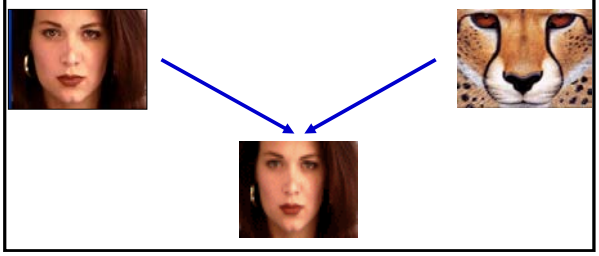
Lecture 7:

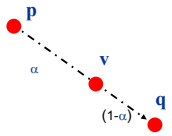
Image Morphing



Averaging vectors

- $\mathbf{v} = \mathbf{p} + \alpha(\mathbf{q} - \mathbf{p})$

  $= (1 - \alpha)\,\mathbf{p} + \alpha\,\mathbf{q}$ where $\alpha = \|\mathbf{q} - \mathbf{v}\|$



- **p** and **q** can be anything:
  - points on a plane (2D) or in space (3D)
  - Colors in RGB or HSV (3D)
  - Whole images … etc.

Idea #1: Cross-Dissolving / Cross-fading

- Interpolate whole images:

$$I_{halfway} = \alpha * I_1 + (1 - \alpha) * I_2$$

- This is called **cross-dissolving** in film industry
- But what if the images are not aligned?

Idea #2: Align, then cross-disolve

- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

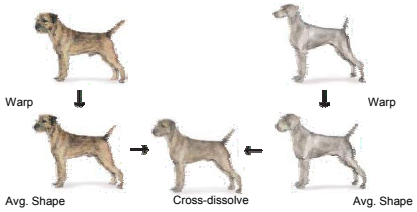
Failures: Averaging Images

- Global alignment doesn't work.

Dog Averaging

- What to do?
  - Cross-dissolve doesn't work
  - Global alignment doesn't work
    - Cannot be done with a global transformation (e.g. affine)
  - Any ideas?
- Feature matching!
  - Nose to nose, tail to tail, etc.
  - This is a local (non-parametric) warp

Idea #3: Local warp & cross-dissolve

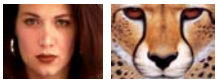


Morphing procedure:

1. Find the average shape (the "mean dog"☺)
   - local warping
2. Find the average color
   - Cross-dissolve the warped images

Morphing Sequence

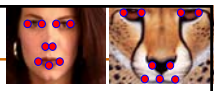
- **Input:** two images I_0 and I_N



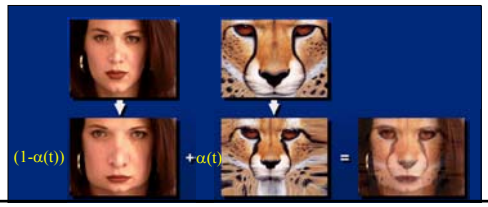
- **Output:** image seq. I_i, with *i=1..N-1*

- User specifies sparse correspondences on the images
  - Pairs of vectors $\{(p^0_j, p^N_j)\}$

Morphing



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



Warping

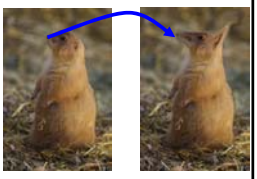
Warping

- Imagine your image is made of rubber
- warp the rubber

Careful: warp vs. inverse warp

How do you perform a given warp:

- Forward warp
  - Potential gap problems



- Inverse lookup the most useful
  - For each output pixel
    - Lookup color at inverse-warped location in input

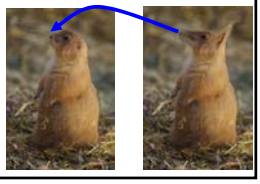
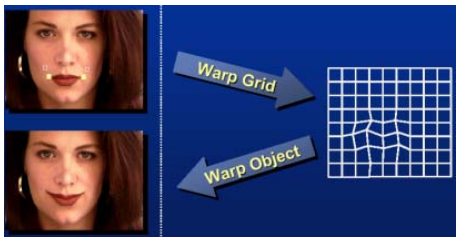


Image Warping – non-parametric

- 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

Warp specification - sparse

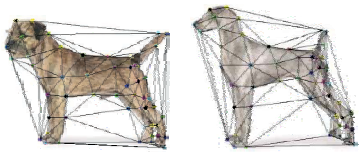
- How can we specify the warp?

  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

Triangular Mesh

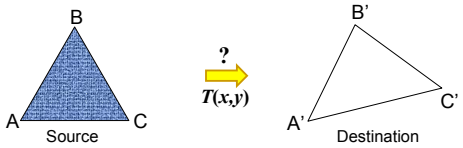


1. Input correspondences at key feature points
2. Define a triangular mesh over the points

   (ex. Delaunay Triangulation)
   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately
   - How do we warp a triangle?
   - ***3 points = affine transformation***!
   - Just like texture mapping

Slide Alyosha Efros

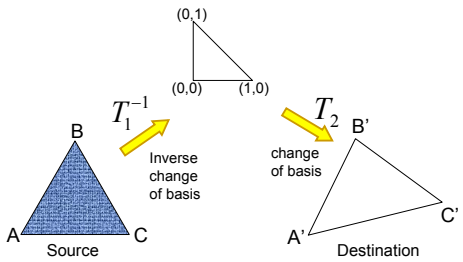
Example: warping triangles



- Given two triangles: ABC and A'B'C' in 2D (3 points = 6 constrains)
- Need to find transform T to transfer all pixels from one to the other.
- What kind of transformation is T?
  - affine
- How can we compute the transformation matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

HINT: warping triangles



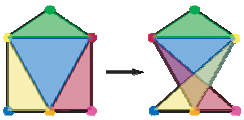
Don't forget to move the origin too!

Problems with triangulation morphing

- Not very continuous - only C^0

Fig. L. Darsa

- Folding problems - relationship between feature locations may not be the same between two objects.

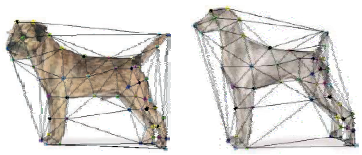


Warp as interpolation

- We are looking for a warping field
  - A function that given a 2D point, returns a warped 2D point
- We have a sparse number of correspondences
  - These specify values of the warping field
- This is an interpolation problem
  - Given sparse data, find smooth function

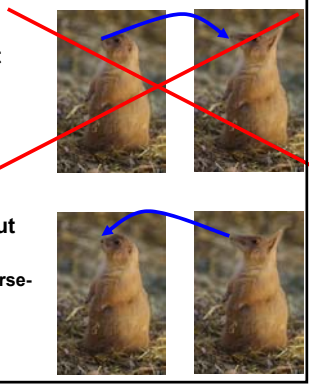
Linear Interpolation

- How do we create an intermediate warp at time t?
  - Assume $\alpha(t) = [0,1]$
  - Simple linear interpolation of each feature pair
  - $(1-\alpha(t))\ \mathbf{p}_0+\alpha(t)\ \mathbf{p}_1$ for corresponding features $\mathbf{p}_0$ and $\mathbf{p}_1$



Applying a warp: USE INVERSE

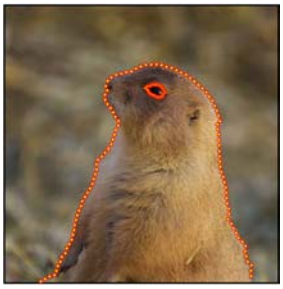
- Forward warp:
  - For each pixel in **input** image
    - Paste color **to warped** location in output
  - Problem: gaps
- Inverse warp
  - For each pixel in **output** image
    - Lookup color **from inverse-warped** location



Morphing

Input images

Feature correspondences

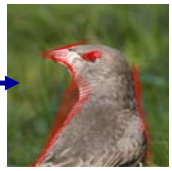



- The feature locations will be our y_i

Interpolate feature location

- Provides the x_i

Warp each image to intermediate location

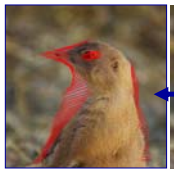



Two different warps: Same target location, different source location

i.e. the x_i are the same (intermediate locations), the y_i are different (source feature locations)

Note: the y_i do not change along the animation, but the x_i are different for each intermediate image

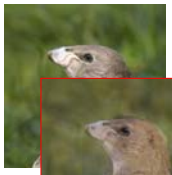
Here we show α=0.5 (the y_i are in the middle)




Warp each image to intermediate location



Interpolate colors linearly

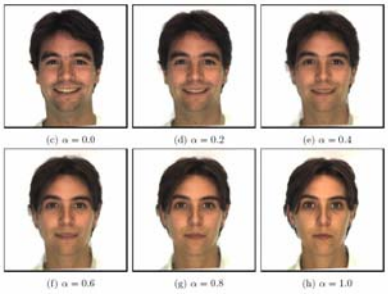

Interpolation weight are a function of time:

$$C=(1-\alpha(t))f^0_t(\mathbf{I}_0)+\alpha(t)f^1_t(\mathbf{I}_1)$$

Bells and whistles

Morphing & matting

- Extract foreground first to avoid artifacts in the background



Uniform morphing

Figure 4. Uniform metamorphosis

Non-uniform morphing

Figure 5. Nonuniform metamorphosis

http://www-cs.ccny.cuny.edu/~wolberg/pub/cgi96.pdf

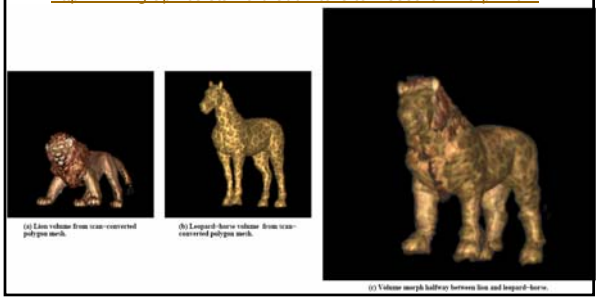
Dynamic Scene

- Lots of manual work
- Automatic:
  - facial features detection and localization
  - face tracking

http://www.youtube.com/watch?v=ZI9OYMRwN1Q

3D Morphing

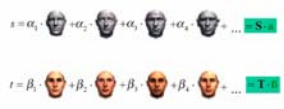
- Feature-Based Volume Metamorphosis Lerios, Garfinkle, and Levoy.
- http://www-graphics.stanford.edu/~tolis/toli/research/morph.html



The Morphable Face Model

- Again, assuming that we have ***m*** such vector pairs in full correspondence, we can form new shapes $\mathbf{S}_{model}$ and new appearances $\mathbf{T}_{model}$ as:

$$\mathbf{S}_{model} = \sum_{i=1}^{m} a_i \mathbf{S}_i \qquad \mathbf{T}_{model} = \sum_{i=1}^{m} b_i \mathbf{T}_i$$

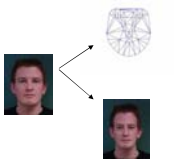


- If number of basis faces ***m*** is large enough to span the face subspace then:
  - Any new face can be represented as a pair of vectors $(\alpha_1, \alpha_2, \ldots, \alpha_m)^T$ and $(\beta_1, \beta_2, \ldots, \beta_m)^T$!

The Morphable Face Model

The actual structure of a face is captured in:

- the shape vector **S** = $(x_1, y_1, x_2, \ldots, y_n)^T$, containing the (x, y) coordinates of the n vertices of a face, and
- the appearance (texture) vector **T** = $(R_1, G_1, B_1, R_2, \ldots, G_n, B_n)^T$, containing the color values of the mean-warped face image.



Shape S

Appearance T

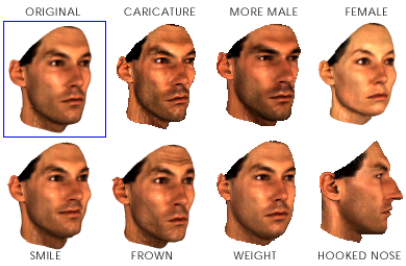
Subpopulation Means

- Examples:
  - Happy faces
  - Young faces
  - Asian faces
  - Etc.
  - Sunny days
  - Rainy days
  - Etc.
  - Etc.

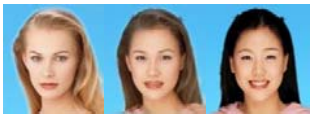
Average female

Average male

Using 3D Geometry: Blanz & Vetter, 1999



show SIGGRAPH video



- Given two photos, produce a 60-frame morph animation