Image Warping and Morphing

Frédo Durand
Bill Freeman
MIT - EECS
Intelligent design & image warping

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

- Importance of shape and structure in evolution
Important scientific question

• How to turn Dr. Jekyll into Mr. Hyde?
• How to turn a man into a werewolf?

• Powerpoint cross-fading?
Important scientific question

- How to turn Dr. Jekyll into Mr. Hyde?
- How to turn a man into a werewolf?

- Powerpoint cross-fading?

- or

- Image Warping and Morphing?

From An American Werewolf in London
Digression: old metamorphoses

- Unless I’m mistaken, both employ the trick of making already-applied makeup turn visible via changes in the color of the lighting, something that works only in black-and-white cinematography. It’s an interesting alternative to the more familiar Wolf Man time-lapse dissolves. This technique was used to great effect on Fredric March in Rouben Mamoulian’s 1932 film of *Dr. Jekyll and Mr. Hyde*, although Spencer Tracy eschewed extreme makeup for his 1941 portrayal.
• *Jekyll & Hide 1934:*
  – 35:13
  – ch18 1:06:45
  – ch19 1:17:50

• *Jekyll & Hide 1941:*
  – ch20 1:25:13
Averaging images

• Cross-fading
  – Pretty much the compositing equation
    \[ C = t \ F \ (1-t) \ B \]
Averaging vectors

\[ V = t \mathbf{P} + (1-t) \mathbf{Q} \]
Warping & Morphing combine both

- For each pixel
  - Transform its location like a vector
  - Then linearly interpolate like an image
Morphing

• Input: two images $I_0$ and $I_N$

• Expected output: image sequence $I_i$, with $i \in 1..N-1$

• 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 $P^t_i = (1 - t) P^0_i + t P^1_i$
  - 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
Warping
Warping

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

No prairie dogs were armed when creating this image
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 – 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

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
   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately from source to destination
Problems with triangulation morphing

- Not very continuous
  - only $C^0$

- Folding problems

Fig. L. Darsa
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
Interpolation in 1D

• We are looking for a function $f$

• We have $N$ data points: $x_i, y_i$
  – Scattered: spacing between $x_i$ is non-uniform

• We want $f$ so that
  – For each $i$, $f(x_i) = y_i$
  – $f$ is smooth

• Depending on notion of smoothness, different $f$
Radial Basis Functions (RBF)

- Place a smooth kernel $R$ centered on each data point $x_i$

$$f(z) = \sum \alpha_i R(z, x_i)$$
Radial Basis Functions (RBF)

• Place a smooth kernel $R$ centered on each data point $x_i$

• $f(z) = \Sigma \alpha_i R(z, x_i)$

• Find weights $\alpha_i$ to make sure we interpolate the data for each $i$, $f(x_i)=y_i$
Kernel

- Many choices
- e.g. inverse multiquadric

\[
R(z, x_i) = \frac{1}{\sqrt{c + \|z - x_i\|^2}}
\]

- where \( c \) controls falloff
- Lazy way: set \( c \) to an arbitrary constant (pset 4)
- Smarter way: \( c \) is different for each kernel. For each \( x_i \), set \( c \) as the squared distance to the closest other \( x_j \)
Enforcing interpolation

- \( f(z) = \sum \alpha_i R(z, x_i) \)
- \( N \) equations
  - for each \( j \), \( f(x_j) = y_j \)
  - \( \sum \alpha_i R(x_j, x_i) = y_j \)
- \( N \) unknowns \( \alpha_i \)
- Just inverse the matrix
Important note

- \( f(z) = \sum \alpha_i R(z, x_i) \)
  for each \( j, \sum \alpha_i R(x_j, x_i) = y_j \)

- Note that
  the influence of each function is non-zero everywhere
  at a data point, the value of the other bases is not zero

- In contrast to e.g. various interpolation splines
Variations of RBF

• Lots of possible kernels
  – Gaussians $e^{-r^2/2\sigma}$
  – Thin-plate splines $r^2 \log r$

• Sometimes add a global polynomial term
Recap: 1D scattered data interpolation

- **Sparse input/output pairs** $x_i, y_i$
  - non-uniformly sampled
- **RBFs (Radial Basis Functions)**
  - Weighted sum of kernels $R$ centered on data points $x_i$
    \[
    f(z) = \sum \alpha_i R(z, x_i)
    \]
  - Compute the weights $\alpha_i$ by enforcing interpolation
    \[f(x_j)=y_j\]
  - Simple linear system
Recap: 1D scattered data interpolation

- Sparse input/output pairs $x_i, y_i$
  - non-uniformly sampled
- RBFs (Radial Basis Functions)
  - Weighted sum of kernels $R$ centered on data points $x_i$
    \[ f(z) = \sum \alpha_i R(z, x_i) \]
  - Compute the weights $\alpha_i$ by enforcing interpolation
    \[ f(x_j) = y_j \]
  - Simple linear system

QUESTION?
RBF for warping: 2D case

- Instead of $f: \mathbb{R} \rightarrow \mathbb{R}$, we now deal with $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$
  - For each 2D point, $f$ gives us another 2D warped point
- We have N data points
  - Pairs of input 2D vector, output 2D vector
  - Careful: $x_i$ is now a 2D vector, so is $y_i$
  - Don't be confused with coordinates $(x,y)$
- Place 2D kernels at each data point
- The weights $\alpha_i$ are now 2D vectors
- Solve a linear system of $2N$ equations and $2N$ unknowns
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
Example
Example

- Fold problems
  - Oh well…
1D equivalent of folds

- There is no guarantee that our 1D RBF is monotonic
- Yes, it means that the notion of inverse of the warp is questionable.

result (remember, inverse warp)
Hardcore Photoshop for portrait
Figure 9.37
Selecting the entire left side of the image avoids potential artifacts.

Figure 9.38
Dragging a Free Transform handle to narrow the selected area.

Figure 9.39
The Liquify filter's Warp tool pushes pixels forward as you drag.
Step Three:
Get the Push Left tool from the Toolbar (as shown here). It was called the Shift Pixels tool in Photoshop 6 and 7, but Adobe realized that you were getting used to the name, so they changed it, just to keep you off balance.

Step Four:
Choose a relatively small brush size (like the one shown here) using the Brush Size field near the top-right of the Liquify dialog. With it, paint a downward stroke starting just above and outside the love handle and continuing downward. The pixels shifts back in toward the body, removing the love handle as you paint. (Note: If you need to remove love handles on the left side of the body, paint upward rather than downward. Why? That's just the way it works.) When you click OK, the love handle repair is complete.
Morphing
Feature correspondences

- The feature locations will be our $x_i$
- Yes, in this example, the number of features is excessive
Interpolate feature location

- Provides the $y_i$
Warp each image to intermediate location

Two different warps: Same target location, different source location

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

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

Here we show $t=0.5$ (the $y_i$ are in the middle)
Warp each image to intermediate location
Interpolate colors linearly

Interpolation weight are a function of time:
\[
C = (1-t)f^0_t(I_0) + tf^1_t(I_1)
\]
Recap

• For each intermediate frame $I_t$
  – Interpolate feature 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
Movie time

- MJ BW: 23:01
- Willow:
Resampling
The sampling problem

- Parts are magnified
- Parts are minified
- Sometimes anisotropic

- Same problem for 3D texture mapping
Intuition

Plain lookup is bad
(But good news: that's all we ask for pset 3)

• In magnified regions, not smooth enough
• In minified regions, it creates aliasing

What we want

In magnified regions, smooth interpolation
In minified regions, take the average

We need good signal processing framework to do this
Similar case: texture aliasing

- *Aliasing* is the under-sampling of a signal, and it's especially noticeable during animation.
Fundamentals of Texture Mapping and Image Warping

Master's Thesis
under the direction of Carlo Séquin

Paul S. Heckbert

Dept. of Electrical Engineering and Computer Science
University of California, Berkeley, CA 94720

©1989 Paul S. Heckbert

June 17, 1989
Resampling

2D texture space

warp due to perspective

2D screen space
Notations

- **Input signal** $f(u)$
- **Forward mapping** (texture-to-screen) $x = m(u)$
- **Output signal** $g(x)$

Warning: I sloppily changed my notations: $f$ is signal, warp is $m$
Resampling

- What do we need to do?
Resampling

1. Reconstruct the continuous signal from the discrete input signal
2. Warp the domain of the continuous signal
3. Prefilter the warped continuous signal
4. Sample this signal
Figure 3.11: The four steps of ideal resampling: reconstruction, warp, prefiltter, and sample.
Resampling: progression

- Discrete input texture $f(u)$ for integer $u$
- Reconstructed input texture
  \[ f_c(u) = f(u) \otimes r(u) = \sum f(k) r(u-k) \]
- Warped texture $g_c(x) = f_c(m^{-1}(x))$
- Band-limited output
  \[ g'_c(x) = g_c(x) h(x) = \int g_c(t) h(x-t) \, dt \]
- Discrete output $g(x) = g'(x) \otimes i(x)$
Resampling

source space

discrete input

reconstruct

reconstructed input

resampling with prefiltering
[Heckbert 89]

destination space

discrete output

sample

continuous output
Put it together

- Discrete input texture \( f(u) \) for integer \( u \)
- Reconstructed input texture \( f_c(u) = f(u) \otimes r(u) = \sum f(k) r(u-k) \)
- Warped texture \( g_c(x) = f_c(m^{-1}(x)) \)
- Band-limited output \( g'_c(x) = g_c(x) h(x) = \int g_c(t) h(x-t) \, dt \)
- Discrete output \( g(x) = g'(x) \otimes i(x) \)
- \( g(x) = g'_c(x) \)
Put it together

- Discrete input texture $f(u)$ for integer $u$
- Reconstructed input texture $f_c(u) = f(u) \otimes r(u) = \sum f(k) r(u-k)$
- Warped texture $g_c(x) = f_c(m^{-1}(x))$
- Band-limited output $g'_c(x) = g_c(x) h(x) = \int g_c(t) h(x-t) \, dt$
- Discrete output $g(x) = g'(x) \otimes i(x)$

\[ g(x) = g'_c(x) \]

\[ = \int g_c(t) h(x-t) \, dt \]
\[ = \int f_c(m^{-1}(t)) h(x-t) \, dt \]
\[ = \int h(x-t) \sum f(k) r(m^{-1}(t)-k) \, dt \]
\[ = \sum f(k) \rho(x, k) \]

- Where $\rho(x,k) = \int h(x - t) r(m^{-1}(t) - k) \, dt$
Resampling – convolution view

• Ignoring normalization

\[ g(x) = \sum_i f(x_i) r_i(m^{-1}(x)) \otimes h(x) \]

• The image space resampling filter combines a warped reconstruction filter and a low-pass filter
Resampling

- Resampling reconstruction kernels
- Reconstructed input
- Reconstruction kernels
- Irregular spacing
- Color
Resampling

1. **Source Space**
2. **Warp**
3. **Filter**
4. **Sample**

Source Space

Destination Space

Destination Space

Destination Space
Resampling

1. Source Space
2. Warp
3. Filter
4. Sample

- Low-pass filter
- Convolution
- Warped reconstruction kernel
- Sum of resampling filters
- Resampling filters
Resampling – convolution view

- Ignoring normalization

\[ g(x) = \sum_i f(x_i) r_i(m^{-1}(x)) \otimes h(x) \]

- The image space resampling filter combines a warped reconstruction filter and a low-pass filter

- This is great, but how do we warp reconstruction filters?
Resampling

- Use local affine approximation of warp
- Elliptical Gaussian kernels [Heckbert 89]
  - Closed under affine mappings and convolution

\[
g(x) = \sum_i c_i r_i(\tilde{m}_i^{-1}(x)) \otimes h(x)
\]

\[
= \sum_i c_i G_i(x)
\]

Gaussian resampling kernel
(EWA resampling kernel)
Resampling filter

• Depends on local warp
• For perspective, approximated by local affine at center of kernel
• Not bad approximation because filter small at periphery
Resampling Filter

warped reconstruction kernel  low-pass filter  resampling filter

minification

magnification
EWA resampling
Image Quality Comparison

- Trilinear mipmapping

![Image Comparison](image.png)

EWA  trilinear mipmapping
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
Video

- Lots of manual work
View morphing
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.
Extensions
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.
The Morphable face model

- Again, assuming that we have $m$ such vector pairs in full correspondence, we can form new shapes $S_{model}$ and new appearances $T_{model}$ as:

$$S_{model} = \sum_{i=1}^{m} a_i S_i$$
$$T_{model} = \sum_{i=1}^{m} b_i T_i$$

- 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
- $$(\alpha_1, \alpha_2, \ldots, \alpha_m)^T\text{ and } (\beta_1, \beta_2, \ldots, \beta_m)^T$$!
Shape Vector

Provides alignment!

= 43

Slide Alyosha Efros
Deviations from the mean

\[ \Delta x = x - \bar{x} \]

\[ x = \bar{x} + 1.7 \]
Subpopulation means

• **Examples:**
  – Happy faces
  – Young faces
  – Asian faces
  – Etc.
  – Sunny days
  – Rainy days
  – Etc.
  – Etc.
The average face

- [http://www.uni-regensburg.de/Fakultaeten/phil_Fak_II/Psychologie/Psy_II/beautycheck/english/index.htm](http://www.uni-regensburg.de/Fakultaeten/phil_Fak_II/Psychologie/Psy_II/beautycheck/english/index.htm)

On the left: the “real” Miss Germany 2002 (= Miss Berlin) and on the right: the “virtual” Miss Germany, which was computed by blending together all contestants of the final round and was rated as being much more attractive.
show SIGGRAPH video
Manipulating Facial Appearance through Shape and Color

Duncan A. Rowland and David I. Perrett
St Andrews University
IEEE CG&A, September 1995
Morphable face models


- http://www.kyb.mpg.de/publication.html?user=volker
EigenFaces

First popular use of PCA on images was for modeling and recognition of faces [Kirby and Sirovich, 1990, Turk and Pentland, 1991]

- Collect a face ensemble
- Normalize for contrast, scale, & orientation.
- Remove backgrounds
- Apply PCA & choose the first $N$ eigen-images that account for most of the variance of the data.
Figure-centric averages


**Averages**: Hundreds of images containing a person are averaged to reveal regularities in the intensity patterns across all the images.
Jason Salavon

Homes for Sale

109 Homes for Sale, Seattle/Tacoma
117 Homes for Sale, Chicagoland
124 Homes for Sale, The 5 Boroughs
121 Homes for Sale, LA/Orange County
114 Homes for Sale, Dallas/Ft. Worth Metroplex
112 Homes for Sale, Miami-Dade County

More at: http://www.salavon.com/
“100 Special Moments” by Jason Salavon

Why blurry?

Little Leaguer

Kids with Santa

The Graduate

Newlyweds

Slide Alyosha Efros
3D morphing

- Feature-Based Volume Metamorphosis Lerios, Garfinkle, and Levoy.
3D morphing

- Feature-Based Volume Metamorphosis Lerios, Garfinkle, and Levoy.
Automatic morphing

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
- Scattered data interpolation
References
Refs

- [Portal ACM](http://portal.acm.org/citation.cfm?id=134003&coll=GUIDE&dl=GUIDE&CFID=72901489&CFTOKEN=24335444)
- [Visgraf IMPA](http://www.visgraf.impa.br/cgi-bin/morphQuery.cgi?output=html)
- [CG TUWIEEN](http://www.cg.tuwien.ac.at/research/ca/mrm/)
- [Impa Morph](http://w3.impa.br/~morph/sites.html)
- [SIGCourse Slides](http://w3.impa.br/~morph/sig-course/slides.html)
- [OwlNet Rice](http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/morph.html)
- [FMRI Oxford](http://www.fmrib.ox.ac.uk/~yongyue/thinplate.html)
- [UO Guelph](http://www.uoguelph.ca/~mwirth/PHD_Chapter4.pdf)
Software

- http://www.morpheussoftware.net/
- http://www.debugmode.com/winnmorph/
- http://www.stoik.com/products/morphman/mm1_main.htm
- http://www.creativecow.net/articles/zwar_chris/morph/index.html
- http://meesoftware.logicnet.dk/SmartMorph/
- http://www.asahi-net.or.jp/~FX6M-FJMY/mop00e.html
- http://morphing-software-review.toptenreviews.com/
Next time: Panoramas