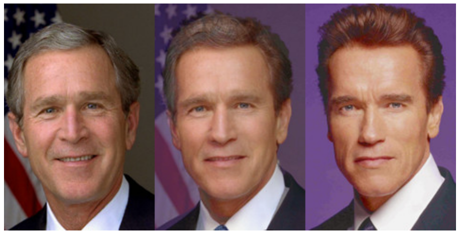
Image Morphing

Morphing is turning one image into another (through a seamless transition)



Source: Wikepedia

Application: Movie Special Effects

- First movies with morphing
  - *Willow*, 1988
  - *Indiana Jones and the Last Crusade*, 1989
- First music video with morphing
  - *Black or White*, Michael Jackson,1991

Application: Registration /Alignment

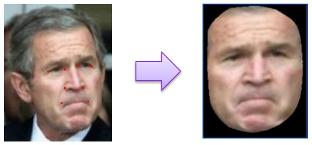
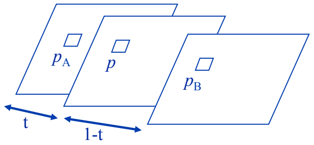


Image Cross-Dissolve

- Pixel-by-pixel color interpolation
- Each pixel p at time $t \in [0, 1]$ is computed by combining a fraction of each pixel's color at the same coordinates in source images A and B:

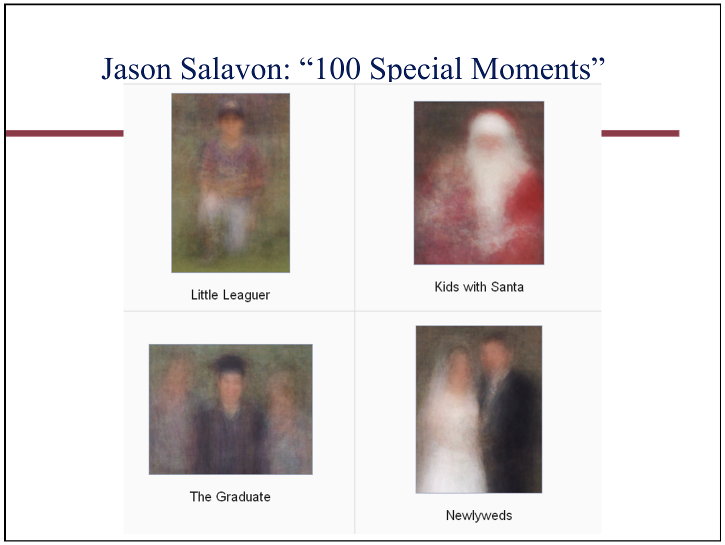
$$p = (1 - t)\, p_A + t\, p_B$$



- Easy, but looks artificial, non-physical

Jason Salavon: “The Late Night Triad”

http://www.salavon.com/



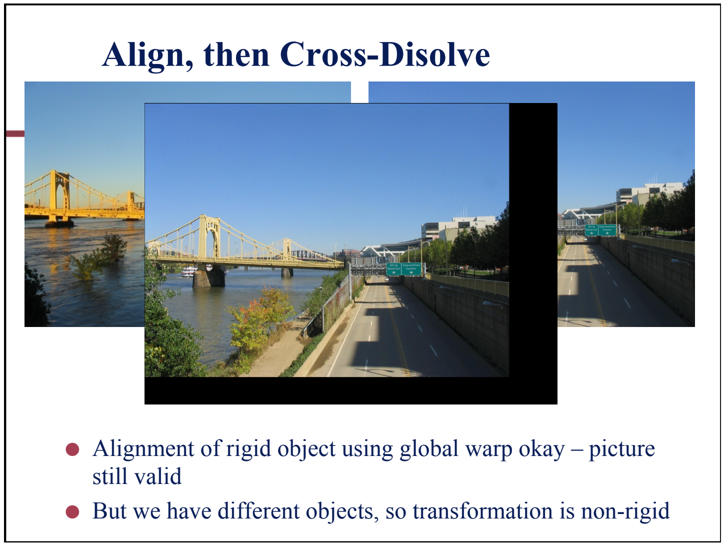
Jason Salavon: “100 Special Moments”

Little Leaguer

Kids with Santa

The Graduate

Newlyweds



Align, then Cross-Disolve

- Alignment of rigid object using global warp okay – picture still valid
- But we have different objects, so transformation is non-rigid

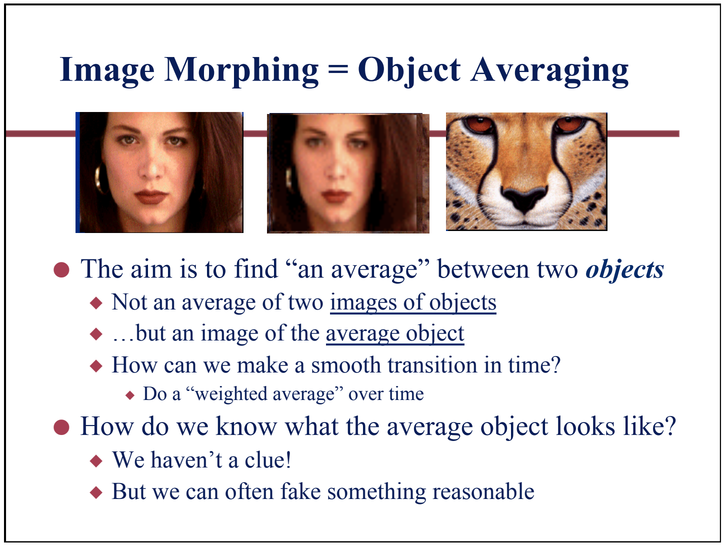
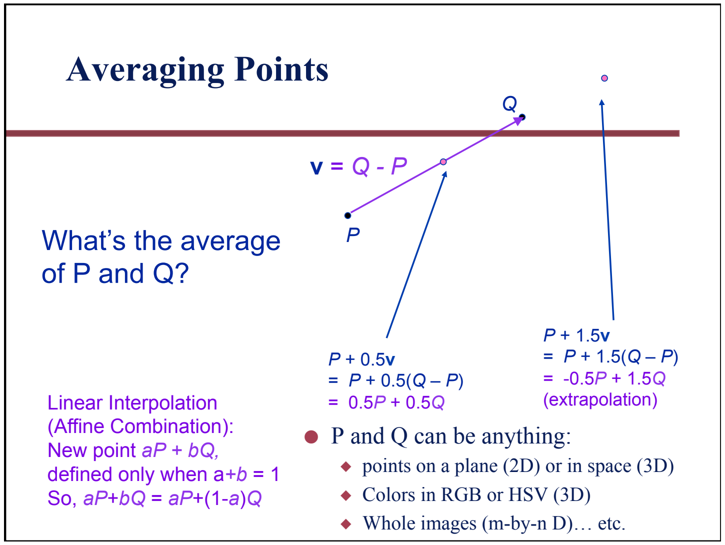


Image Morphing = Object Averaging

- The aim is to find “an average” between two ***objects***
  - Not an average of two images of objects
  - …but an image of the average object
  - How can we make a smooth transition in time?
    - Do a “weighted average” over time
- How do we know what the average object looks like?
  - We haven’t a clue!
  - But we can often fake something reasonable



Averaging Points

Q

$\mathbf{v}$ = Q - P

P

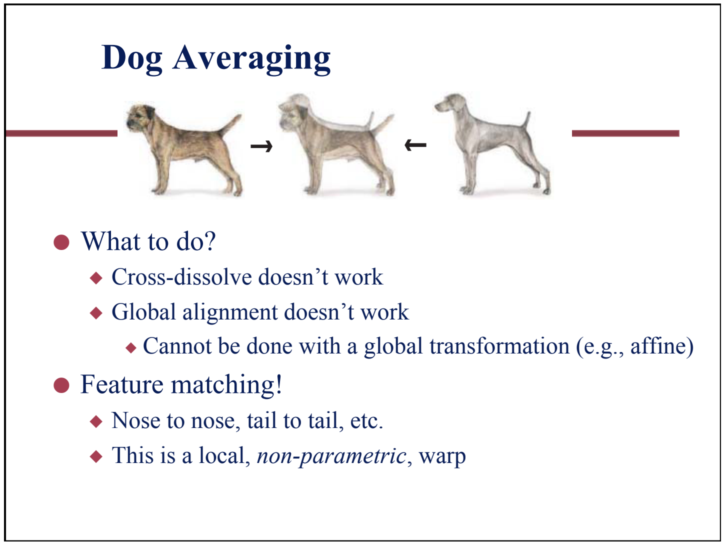
What's the average of P and Q?

P + 0.5$\mathbf{v}$
= P + 0.5(Q – P)
= 0.5P + 0.5Q

P + 1.5$\mathbf{v}$
= P + 1.5(Q – P)
= -0.5P + 1.5Q
(extrapolation)

Linear Interpolation
(Affine Combination):
New point $aP + bQ$,
defined only when $a+b$ = 1
So, $aP+bQ = aP+(1-a)Q$

- P and Q can be anything:
  - points on a plane (2D) or in space (3D)
  - Colors in RGB or HSV (3D)
  - Whole images (m-by-n D)… etc.



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)
- Feature matching!
  - Nose to nose, tail to tail, etc.
  - This is a local, *non-parametric*, warp

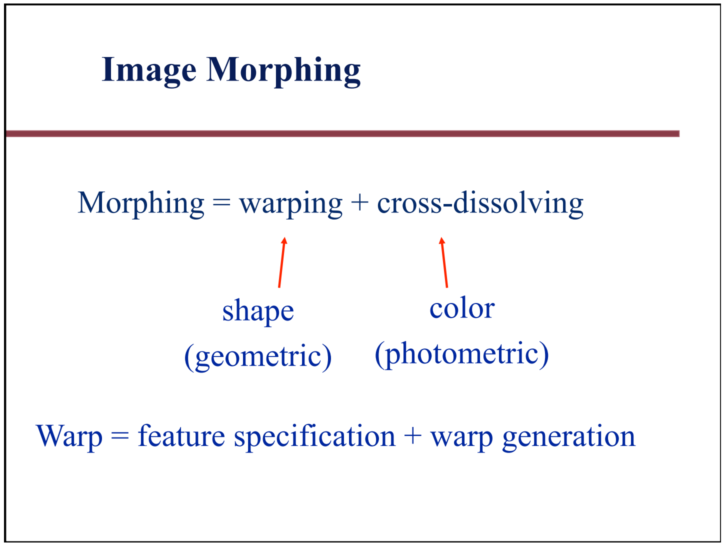


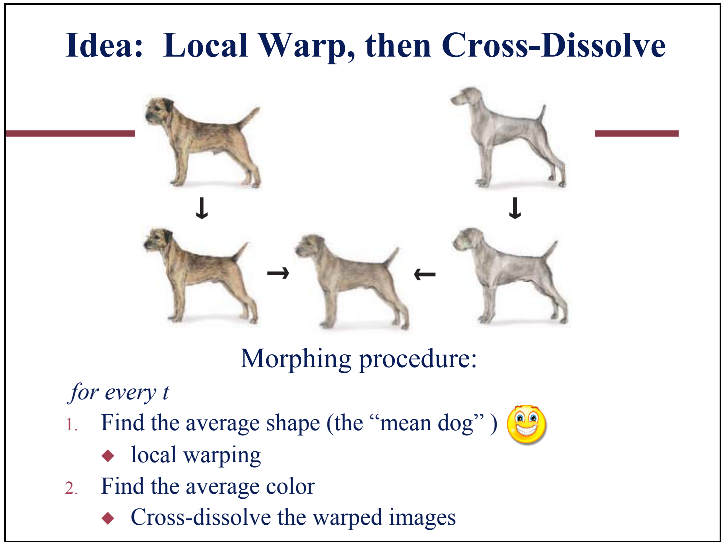
Image Morphing

Morphing = warping + cross-dissolving

shape
(geometric)

color
(photometric)

Warp = feature specification + warp generation



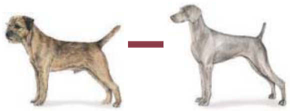
Idea: Local Warp, then Cross-Dissolve

Morphing procedure:

for every t

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

Local (Non-Parametric) Image Warping



- Need to specify a more detailed warp function
  - Global warps are functions of a few (e.g., 2, 4, 8) parameters
  - Non-parametric warps $u(x, y)$ and $v(x, y)$ can be defined independently for every single location x, y
  - Once we know vector field u, v we can easily warp each pixel (use backward warping with interpolation)

Mesh-based Warp Specification

- How can we specify the warp?

  Specify corresponding *spline control points*

  - *Interpolate* to a complete warping function

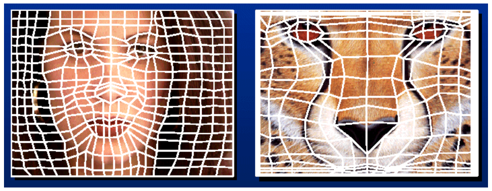
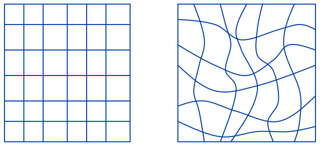


Image Morphing

- **Mesh-based image morphing**
  - G. Wolberg, *Digital Image Warping*, 1990
  - Warp between corresponding grid points in source and destination images
  - Interpolate between grid points, e.g., linearly using three closest grid points

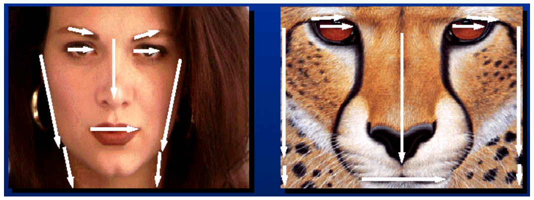


  - Fast, but hard to control so as to avoid unwanted distortions

Sparse Warp Specification

- How can we specify the warp?

  2. Specify corresponding line segments (***vectors***)
     - *Interpolate* to a complete warping function



Feature-based Image Morphing

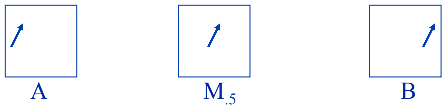
- T. Beier and S. Neely, *Proc. SIGGRAPH* 1992
- Distort color *and* shape
  - ⇒ image warping + cross-dissolving
- Warping transformation partially defined by user interactively specifying corresponding pairs of vectors in the source and destination images; only a sparse set is required (but carefully chosen)
- Compute dense pixel correspondences, defining continuous mapping function, based on weighted combination of displacement vectors of a pixel from all of the input vectors
- Interpolate pixel positions and colors (2D linear interpolation)

Beier and Neely Algorithm

- **Given**: 2 images, A and B, and their corresponding sets of line segments, L_A and L_B, respectively
- **Foreach** intermediate frame time $t \in [0, 1]$ **do**
  - **Linearly interpolate** the *position* of each line
    - $L_t[i] = \text{Interpolate}(L_A[i], L_B[i], t)$
  - **Warp** image A to destination shape
    - $WA = \text{Warp}(A, L_A, L_t)$
  - **Warp** image B to destination shape
    - $WB = \text{Warp}(B, L_B, L_t)$
  - **Cross-dissolve** by fraction t
    - $\text{MorphImage} = \text{CrossDissolve}(WA, WB, t)$

Example: Translation

- Consider images where there is one line segment pair, and it is **translated** from image A to image B:

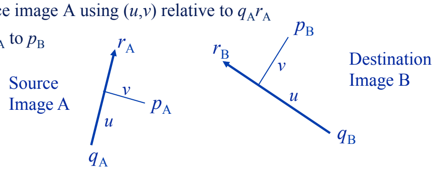


- First, linearly interpolate position of line segment in M
- Second, for each pixel (x, y) in M, find corresponding pixels in $A(x\text{-a}, y)$ and $B(x\text{+a}, y)$, and average them

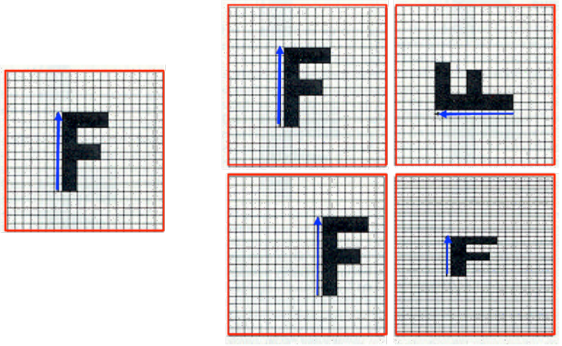
Line Feature-based Warping

- **Goal**: Define a continuous function that warps a source image to a destination image from a sparse set of corresponding, oriented, **line segment features -** each pixel's position defined relative to these line segments
- **Warping with one line pair**:

```
foreach pixel $p_B$ in destination image B do
    find dimension-less coordinates $(u,v)$ relative to oriented line segment $q_B r_B$
    find $p_A$ in source image A using $(u,v)$ relative to $q_A r_A$
    copy color at $p_A$ to $p_B$
```

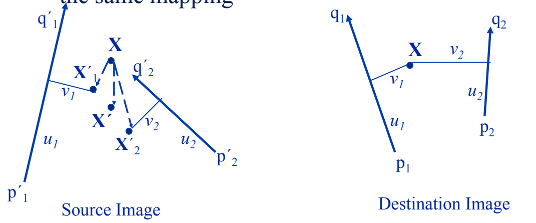


Single Line-Pair Examples



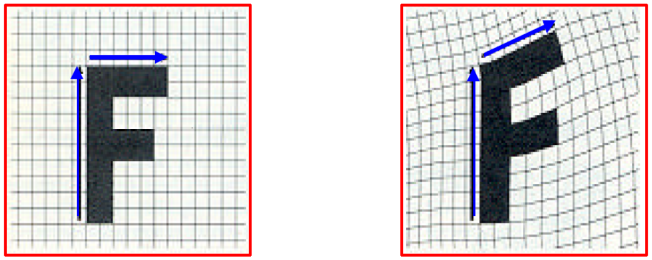
Feature-based Warping (cont.)

- **Warping with multiple line pairs**
  - Use a **weighted combination** of the points defined by the same mapping

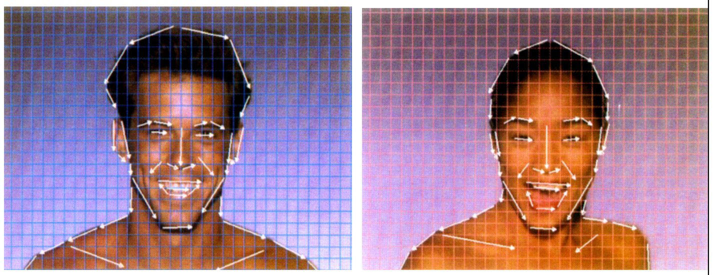


X′ = weighted average of D_1 and D_2, where D_i = $\mathbf{X}'_i$ - **X**, and weight = (length(p_iq_i)^c / (a + |v_i|))^b , for constants a, b, c

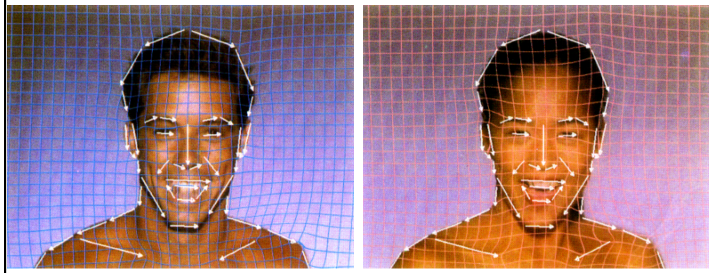
Resulting Warp



2 Input Images with Line Correspondences



Images Warped to Same “Shape”



Warped Shapes without Grid Lines

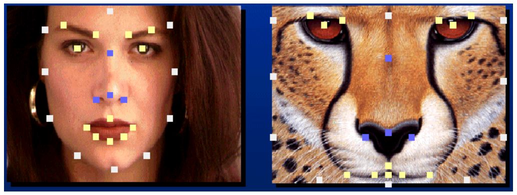
Cross-Dissolved Result

Sparse Warp Specification

- How can we specify the warp?

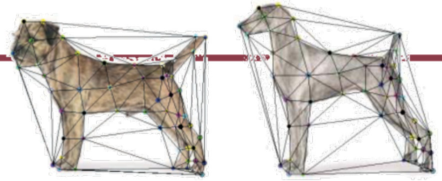
  Specify corresponding ***points***

  - *Interpolate* to a complete warping function



How do we go from feature points to pixels?

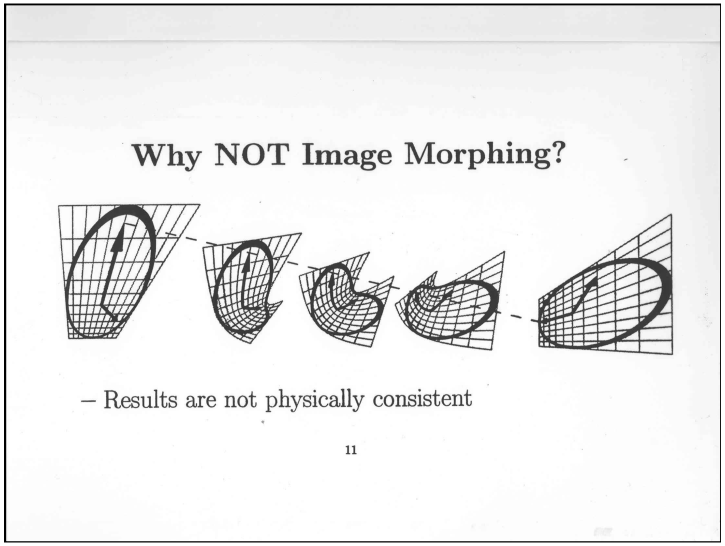
Point Feature Morphing: Triangular Mesh



1. Input correspondences at landmark/fiducial feature points
2. Define a triangular mesh over the points
   - Same mesh in both images
   - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately from source to destination
   - How do we warp a triangle?
   - 3 points = affine warp
   - Just like texture mapping

Morphing between Two Image Sequences

- **Goal**: Given two animated sequences of images, create a morph sequence
- User defines corresponding line segments in pairs of **key frames** in the two sequences
- At frame *i*, compute the two sets of line segments by interpolating between the nearest bracketing key frames' line sets
- Apply 2-image morph algorithm for $t = 0.5$ only to obtain morph frame *i*



Why NOT Image Morphing?

– Results are not physically consistent

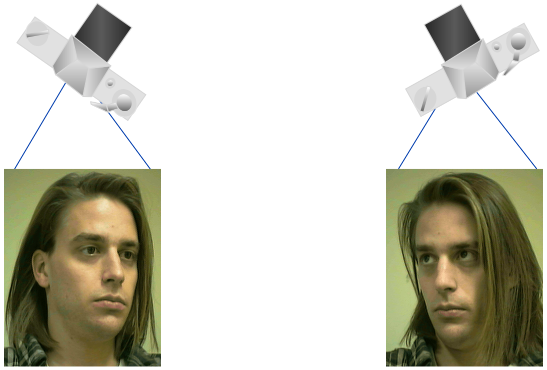
11

Geometrically-Correct Pixel Reprojection

- What geometric information is needed to generate **optically-correct** virtual camera views?
  - Dense pixel correspondences between two input views
  - Known geometric relationship between the two cameras
    - Epipolar geometry

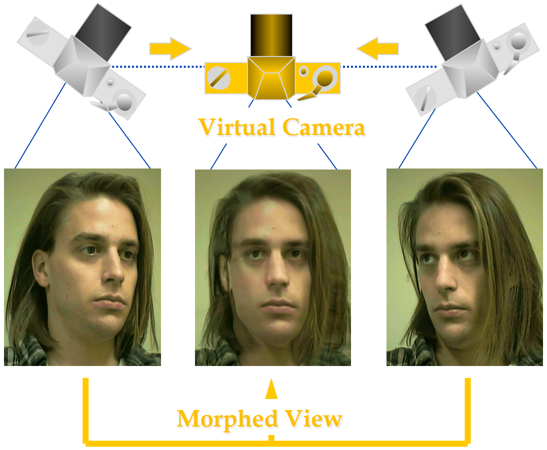
View Morphing

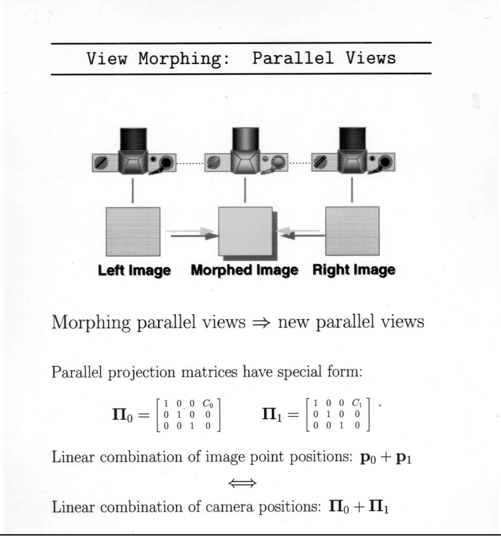
- Seitz and Dyer, *Proc. SIGGRAPH* 1995
- **Given**: Two views of an unknown rigid scene, with no camera information known, **compute new views from a virtual camera** at viewpoints in-between two input views



Photograph

Photograph





View Morphing: Parallel Views

Left Image
Morphed Image
Right Image

Morphing parallel views ⇒ new parallel views

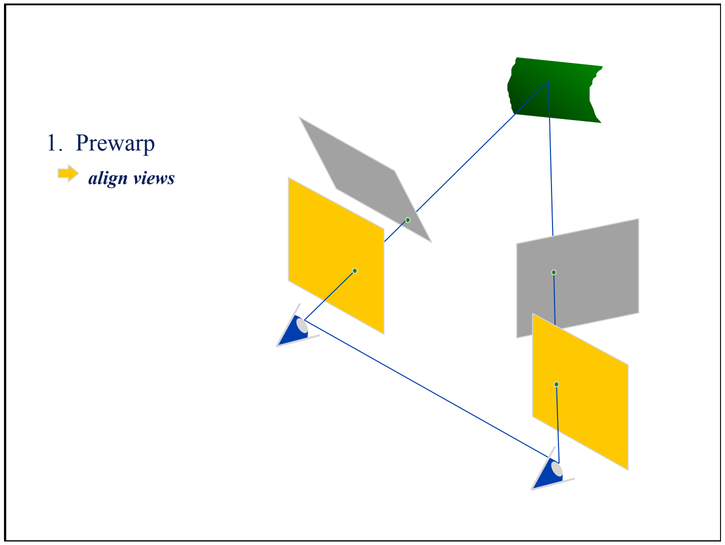
Parallel projection matrices have special form:

$$\mathbf{\Pi}_0 = \begin{bmatrix} 1 & 0 & 0 & C_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{\Pi}_1 = \begin{bmatrix} 1 & 0 & 0 & C_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

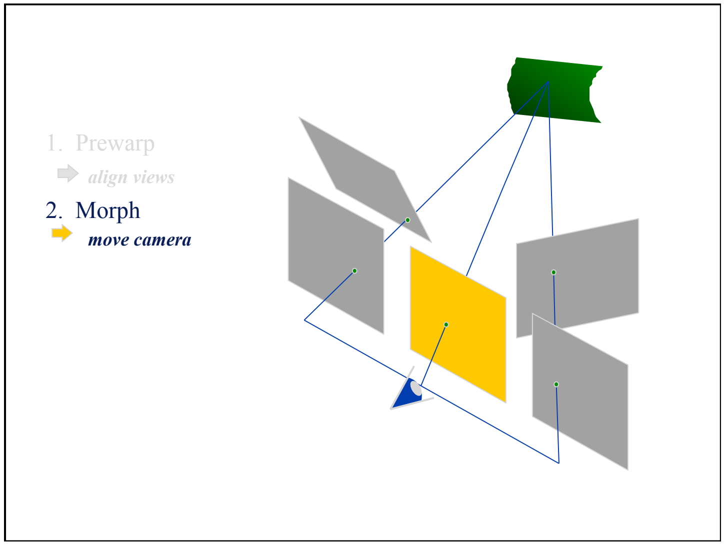
Linear combination of image point positions: $\mathbf{p}_0 + \mathbf{p}_1$

$$\Longleftrightarrow$$

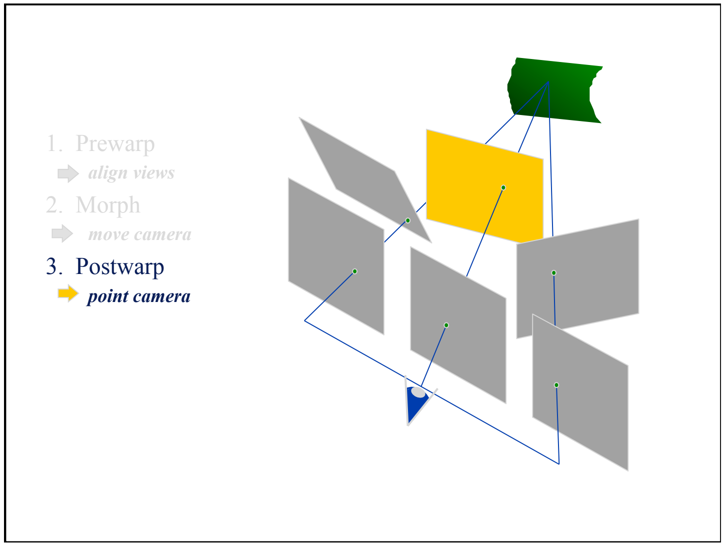
Linear combination of camera positions: $\mathbf{\Pi}_0 + \mathbf{\Pi}_1$



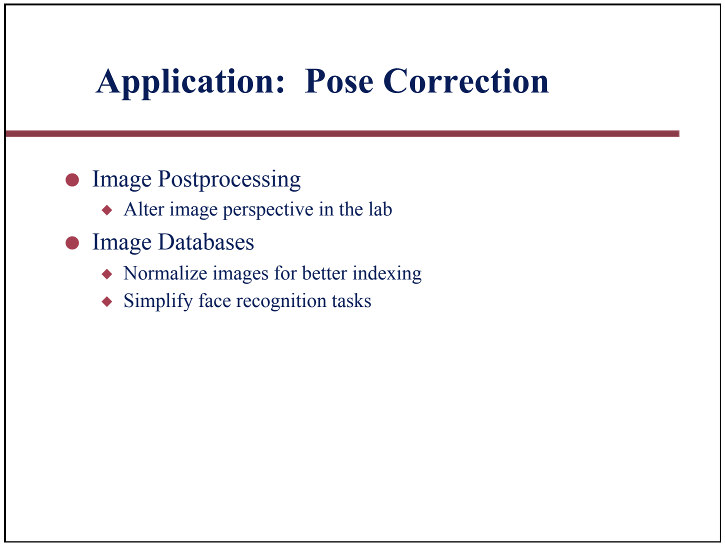
1. Prewarp
   - *align views*



1. Prewarp
   - *align views*
2. Morph
   - *move camera*



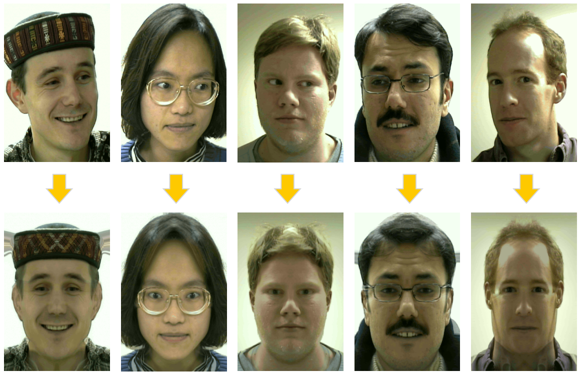
1. Prewarp
   - *align views*
2. Morph
   - *move camera*
3. Postwarp
   - *point camera*



Application: Pose Correction

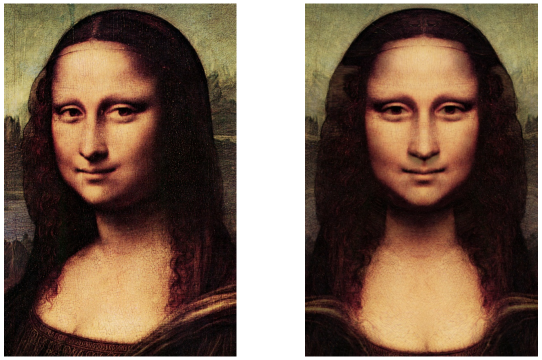
- Image Postprocessing
  - Alter image perspective in the lab
- Image Databases
  - Normalize images for better indexing
  - Simplify face recognition tasks

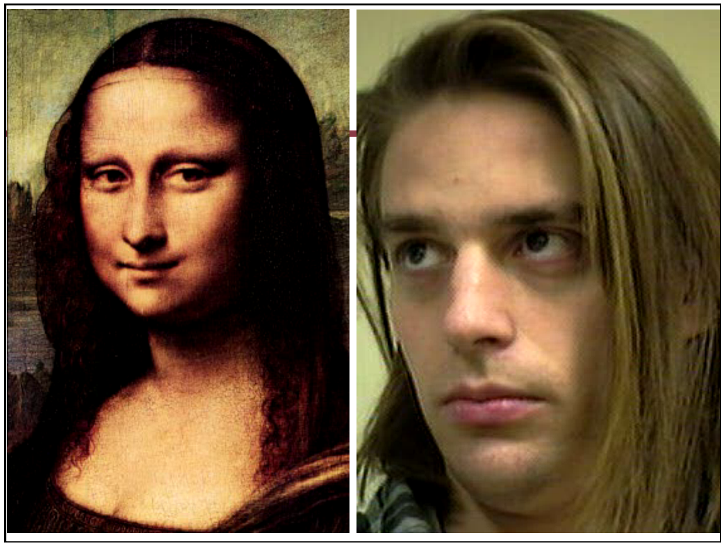
Original Photographs



Frontal Poses

Another Example



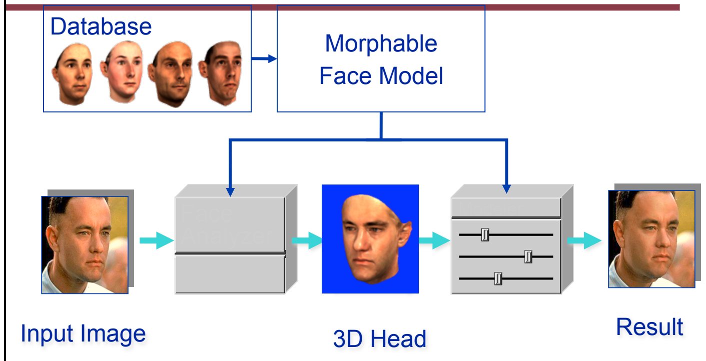


A Morphable Model for the Synthesis of 3D Faces

V. Blanz and T. Vetter

Proc. SIGGRAPH 1999

Synthesis of Faces



Database

Morphable Face Model

Input Image

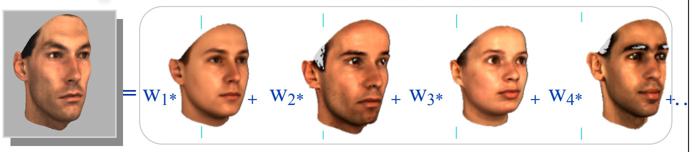
3D Head

Result

Approach: Example-based Modeling of Faces

2D Image

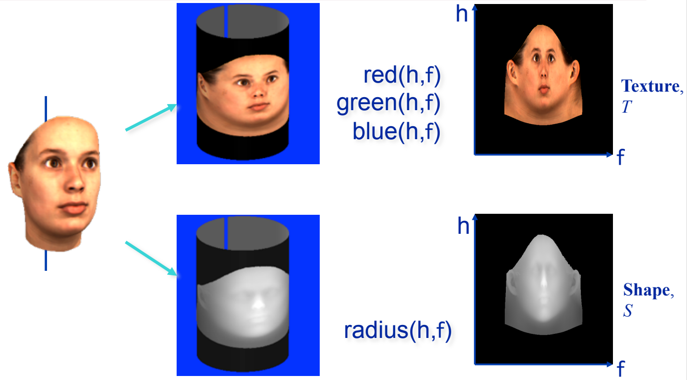
3D Face Models



Linear combination of exemplar face models
(all exemplar faces are in full correspondence)

200 exemplar faces

Face Representation using Cylindrical Coordinates



red(h,f)
green(h,f)
blue(h,f)

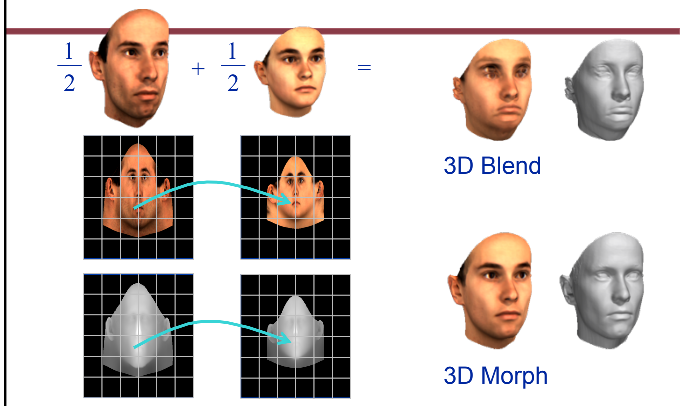
Texture, *T*

radius(h,f)

Shape, *S*

70,000 points per image

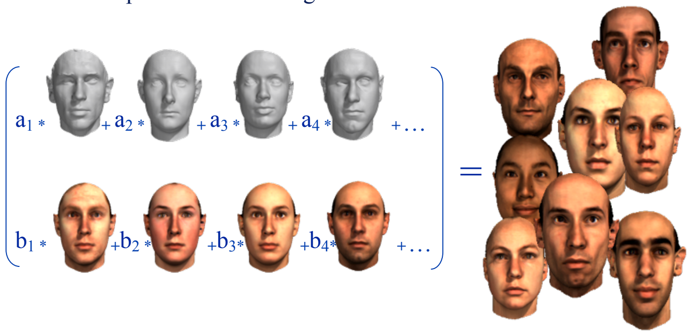
Morphing 3D Faces



Dense point correspondences across all exemplar faces

Vector Space of 3D Faces

- A Morphable Model can generate new faces

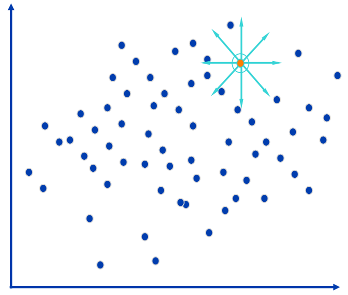


a_1 * + a_2 * + a_3 * + a_4 * +…
b_1 * + b_2 * + b_3 * + b_4 * +…
=

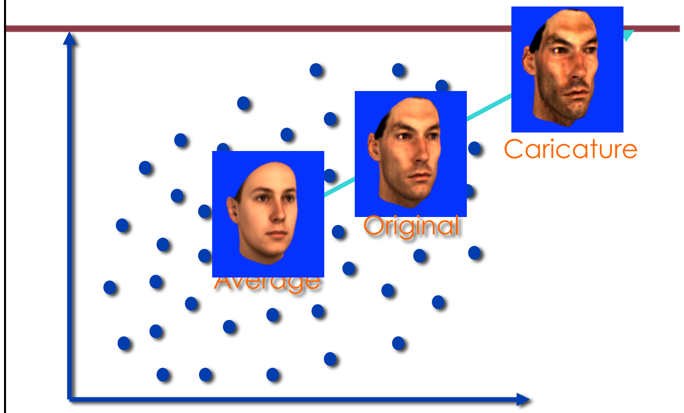
Modeling the Appearance of Faces

A face is represented as a point in a “face space”

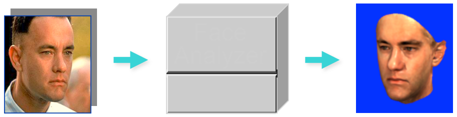
- Which directions code for specific attributes?



Modeling in Face Space



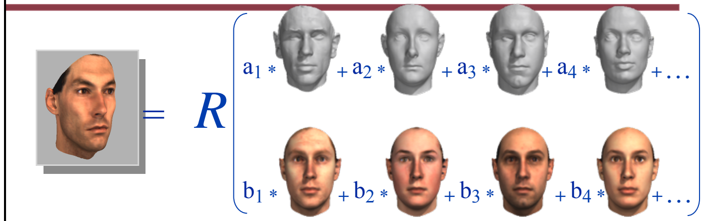
3D Shape from Images



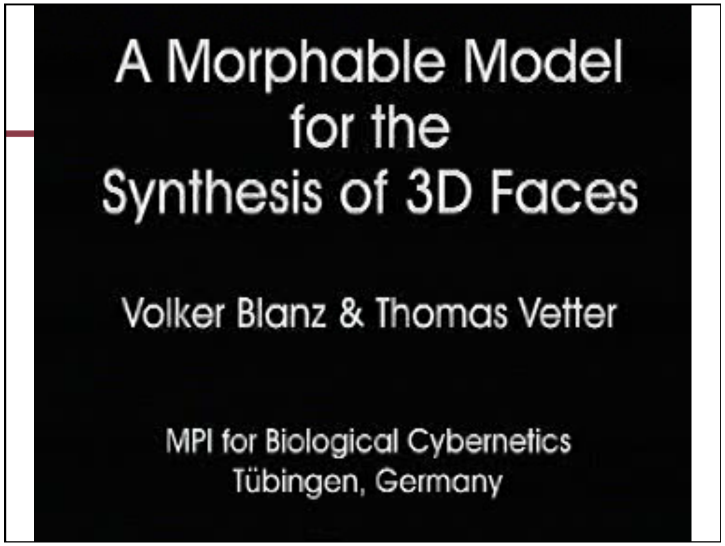
Input Image

3D Head

Matching a Morphable 3D Face Model


= R
a1 * + a2 * + a3 * + a4 * + . . .
b1 * + b2 * + b3 * + b4 * + . . .

Optimization problem over the range of values of a_i and b_i delimited by the training faces, plus rendering parameters, ρ, such as camera pose and illumination

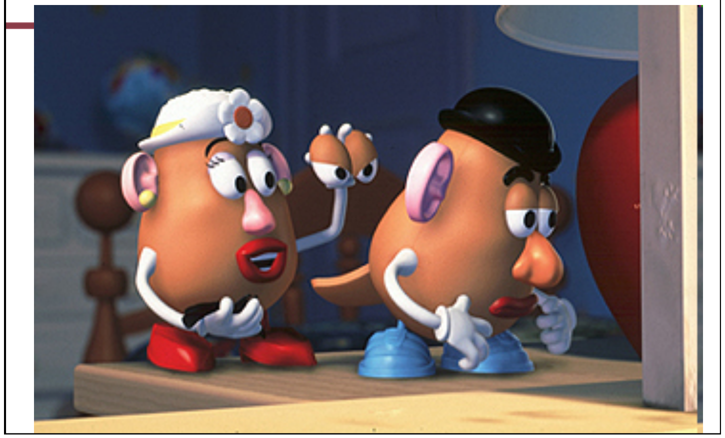


A Morphable Model for the Synthesis of 3D Faces

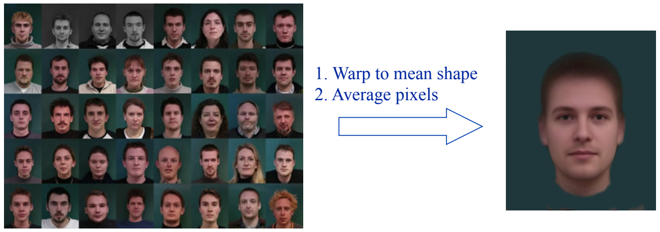
Volker Blanz & Thomas Vetter

MPI for Biological Cybernetics
Tübingen, Germany

More Fun with Faces



The Average Face



http://graphics.cs.cmu.edu/courses/15-463/2004_fall/www/handins/brh/final/

Slide credit: A. Efros

Subpopulation Means

Average female
(64 faces)

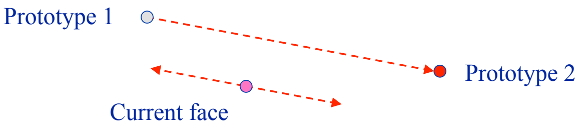
Average male
(32 faces)

- 250 control points per face manually specified

http://www.beautycheck.de

Manipulating Faces

- How can we make a face look more male/female, young/old, happy/sad, etc.?
  1. Obtain two prototypes spanning the desired axis (gender, age, expression, etc.) and find the difference vector between them
  2. Add scaled versions of this vector to a given image



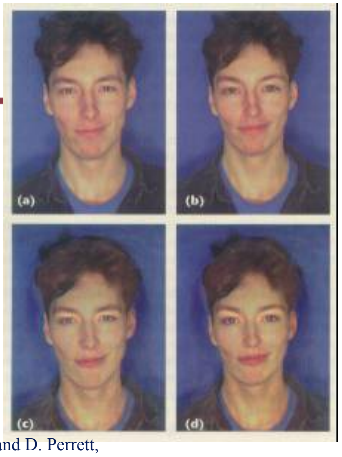
D. Rowland and D. Perrett,
"Manipulating Facial Appearance through Shape and Color," IEEE CG&A,
September 1995

Changing Gender

Deform shape or color of an input face in the direction of "more female"

● original shape

● color



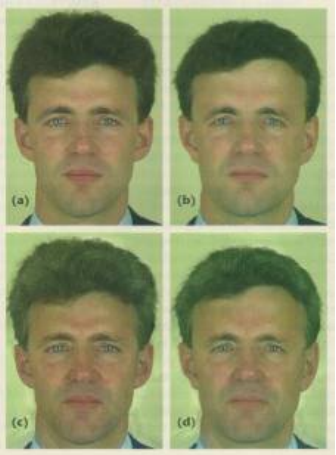
D. Rowland and D. Perrett,
"Manipulating Facial Appearance through Shape and Color," IEEE CG&A,
September 1995

Slide credit: A. Efros

Changing Age

Face becomes "rounder" and "more textured" and "grayer"

- original shape
- color



D. Rowland and D. Perrett, "Manipulating Facial Appearance through Shape and Color," IEEE CG&A, September 1995

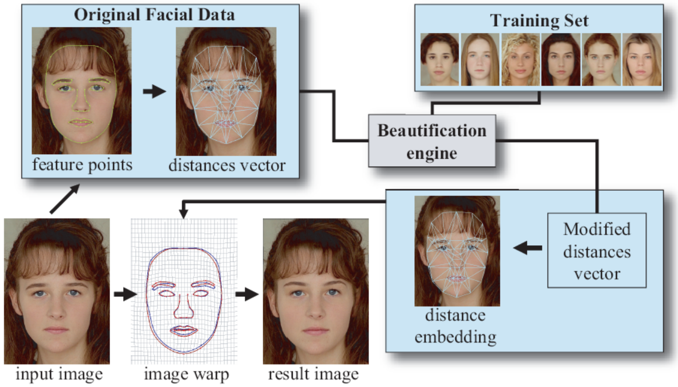
Slide credit: A. Efros

Data-Driven Enhancement of Facial Attractiveness

T. Leyvand, D. Cohen-Or, G. Dror, and D. Lischinski

SIGGRAPH 2008

System Overview



Original Facial Data

feature points

distances vector

Training Set

Beautification engine

Modified distances vector

distance embedding

input image

image warp

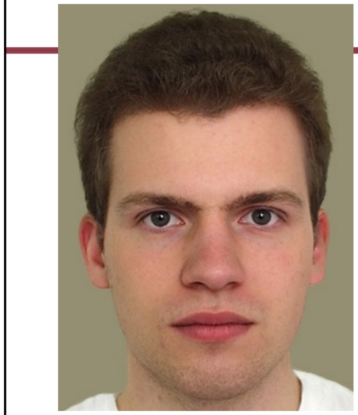
result image

Which Face is More Attractive?

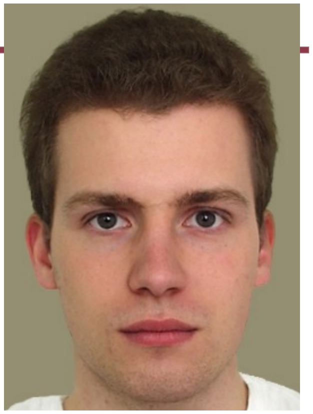
original

beautified

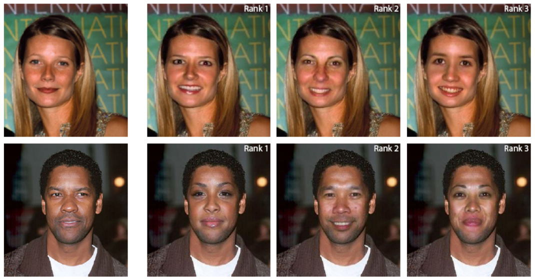
Which Face is More Attractive?



original



beautified



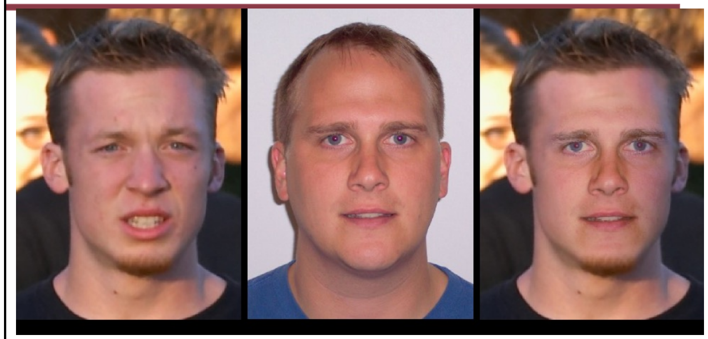
(a) Original photographs (b) After automatic face replacement

Face Swapping: Automatically Replacing Faces in Photographs

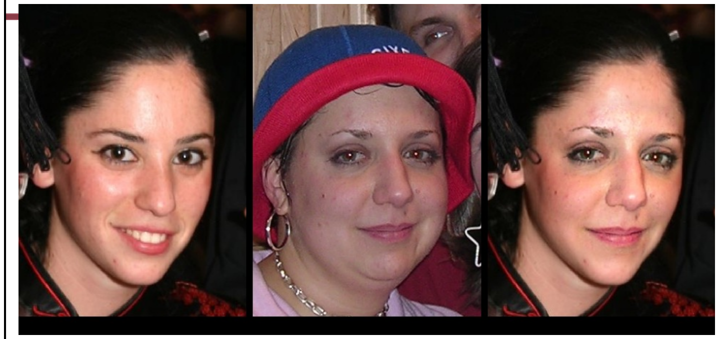
D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, S. K. Nayar

SIGGRAPH 2008

Face Replacement Results

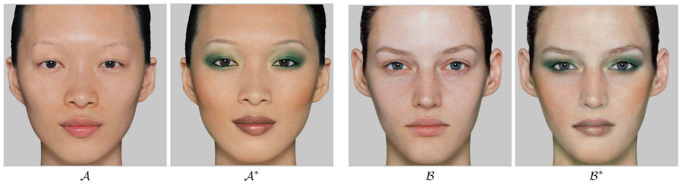


Face Replacement Results



More Results

Example-Based Cosmetic Transfer



W.-S. Tong, C.-K. Tang, M. Brown, Y.-Q. Xu
Pacific Graphics 2007