Warping & Morphing

CSE 590 Computational Photography
Tamara Berg
Announcements

Project Proposals

• Send email to sbu590@gmail, with 1-2 paragraph summary by Oct 5
• Presentations in class Oct 10 (5 min)
  – Idea
  – Data
  – Challenges
  – Timeline

HW3 up today, due Oct 16
  Please remember to submit your code via email!

Picture time

Office hours today short due to travel
Quiz

1) What is image warping? Give an example.

2) What are homogeneous coordinates and what are they useful for?

3) What kinds of transformations are included in affine transforms?

4) Give the high level steps to morph one image into another.

5) What extension to morphing did today’s paper propose?
Image Warping
Importance of shape and structure in evolution

Fig. 517. *Argyropelecus Olfersi*.  
Fig. 518. *Sternopyx diaphana*.  
Skulls of a human, a chimpanzee and a baboon and transformations between them.
Image Transformations

image filtering: change **range** of image

\[ g(x) = T(f(x)) \]

image warping: change **domain** of image

\[ g(x) = f(T(x)) \]
Image Transformations

image filtering: change **range** of image
\[ g(x) = T(f(x)) \]

image warping: change **domain** of image
\[ g(x) = f(T(x)) \]
Parametric (global) warping

Examples of parametric warps:

- translation
- rotation
- aspect
- affine
- perspective
- cylindrical
Parametric (global) warping

Transformation $T$ is a coordinate-changing machine:

$$p' = T(p)$$

What does it mean that $T$ is global?

- Is the same for any point $p$
- Can be described by just a few numbers (parameters)

Let's represent $T$ as a matrix:

$$p' = Mp$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$
Scaling a coordinate means multiplying each of its components by a scalar.

Uniform scaling means this scalar is the same for all components:
Non-uniform scaling: different scalars per component:

\[ X \times 2, \quad Y \times 0.5 \]
Scaling

Scaling operation:

\[ x' = ax \]
\[ y' = by \]

Or, in matrix form:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  a & 0 \\
  0 & b
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

What’s inverse of S?
2-D Rotation

\[ (x', y') = (x \cos(\theta) - y \sin(\theta), x \sin(\theta) + y \cos(\theta)) \]
2-D Rotation

\[(x', y')\]

\[(x, y)\]
2-D Rotation

\[ x = r \cos (\phi) \]
\[ y = r \sin (\phi) \]
\[ x' = r \cos (\phi + \theta) \]
\[ y' = r \sin (\phi + \theta) \]
2-D Rotation

\[ x = r \cos(\phi) \]
\[ y = r \sin(\phi) \]
\[ x' = r \cos(\phi + \theta) \]
\[ y' = r \sin(\phi + \theta) \]

Trig Identity…
\[ x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta) \]
\[ y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta) \]
2-D Rotation

\[
\begin{align*}
\theta &\quad (x', y') \\
\phi &\quad (x, y)
\end{align*}
\]

\[
x = r \cos (\phi)
\]
\[
y = r \sin (\phi)
\]
\[
x' = r \cos (\phi + \theta)
\]
\[
y' = r \sin (\phi + \theta)
\]

Trig Identity…
\[
x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)
\]
\[
y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)
\]

Substitute…
\[
x' = x \cos(\theta) - y \sin(\theta)
\]
\[
y' = x \sin(\theta) + y \cos(\theta)
\]
2-D Rotation

This is easy to capture in matrix form:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  \cos(\theta) & -\sin(\theta) \\
  \sin(\theta) & \cos(\theta)
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

What is the inverse transformation?

- Rotation by \(-\theta\)
- For rotation matrices \(R^{-1} = R^T\)
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Identity?

\[ x' = x \]
\[ y' = y \]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Identity?

\[ \begin{align*}
  x' &= x \\
  y' &= y
\end{align*} \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Identity?

\[ x' = x \]
\[ y' = y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

2D Scale?

\[ x' = s_x \times x \]
\[ y' = s_y \times y \]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Identity?

\[ x' = x \]
\[ y' = y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

2D Scale?

\[ x' = s_x \cdot x \]
\[ y' = s_y \cdot y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  s_x & 0 \\
  0 & s_y
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Rotate around (0,0)?

\[
\begin{align*}
x' &= \cos \Theta \cdot x - \sin \Theta \cdot y \\
y' &= \sin \Theta \cdot x + \cos \Theta \cdot y
\end{align*}
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

2D Shear?

\[
\begin{align*}
x' &= x + sh_x \cdot y \\
y' &= sh_y \cdot x + y
\end{align*}
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

\[ x' = -x \]
\[ y' = y \]
What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

\[ x' = -x \]
\[ y' = y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  -1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

\[ x' = -x \]
\[ y' = y \]

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= 
\begin{bmatrix}
  -1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

2D Mirror over (0,0)?

\[ x' = -x \]
\[ y' = -y \]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

\[
x' = -x \\
y' = y
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]

2D Mirror over (0,0)?

\[
x' = -x \\
y' = -y
\]

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Translation?

\[ x' = x + t_x \]
\[ y' = y + t_y \]
2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Translation?

\[ x' = x + t_x \]

\[ y' = y + t_y \]

NO!
All 2D Linear Transformations

Linear transformations are combinations of …

- Scale,
- Rotation,
- Shear, and
- Mirror

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\
y \end{bmatrix}
\]

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix} a & b & e & f & i & j \\ c & d & g & h & k & l \end{bmatrix} \begin{bmatrix} x \\
y \end{bmatrix}
\]
Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

\[ x' = x + t_x \]
\[ y' = y + t_y \]
Homogeneous Coordinates

**Homogeneous coordinates**
- represent coordinates in 2 dimensions with a 3-vector

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]
Homogeneous Coordinates

2D Points $\rightarrow$ Homogeneous Coordinates
   Append 1 to every 2D point: $(x, y) \rightarrow (x, y, 1)$
Homogeneous coordinates $\rightarrow$ 2D Points
   Divide by third coordinate $(x, y, w) \rightarrow (x/w, y/w)$

Special properties
   Scale invariant: $(x, y, w) = k \times (x, y, w)$
   $(x, y, 0)$ represents a point at infinity
   $(0, 0, 0)$ is not allowed
Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

\[ x' = x + t_x \]
\[ y' = y + t_y \]
Q: How can we represent translation as a 3x3 matrix?

\[ x' = x + t_x \]
\[ y' = y + t_y \]

A: Using the rightmost column:

\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\textit{Translation}
Translation

Example of translation

Homogeneous Coordinates

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

\( t_x = 2 \)
\( t_y = 1 \)
Translation

Example of translation

Homogeneous Coordinates

\[
\begin{bmatrix}
x' \\
y' \\
1
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix} =
\begin{bmatrix}
x + t_x \\
y + t_y \\
1
\end{bmatrix}
\]

\[t_x = 2\]
\[t_y = 1\]
Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Translate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  \cos \Theta & -\sin \Theta & 0 \\
  \sin \Theta & \cos \Theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Rotate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix}
= \begin{bmatrix}
  1 & \text{sh}_x & 0 \\
  \text{sh}_y & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Shear
Matrix Composition

Transformations can be combined by matrix multiplication

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} = \begin{pmatrix}
  1 & 0 & tx \\
  0 & 1 & ty \\
  0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
  \cos \Theta & -\sin \Theta & 0 \\
  \sin \Theta & \cos \Theta & 0 \\
  0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
  sx & 0 & 0 \\
  0 & sy & 0 \\
  0 & 0 & 1
\end{pmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

\[
p' = T(t_x,t_y) \quad R(\Theta) \quad S(s_x,s_y) \quad p
\]
Affine Transformations

Affine transformations are combinations of ...

- Linear transformations, and
- Translations

Properties of affine transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition
- Models change of basis

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix}
= \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]
Projective Transformations

Projective transformations …
- Affine transformations, and
- Projective warps

Properties of projective transformations:
- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix} = \begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{bmatrix}\begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]
Recovering Transformations

What if we know $f$ and $g$ and want to recover the transform $T$?

- willing to let user provide correspondences
  - How many do we need?
Translation: # correspondences?

How many correspondences needed for translation?
Translation: # correspondences?

How many correspondences needed for translation?
How many Degrees of Freedom?
Translation: # correspondences?

How many correspondences needed for translation?
How many Degrees of Freedom?
What is the transformation matrix?

\[ M = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix} \]
Euclidian: # correspondences?

How many correspondences needed for translation+rotation?
How many DOF?
Affine: # correspondences?

How many correspondences needed for affine?
How many DOF?
Affine: # correspondences?

How many correspondences needed for affine?

How many DOF?

\[
\begin{bmatrix}
x' \\
y' \\
w \\
\end{bmatrix}
= \begin{bmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
w \\
\end{bmatrix}
\]
How many correspondences needed for projective?
How many DOF?

\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix} =
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]
Example: warping triangles

Given two triangles: ABC and A’B’C’ in 2D (12 numbers)
Need to find transform T to transfer all pixels from one to the other.

How can we compute the transformation matrix:

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix}
    a & b & c \\
    d & e & f \\
    0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]
Example: warping triangles

Given two triangles: ABC and A’B’C’ in 2D (12 numbers)
Need to find transform T to transfer all pixels from one to the other.

How can we compute the transformation matrix:

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

cp2transform in matlab!
input – correspondences.
output – transformation.
Image warping

Given a coordinate transform \((x',y') = T(x,y)\) and a source image \(f(x,y)\), how do we compute a transformed image \(g(x',y') = f(T(x,y))\)?
Forward warping

Send each pixel \( f(x,y) \) to its corresponding location \( (x',y') = T(x,y) \) in the second image

Q: what if pixel lands “between” two pixels?
Forward warping

Send each pixel \( f(x,y) \) to its corresponding location \( (x’,y’) = T(x,y) \) in the second image

Q: what if pixel lands “between” two pixels?
A: distribute color among neighboring pixels \( (x’,y’) \)
   – Known as “splatting”
Image warping

Given a coordinate transform \((x', y') = T(x, y)\) and a source image \(f(x, y)\), how do we compute a transformed image \(g(x', y') = f(T(x, y))\)?

`imtransform` in MATLAB!

input – transform and source image
output – transformed image.
Warping in action

HP commercial
Warping in action

Target + Source
Warping in action

Target + Source = Result
Warping in action

User clicks on 4 points in target, 4 points in source.
User clicks on 4 points in target, 4 points in source.
Find best affine transformation between source subimg and target subimg.
Warping in action

User clicks on 4 points in target, 4 points in source. Find best affine transformation between source subimg and target subimg. Transform the source subimg accordingly.
Warping in action

User clicks on 4 points in target, 4 points in source.
Find best affine transformation between source subimg and target subimg.
Transform the source subimg accordingly.
Insert the transformed source subimg into the target image.
Warping in action

User clicks on 4 points in target, 4 points in source. Find best affine transformation between source subimg and target subimg. Transform the source subimg accordingly. Insert the transformed source subimg into the target image.
Warping in action

Target

Source

= Result
Exercise – Implement Image Warping

User clicks on 4 points in target, 4 points in source.

Find best affine transformation between source subimg and target subimg.

Transform the source subimg accordingly.

Insert the transformed source subimg into the target image.

Hints: use a mask image (warped the same way as the source), can assume the source region is a rectangle, Useful functions - ginput, cp2transform, imtransform.
Image Morphing - Women in Art video

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

http://www.youtube.com/watch?v=L0GKp-uvjO0
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 \( t \)

How do we know what the average object looks like?
• We haven’t a clue!
• But we can often fake something reasonable
  – Usually required user/artist input
Idea #1: Cross-Dissolve

Interpolate whole images:

\[ \text{Image}_{\text{halfway}} = (1-t) \times \text{Image}_1 + t \times \text{image}_2 \]

This is called **cross-dissolve** in film industry

But what if the images aren’t aligned?
Idea #2: Align, then cross-dissolve

Align first, then cross-dissolve

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

Morphing procedure:

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

for every $t$, 

for every $t$,
Local (non-parametric) Image Warping

Need to specify a more detailed warp function

- Global warps were functions of a few (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
Image Warping – non-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 do it?

How do we go from feature points to pixels?
1. Input correspondences at key 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!
Triangulations

A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.

There are an exponential number of triangulations of a point set.
An $O(n^3)$ Triangulation Algorithm

Repeat until impossible:

- Select two sites.
- If the edge connecting them does not intersect previous edges, keep it.
“Quality” Triangulations

Let $\alpha(T_i) = (\alpha_{i1}, \alpha_{i2}, ..., \alpha_{i3})$ be the vector of angles in the triangulation $T$ in increasing order:

- A triangulation $T_1$ is “better” than $T_2$ if the smallest angle of $T_1$ is larger than the smallest angle of $T_2$
- Delaunay triangulation is the “best” (maximizes the smallest angles)

![good triangulation](image1)
![bad triangulation](image2)

- good
- bad
Improving a Triangulation

In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.

If an edge flip improves the triangulation, the first edge is called “*illegal”*. 
Illegal Edges

An edge $pq$ is “illegal” iff one of its opposite vertices is inside the circle defined by the other three vertices (see Thale’s theorem)

- A triangle is Delaunay iff no other points are inside the circle through the triangle’s vertices
- The Delaunay triangulation is not unique if more than three nearby points are co-circular
- The Delaunay triangulation does not exist if three nearby points are colinear
Naïve Delaunay Algorithm

Start with an arbitrary triangulation. Flip any illegal edge until no more exist.
Could take a long time to terminate.
Delaunay Triangulation by Duality

Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.

• The DT may be constructed in $O(n \log n)$ time
• This is what Matlab’s `delaunay` function uses
1. Input correspondences at key 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!
Image Morphing

We know how to warp one image into the other, but how do we create a morphing sequence?

1. Create an intermediate shape (by interpolation)
2. Warp both images towards it
3. Cross-dissolve the colors in the newly warped images
Warp interpolation

How do we create an intermediate warp at time $t$?

- Assume $t = [0,1]$
- Simple linear interpolation of each feature pair
- $(1-t)p_1 + tp_0$ for corresponding features $p_0$ and $p_1$
Morphing & matting

Extract foreground first to avoid artifacts in the background

Slide by Durand and Freeman
Summary of morphing

1. Define corresponding points
2. Define triangulation on points
   • Use same triangulation for both images
3. For each $t = 0:\text{step}:1$
   a. Compute the average shape (weighted average of points)
   b. For each triangle in the average shape
      – Get the affine projection to the corresponding triangles in each image
      – For each pixel in the triangle, find the corresponding points in each image and set value to weighted average (optionally use interpolation)
   c. Save the image as the next frame of the sequence
View Morphing

Steven M. Seitz, Charles R. Dyer
View Morphing: Key Idea

Beier-Neely morph is NOT shape-preserving!

- distortions
- un-natural

A Shape-Distorting Morph
Source: Steven M. Seitz, Charles R. Dyer
View Morphing: Key Idea

View morphing uses 3D shape preserving morph!

- no distortions
- natural

A morph is **3D shape preserving** if the results of two different views represent new views of the same object.

A Shape-Distorting Morph
Source: Steven M. Seitz, Charles R. Dyer

Slide credit: Irwin Chiu Hau
Why do we care?

View morphing is efficient

Produces new views without
• 3D modeling
• Taking additional photos

View morphing creates impressive effects
• Camera motion
• Image morphing

trueSpace
Source: www.caligari.com

Slide credit: Irwin Chiu Hau
How to do View Morphing?

View morphing in three steps
Results

Facial view morphs
Source: Steven M. Seitz, Charles R. Dyer

Slide credit: Irwin Chiu Hau
Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and its reflection (right).
Results

Top: Image Morph
Bottom: View Morph