Image warping/morphing

Digital Visual Effects

Yung-Yu Chuang

with slides by Richard Szeliski, Steve Seitz, Tom Funkhouser and Alexei Efros

Image formation

Sampling and quantization
What is an image

- We can think of an image as a function, \( f: \mathbb{R}^2 \rightarrow \mathbb{R} \):
  - \( f(x, y) \) gives the intensity at position \((x, y)\)
  - defined over a rectangle, with a finite range:
    - \( f: [a,b] \times [c,d] \rightarrow [0,1] \)

- A color image
  
  \[
  f(x, y) = \begin{bmatrix}
  r(x, y) \\
  g(x, y) \\
  b(x, y)
  \end{bmatrix}
  \]

A digital image

- We usually operate on digital (discrete) images:
  - Sample the 2D space on a regular grid
  - Quantize each sample (round to nearest integer)
- If our samples are \( D \) apart, we can write this as:
  
  \[
  f[i,j] = \text{Quantize}\{ f(iD, jD) \}
  \]
- The image can now be represented as a matrix of integer values

<table>
<thead>
<tr>
<th>( i )</th>
<th>( j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>79</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>176</td>
<td>135</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>99</td>
</tr>
</tbody>
</table>

Image warping

image filtering: change range of image

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

image warping: change domain of image

\[ g(x) = f(h(x)) \]
Parametric (global) warping

Examples of parametric warps:
- translation
- rotation
- aspect
- affine
- perspective
- cylindrical

Scaling

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

\[
\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 2x \\ 2y \end{bmatrix}
\]

- Non-uniform scaling: different scalars per component:

\[
\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} 2x \\ 0.5y \end{bmatrix}
\]

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)

Represent \( T \) as a matrix: \( p' = M p \)

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}
\]
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

- This is easy to capture in matrix form:
  \[
  \begin{bmatrix}
  x' \\
  y'
  \end{bmatrix}
  =
  \begin{bmatrix}
  \cos(\theta) & -\sin(\theta) \\
  \sin(\theta) & \cos(\theta)
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]
  \[ R \]
- Even though \( \sin(\theta) \) and \( \cos(\theta) \) are nonlinear to \( \theta \),
  - \( x' \) 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, \( \det(R) = 1 \) so \( R^{-1} = R^T \)

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

  **2D Rotate around (0,0)?**
  \[
  x' = \cos(\theta) \times x - \sin(\theta) \times y \\
  y' = \sin(\theta) \times x + \cos(\theta) \times y
  \]
  \[
  \begin{bmatrix}
  x' \\
  y'
  \end{bmatrix}
  =
  \begin{bmatrix}
  \cos(\theta) & -\sin(\theta) \\
  \sin(\theta) & \cos(\theta)
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]

  **2D Shear?**
  \[
  x' = x + s_{xy} \times y \\
  y' = s_{yx} \times x + y
  \]
  \[
  \begin{bmatrix}
  x' \\
  y'
  \end{bmatrix}
  =
  \begin{bmatrix}
  1 & s_{xy} \\
  s_{yx} & 1
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  \]
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)?

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

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

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

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

Translation

- Example of translation

Homogeneous Coordinates

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

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

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

**Image warping**

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

**Forward warping**

- Send each pixel \( I(x) \) to its corresponding location \( x' = T(x) \) in \( I'(x') \)
Forward warping

\[
\text{fwarp}(I, I', T) \\
\{ \\
\quad \text{for } (y=0; y<I.\text{height}; y++) \\
\quad \quad \text{for } (x=0; x<I.\text{width}; x++) \\
\quad \quad \quad (x', y') = T(x, y); \\
\quad \quad \quad I'(x', y') = I(x, y); \\
\quad \} \\
\}
\]

- Send each pixel \( I(x) \) to its corresponding location \( x' = T(x) \) in \( I'(x') \)
- What if pixel lands "between" two pixels?
- Will be there holes?
- Answer: add "contribution" to several pixels, normalize later (splatting)

Inverse warping

- Get each pixel \( I'(x') \) from its corresponding location \( x = T^{-1}(x') \) in \( I(x) \)
Inverse warping

\[ \text{iwarp}(I, I', T) \]
\[
\{ 
\text{for (y=0; y<I'.height; y++)}
\text{for (x=0; x<I'.width; x++)} \}
\]
\[
(x, y) = T^{-1}(x', y');
\]
\[
I'(x', y') = I(x, y);
\]

Inverse warping

- Get each pixel \( I'(x') \) from its corresponding location \( x = T^{-1}(x') \) in \( I(x) \)

- What if pixel comes from “between” two pixels?
- Answer: resample color value from interpolated (prefiltered) source image

Sampling

\[ \text{Sampling}(\text{band limited}) \]
The reconstructed function is obtained by interpolating among the samples in some manner.

Reconstruction (interpolation)

- Computed weighted sum of pixel neighborhood; output is weighted average of input, where weights are normalized values of filter kernel k
  \[ p = \frac{\sum q_i k(q_i)}{\sum q_i} \]
  
  ```
  color=0;
  weights=0;
  for all q’s dist < width
    d = dist(p, q);
    w = kernel(d);
    color += w*q.color;
    weights += w;
  p.Color = color/weights;
  ```

Reconstruction (interpolation)

- Possible reconstruction filters (kernels):
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc (optimal reconstruction)
Bilinear interpolation (triangle filter)

- A simple method for resampling images

\[ f(x, y) = (1 - a)(1 - b) f[i, j] + a(1 - b) f[i + 1, j] + ab f[i + 1, j + 1] + (1 - a)b f[i, j + 1] \]

Non-parametric image warping

- Specify a more detailed warp function
- Splines, meshes, optical flow (per-pixel motion)

Non-parametric image warping

- Mappings implied by correspondences
- Inverse warping

\[ P = w_A A + w_B B + w_C C \]

\[ P' = w_A A' + w_B B' + w_C C' \]

Barycentric coordinate
Barycentric coordinates

\[ P = t_1 A_1 + t_2 A_2 + t_3 A_3 \]
\[ t_1 + t_2 + t_3 = 1 \]

Non-parametric image warping

\[ P = w_A A + w_B B + w_C C \]
\[ P' = w_A A' + w_B B' + w_C C' \]

Barycentric coordinate

Non-parametric image warping

- Gaussian: \( \rho(r) = e^{-r^2} \)
- Thin plate spline: \( \rho(r) = r^2 \log(r) \)

\[ \Delta P = \frac{1}{K} \sum_i k_x (P') \Delta X_i \]

Demo

- Warping is a useful operation for mosaics, video matching, view interpolation and so on.
Image morphing

The goal is to synthesize a fluid transformation from one image to another.

Cross dissolving is a common transition between cuts, but it is not good for morphing because of the ghosting effects.

Artifacts of cross-dissolving

Why ghosting?
• Morphing = warping + cross-dissolving
  - shape (geometric)
  - color (photometric)

http://www.salavon.com/
**Image morphing**

- **image #1**
- **cross-dissolving**
- **image #2**

**Face averaging by morphing**

- create a morphing sequence: for each time t
  1. Create an intermediate warping field (by interpolation)
  2. Warp both images towards it
  3. Cross-dissolve the colors in the newly warped images

- average faces

**Morphing sequence**
An ideal example (in 2004)

Warp specification (mesh warping)

- How can we specify the warp?
  1. Specify corresponding spline control points
  
  interpolate to a complete warping function

  easy to implement, but less expressive

Warp specification

- How can we specify the warp
  2. Specify corresponding points

  interpolate to a complete warping function
Solution: convert to mesh warping

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

Warp specification (field warping)

- How can we specify the warp?
  3. Specify corresponding vectors
     - interpolate to a complete warping function
     - The Beier & Neely Algorithm

Beier&Neely (SIGGRAPH 1992)

- Single line-pair PQ to P’Q’:

\[
\begin{align*}
  u &= \frac{(X-P) \cdot (Q-P)}{\|Q-P\|^2} \\
  v &= \frac{(X-P) \cdot \text{Perpendicular}(Q-P)}{\|Q-P\|} \\
  X' &= P' + u \cdot (Q' - P') + v \cdot \text{Perpendicular}(Q' - P') / \|Q' - P'\|
\end{align*}
\]

Algorithm (single line-pair)

- For each X in the destination image:
  1. Find the corresponding u,v
  2. Find X’ in the source image for that u,v
  3. \( \text{destinationImage}(X) = \text{sourceImage}(X') \)

- Examples:
  - Affine transformation
Multiple Lines

\[ D_i = X'_i - X_i \]

\[ \text{weight}[i] = \left( \frac{\text{length}[i]^p}{a + \text{dist}[i]} \right)^b \]

Length = length of the line segment, 
Dist = distance to line segment

The influence of \( a, p, b \). The same as the average of \( X'_i \).

Full Algorithm

\[
\text{WarpImage(SourceImage, L'[...], L[...])}
\]

\[
\begin{align*}
&\text{begin} \\
&\hspace{1cm} \text{foreach destination pixel X do} \\
&\hspace{2cm} \text{XSum} = (0,0) \\
&\hspace{2cm} \text{WeightSum} = 0 \\
&\hspace{2cm} \text{foreach line L[i] in destination do} \\
&\hspace{3cm} \text{X}'[i] = \text{X transformed by (L[i], L'[i])} \\
&\hspace{3cm} \text{weight}[i] = \text{weight assigned to X[i]} \\
&\hspace{3cm} \text{XSum} = \text{Xsum} + \text{X}'[i] \times \text{weight}[i] \\
&\hspace{3cm} \text{WeightSum} += \text{weight}[i] \\
&\hspace{2cm} \text{end} \\
&\hspace{1cm} \text{X'} = \text{Xsum}/\text{WeightSum} \\
&\hspace{1cm} \text{DestinationImage(X)} = \text{SourceImage(X')} \\
&\text{end} \\
&\text{return Destination} \\
\end{align*}
\]

Resulting warp

Comparison to mesh morphing

- Pros: more expressive
- Cons: speed and control
**Warp interpolation**

- How do we create an intermediate warp at time $t$?
  - linear interpolation for line end-points
  - But, a line rotating 180 degrees will become 0 length in the middle
  - One solution is to interpolate line mid-point and orientation angle

$$
\begin{align*}
 \text{Warp}_{0} & = \text{WarpImage} ( \text{Image}_0, L_0[...], L[...]) \\
 \text{Warp}_{1} & = \text{WarpImage} ( \text{Image}_1, L_1[...], L[...]) \\
 \text{FinalImage}(p) & = (1-t) \text{Warp}_{0}(p) + t \text{Warp}_{1}(p)
\end{align*}
$$

**Animation**

GenerateAnimation(Image$_0$, $L_0[...], Image_1, L_1[...]$)
begin
  foreach intermediate frame time $t$ do
    for $i=1$ to number of line-pairs do
      $L[i]$ = line $t$-th of the way from $L_0[i]$ to $L_1[i]$.
    end
    $\text{Warp}_0 = \text{WarpImage} ( \text{Image}_0, L_0[...], L[...])$
    $\text{Warp}_1 = \text{WarpImage} ( \text{Image}_1, L_1[...], L[...])$
    foreach pixel $p$ in FinalImage do
      FinalImage($p$) = $(1-t)\text{Warp}_0(p) + t\text{Warp}_1(p)$
    end
  end
end

**Animated sequences**

- Specify keyframes and interpolate the lines for the inbetween frames
- Require a lot of tweaking

**Results**

*Michael Jackson’s MTV “Black or White”*
**Multi-source morphing**

References