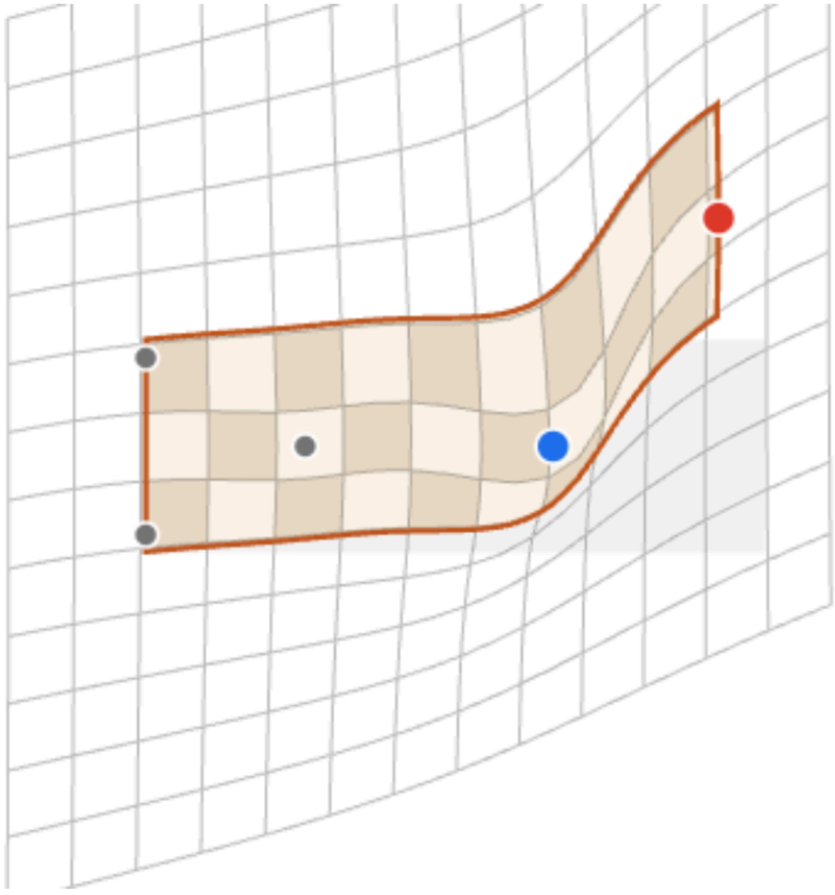


Push | Bloat | Pucker | Twirl | Reset

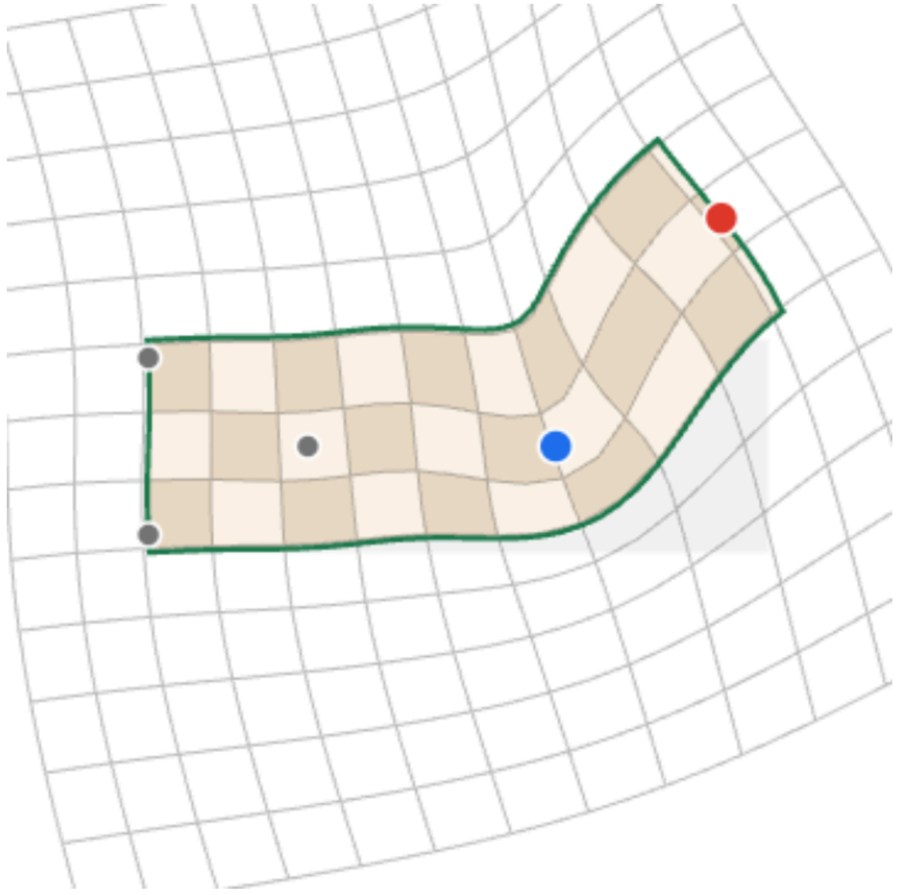
Demo: Shape-preserving warp

Speaker notes

Shape-preserving warping



Affine MLS



Rigid MLS (not an exact isometry)