Computational Photography and Video:
Warping, Morphing and Mosaics

Prof. Marc Pollefeys
Today’s schedule

- Last week’s recap
- Warping
- Morphing
- Mosaics
Today’s schedule

- Last week’s recap
- Warping
- Mosaics
- Morphing
Exposure: shutter vs. aperture

• Trade-off motion blur vs. depth-of-field

(also gain/ISO sensitivity)
Imperfect lenses: aberrations, etc.
Sensors and color

![Diagram of sensor with color correction and wavelength chart](Image)

- **Imager** (354x292)
- **Column Amplifiers**
- **SRAM**
- **Control and Host Interface**
- **Color Interpolation**

**Wavelength in nanometers**

<table>
<thead>
<tr>
<th>360</th>
<th>435</th>
<th>500</th>
<th>520</th>
<th>565</th>
<th>590</th>
<th>625</th>
<th>740</th>
</tr>
</thead>
<tbody>
<tr>
<td>Violet</td>
<td>Blue</td>
<td>Cyan</td>
<td>Green</td>
<td>Yellow</td>
<td>Orange</td>
<td>Red</td>
<td></td>
</tr>
</tbody>
</table>

**ETH**
<table>
<thead>
<tr>
<th>Schedule</th>
<th>Computational Photography and Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 Feb</td>
<td>Introduction to Computational Photography</td>
</tr>
<tr>
<td>3 Mar</td>
<td>More on Camera, Sensors and Color</td>
</tr>
<tr>
<td>Assignment 1</td>
<td></td>
</tr>
<tr>
<td>10 Mar</td>
<td>Warping, Mosaics and Morphing</td>
</tr>
<tr>
<td>Assignment 2</td>
<td></td>
</tr>
<tr>
<td>17 Mar</td>
<td>Blending and compositing</td>
</tr>
<tr>
<td>Assignment 3</td>
<td></td>
</tr>
<tr>
<td>24 Mar</td>
<td>High-dynamic range</td>
</tr>
<tr>
<td>Assignment 4</td>
<td></td>
</tr>
<tr>
<td>31 Mar</td>
<td>TBD</td>
</tr>
<tr>
<td>Project proposals</td>
<td></td>
</tr>
<tr>
<td>7 Apr</td>
<td>Easter holiday – no classes</td>
</tr>
<tr>
<td>14 Apr</td>
<td>TBD</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
</tr>
<tr>
<td>21 Apr</td>
<td>TBD</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
</tr>
<tr>
<td>28 Apr</td>
<td>TBD</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
</tr>
<tr>
<td>5 May</td>
<td>TBD</td>
</tr>
<tr>
<td>Project update</td>
<td></td>
</tr>
<tr>
<td>12 May</td>
<td>TBD</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
</tr>
<tr>
<td>19 May</td>
<td>TBD</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
</tr>
<tr>
<td>26 May</td>
<td>TBD</td>
</tr>
<tr>
<td>Papers</td>
<td></td>
</tr>
<tr>
<td>2 June</td>
<td>Final project presentation</td>
</tr>
<tr>
<td>Final project presentation</td>
<td></td>
</tr>
</tbody>
</table>
Today’s schedule

- Last week’s recap
- Warping
- Mosaics
- Morphing
Image Warping

image filtering: change **range** of image

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

image warping: change **domain** of image

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

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

image warping: change \textit{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:

$$
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
$$
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar.
- *Uniform scaling* means this scalar is the same for all components:

![Diagram showing scaling](image)
Scaling

- **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?

scaling matrix S
2-D Rotation

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

• This is easy to capture in matrix form:

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

• Even though \(\sin(\theta)\) and \(\cos(\theta)\) are nonlinear functions of \(\theta\),
  
  – \(x'\) is a linear combination of \(x\) and \(y\)
  – \(y'\) is a linear combination of \(x\) and \(y\)

• 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?

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

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

2D Scale around (0,0)?

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

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

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

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

2D Mirror over (0,0)?

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

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  -1 & 0 \\
  0 & -1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
What types of transformations can be represented with a 2x2 matrix?

**2D Translation?**

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

**NO!**

Only linear 2D transformations can be represented with a 2x2 matrix.
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

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

- Add a 3rd coordinate to every 2D point
  - \((x, y, w)\) represents a point at location \((x/w, y/w)\)
  - \((x, y, 0)\) represents a point at infinity
  - \((0, 0, 0)\) is not allowed

Convenient coordinate system to represent many useful transformations
Homogeneous Coordinates

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

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

A: Using the rightmost column:

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

Translation
Translation

Example of translation

Homogeneous Coordinates

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

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

• Basic 2D transformations as 3x3 matrices

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

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

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear
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

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

- Transformations can be combined by matrix multiplication

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

\[p' = T(t_x, t_y) \quad R(\Theta) \quad S(s_x, s_y) \quad p\]
2D image transformations

These transformations are a nested set of groups

- Closed under composition and inverse is a member

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix</th>
<th># D.O.F.</th>
<th>Preserves:</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td>$\begin{bmatrix} I &amp; t \end{bmatrix}_{2\times3}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$\begin{bmatrix} R &amp; t \end{bmatrix}_{2\times3}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>similarity</td>
<td>$\begin{bmatrix} sR &amp; t \end{bmatrix}_{2\times3}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>affine</td>
<td>$\begin{bmatrix} A \end{bmatrix}_{2\times3}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projective</td>
<td>$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Image Warping in Biology

- D'Arcy Thompson
  - http://www-groups.dcs.st-and.ac.uk/~history/Miscellaneous/darcy.html

- Importance of shape and structure in evolution

Fig. 517. *Argyropelecus Olfersi.*
Fig. 518. *Sternoptyx diaphana.*

Skulls of a human, a chimpanzee and a baboon and transformations between them

Slide by Durand and Freeman
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?

$T(x,y)$
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?
Projective: # correspondences?

- How many correspondences needed for projective?
- How many DOF?
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”
   - Check out griddata in Matlab
Inverse warping

- Get each pixel \( g(x',y') \) from its corresponding location \( (x,y) = T^{-1}(x',y') \) in the first image

Q: What if pixel comes from "between" two pixels?
Inverse warping

- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image.

Q: what if pixel comes from “between” two pixels?

A: Interpolate color value from neighbors
- nearest neighbor, bilinear, Gaussian, bicubic
- Check out interp2 in Matlab
Forward vs. inverse warping

Q: which is better?

A: usually inverse—eliminates holes
   – however, it requires an invertible warp function—not always possible...
Today’s schedule

- Last week’s recap
- Warping
- Mosaics
- Morphing
Why Mosaic?

• Are you getting the whole picture?
  – Compact Camera FOV = 50 x 35°
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV          = 200 x 135°
Why Mosaic?

• Are you getting the whole picture?
  – Compact Camera FOV = 50 x 35°
  – Human FOV = 200 x 135°
Mosaics: stitching images together

virtual wide-angle camera
A pencil of rays contains all views

Can generate any synthetic camera view as long as it has the same center of projection!
How to do it?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - If there are more images, repeat

- ...but **wait**, why should this work at all?
  - What about the 3D geometry of the scene?
  - Why aren’t we using it?
Aligning images

Translations are not enough to align the images
The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*
Image reprojection

• Basic question
  – How to relate two images from the same camera center?
    • how to map a pixel from PP1 to PP2

Answer
  • Cast a ray through each pixel in PP1
  • Draw the pixel where that ray intersects PP2

But don’t we need to know the geometry of the two planes in respect to the eye?

Observation:
Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another
Back to Image Warping

Which t-form is the right one for warping PP1 into PP2?
e.g. translation, Euclidean, affine, projective

Translation
Affine
Perspective

2 unknowns
6 unknowns
8 unknowns
Homography

- A: Projective – mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines aren’t
  - but must preserve straight lines
  - same as: project, rotate, reproject

- called Homography

\[
\begin{bmatrix}
wx' \\
wy' \\
w
\end{bmatrix}
= \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & * \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
l
\end{bmatrix}
\]

To apply a homography \(H\)

- Compute \(p' = Hp\) (regular matrix multiply)
- Convert \(p'\) from homogeneous to image coordinates
Image warping with homographies

- Image plane in front
- Black area where no pixel maps to
To unwarp (rectify) an image

- Find the homography $H$ given a set of $p$ and $p'$ pairs
- How many correspondences are needed?
- Tricky to write $H$ analytically, but we can solve for it!
- Find such $H$ that “best” transforms points $p$ into $p'$
- Use least-squares!
Computing a homography

Stack matrix representing equations for 4 or more points

Solve through SVD, least-square solution is given by last right singular vector

(i.e. smallest singular value)

Numerically better if image coordinates scaled to [-1,1]
Panoramas

1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. Blend
changing camera center

Does it still work?

PP1

PP2

synthetic PP
Planar scene (or far away)

- PP3 is a projection plane of both centers of projection, so we are OK!
- This is how big aerial photographs are made
Planar mosaic
Today’s schedule

- Last week’s recap
- Warping
- Mosaics
- Morphing
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
Averaging Points

What’s the average of P and Q?

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

- $P + 0.5v = P + 0.5(Q - P) = 0.5P + 0.5Q$
- $P + 1.5v = P + 1.5(Q - P) = -0.5P + 1.5Q$ (extrapolation)

• 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.
Idea #1: Cross-Dissolve

- Interpolate whole images:
  \[ \text{Image}_{\text{halfway}} = (1-t) \cdot \text{Image}_1 + t \cdot \text{Image}_2 \]
- This is called **cross-dissolve** in film industry

- But what is the images are not 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
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 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 (use backward warping with interpolation)
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 \textit{spline control points}
    - \textit{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?
Triangular Mesh

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!
   - Just like texture mapping
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
  – $(1-t)\cdot p_1 + t\cdot p_0$ for corresponding features $p_0$ and $p_1$
Morphing & matting

- Extract foreground first to avoid artifacts in the background
Women in Art video

- [http://youtube.com/watch?v=nUDIoN-_Hxs](http://youtube.com/watch?v=nUDIoN-_Hxs)
Problem with morphing

- So far, we have performed linear interpolation of feature point positions
- But what happens if we try to morph between two views of the same object?

Figure 2: A Shape-Distorting Morph. Linearly interpolating two perspective views of a clock (far left and far right) causes a geometric bending effect in the in-between images. The dashed line shows the linear path of one feature during the course of the transformation. This example is indicative of the types of distortions that can arise with image morphing techniques.
View morphing

- Seitz & Dyer
  

- Interpolation consistent with 3D view interpolation

Figure 1: View morphing between two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.
Main trick

- Prewarp with a homography to "pre-align" images
- So that the two views are parallel
  - Because linear interpolation works when views are parallel

Figure 4: View Morphing in Three Steps. (1) Original images $I_0$ and $I_1$ are prewarped to form parallel views $\hat{I}_0$ and $\hat{I}_1$. (2) $\hat{I}_s$ is produced by morphing (interpolating) the prewarped images. (3) $\hat{I}_s$ is postwarped to form $I_s$. 
Figure 6: View Morphing Procedure: A set of features (yellow lines) is selected in original images $I_0$ and $I_1$. Using these features, the images are automatically prewarped to produce $\hat{I}_0$ and $\hat{I}_1$. The prewarped images are morphed to create a sequence of in-between images, the middle of which, $\hat{I}_{0.5}$, is shown at top-center. $\hat{I}_{0.5}$ is interactively postwarped by selecting a quadrilateral region (marked red) and specifying its desired configuration, $Q_{0.5}$, in $I_{0.5}$. The postwarps for other in-between images are determined by interpolating the quadrilaterals (bottom).
Figure 10: Image Morphing Versus View Morphing. Top: image morph between two views of a helicopter toy causes the in-between images to contract and bend. Bottom: view morph between the same two views results in a physically consistent morph. In this example the image morph also results in an extraneous hole between the blade and the stick. Holes can appear in view morphs as well.
Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and its reflection (right).
Figure 7: Facial View Morphs. Top: morph between two views of the same person. Bottom: morph between views of two different people. In each case, view morphing captures the change in facial pose between original images $I_0$ and $I_1$, conveying a natural 3D rotation.
Next week

- Image pyramids and graph-cuts