Image warping/morphing

Digital Visual Effects
Yung-Yu Chuang

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></th>
<th>62</th>
<th>79</th>
<th>23</th>
<th>199</th>
<th>120</th>
<th>158</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>9</td>
<td>62</td>
<td>12</td>
<td>79</td>
<td>34</td>
<td>0</td>
<td>48</td>
</tr>
<tr>
<td>176</td>
<td>135</td>
<td>5</td>
<td>199</td>
<td>191</td>
<td>69</td>
<td>0</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>69</td>
<td>144</td>
<td>147</td>
<td>167</td>
<td>102</td>
<td>62</td>
<td>228</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>262</td>
<td>0</td>
<td>166</td>
<td>123</td>
<td>62</td>
<td>0</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>63</td>
<td>127</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>99</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Image warping

image filtering: change range of image
\[
g(x) = h(f(x))
\]
\[\begin{array}{c}
  f \\
  \downarrow \quad h \\
  g
\end{array}\]
\( h(y) = 0.5y + 0.5 \)

image warping: change domain of image
\[
g(x) = f(h(x))
\]
\[\begin{array}{c}
  f \\
  \downarrow \quad h \\
  g
\end{array}\]
\( h(y) = 2y \)

Image warping

image filtering: change range of image
\[
g(x) = h(f(x))
\]
\[\begin{array}{c}
  f \\
  \downarrow \quad h \\
  g
\end{array}\]
\( h(y) = 0.5y + 0.5 \)

image warping: change domain of image
\[
g(x) = f(h(x))
\]
\[\begin{array}{c}
  f \\
  \downarrow \quad h \\
  g
\end{array}\]
\( h(x, y) = [x, y/2] \)
**Parametric (global) warping**

Examples of parametric warps:

- Translation
- Rotation
- Aspect
- Affine
- Perspective
- Cylindrical

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: $p' = M \cdot p$

**Scaling**

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

$f \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$

Non-uniform scaling: different scalars per component:

$g \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2x \\ 0.5y \end{bmatrix}$

$x \times 2,

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

- 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 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?

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

2D Scale around (0,0)?

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

2D Rotate around (0,0)?

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

2D Shear?

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

2x2 Matrices

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

2D Mirror about Y axis?
\[ x' = -x \]
\[ y' = y \]
\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= \begin{bmatrix}
  -1 & 0 \\
  0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

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

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

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**

```
fwp(I, I', T)
{
    for (y=0; y<I.height; y++)
        for (x=0; x<I.width; x++) {
            (x',y') = T(x,y);
            I'(x',y') = I(x,y);
        }
}
```

Some destination may not be covered
Many source pixels could map to the same destination

- 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)

```python
fwp(I, I', T)
{
    for (y=0; y<I.height; y++)
        for (x=0; x<I.width; x++) {
            (x',y') = T(x,y);
            Splatting(I',x',y',I(x,y),kernel);
        }
}
```
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

```c
Inverse warping
iwp(I, I', T)
{
    for (y'=0; y'<I'.height; y'++)
    for (x'=0; x'<I'.width; x'++) {
        (x,y)=T^{-1}(x',y');
        I'(x',y')=Reconstruct(I,x,y,kernel);
    }
}
```
Inverse warping

- No hole, but must resample
- What value should you take for non-integer coordinate? Closest one?

Reconstruction

- Reconstruction generates an approximation to the original function. Error is called aliasing.

Reconstruction

- Computed weighted sum of pixel neighborhood; output is weighted average of input, where weights are normalized values of filter kernel $k$

\[
p = \frac{\sum k(q_i)q_i}{\sum k(q_i)}
\]

\[
\text{color}=0;
\text{weights}=0;
\text{for all } q\text{'s dist < width}
\text{d} = \text{dist}(p, q);
\text{w} = \text{kernel}(d);
\text{color} += \text{w} \times q\text{.color};
\text{weights} += \text{w};
p\text{.Color} = \text{color} / \text{weights};
\]
The reconstructed function is obtained by interpolating among the samples in some manner.
Reconstruction (interpolation)

- Possible reconstruction filters (kernels):
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc (optimal reconstruction)

Bilinear interpolation (triangle filter)

- A simple method for resampling images

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

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
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

Barycentric coordinates

\[ t_1 A_1 + t_2 A_2 + t_3 A_3 \]

\[ t_1 + t_2 + t_3 = 1 \]

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 \[ \rho(r) = e^{-\beta r^2} \]

thin plate spline \[ \rho(r) = r^2 \log(r) \]

radial basis function

\[ \Delta P = \frac{1}{K} \sum_i^K k_{X_i}(P') \Delta X_i \]
Image warping

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

An application of image warping: face beautification

Data-driven facial beautification

Facial beautification
Facial beautification

Training set
- 92 young Caucasian female
- 33 young Caucasian male

Feature extraction
Feature extraction

- Extract 84 feature points by BTSM
- Delaunay triangulation $\rightarrow$ 234D distance vector (normalized by the square root of face area)

BTSM scatter plot for all training faces

234D vector

Support vector regression (SVR)

- Similar concept to SVM, but for regression
- RBF kernels
  
  $f_b(v)$

$$\text{SVM} \quad f(x) = \text{sign}(w^T x + b)$$

$$\text{SVR} \quad f(x) = w^T x + b$$

Minimize $\frac{1}{2} \|w\|^2$

Subject to $y_i (w^T x_i + b) \geq 1$

Minimize $\frac{1}{2} \|w\|^2$

Subject to $\|y_i - (w^T x_i + b)\| \leq \varepsilon$

Beautification engine

Beautification process

- Given the normalized distance vector $v$, generate a nearby vector $v'$ so that $f_b(v') > f_b(v)$

- Two options
  - KNN-based
  - SVR-based
KNN-based beautification

\[ w_i = \frac{b_i}{\|v - v_i\|} \]

\[ v' = \frac{\sum_{i=1}^{K} w_i v_i}{\sum_{i=1}^{K} w_i} \]

SVR-based beautification

- Directly use \( f_b \) to seek \( v' \)
  \[ v' = \arg\min_u E(u), \quad \text{where} \quad E(u) = -f_b(u) \]

- Use standard no-derivative direction set method for minimization

- Features were reduced to 35D by PCA

SVR-based beautification

- Problems: it sometimes yields distance vectors corresponding to invalid human face

- Solution: add log-likelihood term (LP)

\[ E(u) = (\alpha - 1)f_b(u) - \alpha LP(u) \]

- LP is approximated by modeling face space as a multivariate Gaussian distribution

\[ P(\hat{u}) = \frac{1}{(2\pi)^{N/2} \sqrt{\prod_i \lambda_i}} \prod_i \exp\left( -\frac{\beta_i^2}{2\lambda_i} \right) \]

- \( u \)’s projection in PCA space

\[ LP(\hat{u}) = \sum \frac{-\beta_i^2}{2\lambda_i} + \text{const} \]

PCA
**Embedding and warping**

- **Distance embedding**
  - Convert modified distance vector $v'$ to a new face landmark
  \[ E(q_1, \ldots, q_N) = \sum_{e_{ij}} \alpha_{ij} \left( \|q_i - q_j\|^2 - d_{ij}^2 \right)^2 \]
  - 1 if $i$ and $j$ belong to different facial features
  - 10 otherwise

- A graph drawing problem referred to as a stress minimization problem, solved by LM algorithm for non-linear minimization

**Distance embedding**

- Post processing to enforce similarity transform for features on eyes by minimizing
  \[ \sum \|Sp_i - q_i\|^2 \]
  - 
  \[ S = \begin{pmatrix} a & b & t_x \\ -b & a & t_y \\ 0 & 0 & 1 \end{pmatrix} \]
Results (in training set)

User study

<table>
<thead>
<tr>
<th></th>
<th>Original portrait</th>
<th>Warped to mean</th>
<th>KNN-beautified (best)</th>
<th>SVR-beautified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.37 (0.49)</td>
<td>3.75 (0.49)</td>
<td>4.14 (0.51)</td>
<td>4.51 (0.49)</td>
</tr>
</tbody>
</table>

Results (not in training set)

By parts

- Eyes: (a), (b), (c)
- Mouth: (d), (e) full, (f)
Different degrees

50% 100%

Facial collage

Result

• video

Image morphing
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.

image #1  dissolving  image #2

Artifacts of cross-dissolving

http://www.salavon.com/

Image morphing

• Why ghosting?
• Morphing = warping + cross-dissolving

shape (geometric)
color (photometric)
Morphing sequence

Face averaging by morphing

Image 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

An ideal example (in 2004)
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

An ideal example

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’:

\[ u = \frac{(X-P) \cdot (Q-P)}{||Q-P||^2} \]  
\[ v = \frac{(X-P) \cdot \text{Perpendicular}(Q-P)}{||Q-P||} \]  
\[ X' = P' + u \cdot (Q'-P') + \frac{v \cdot \text{Perpendicular}(Q'-P')}{||Q'-P'||} \]

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. destinationImage(X) = sourceImage(X’)

- Examples:

Affine transformation
**Multiple Lines**

\[ D_i = X_i' - X_i \]

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

\( \text{length} = \text{length of the line segment}, \)
\( \text{dist} = \text{distance to line segment} \)

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

---

**Full Algorithm**

```
WarpImage(SourceImage, L[...], L[...])
begin
    foreach destination pixel X do
        XSum = (0, 0)
        WeightSum = 0
        foreach line L[i] in destination do
            X' = X transformed by (L[i], L'[i])
            weight[i] = weight assigned to X'[i]
            XSum = Xsum + X'[i] * weight[i]
            WeightSum += weight[i]
        end
        X' = XSum/WeightSum
    end
    DestinationImage(X) = SourceImage(X')
end
return Destination
```

---

**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

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
    Warp$_0 = $ WarpImage( Image$_0$, L$_0$[...], L[...])
    Warp$_1 = $ WarpImage( Image$_1$, L$_1$[...], L[...])
    foreach pixel $p$ in FinalImage do
      FinalImage(p) = $(1-t)Warp_0(p) + tWarp_1(p)$
    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

Miss Korea

Picasa recognizes them as the same person.
Align and mean face

Morphing sequence

- video

References

- Data-Driven Enhancement of Facial Attractiveness, SIGGRAPH 2008