Image warping/morphing

Digital Visual Effects, Spring 2005
Yung-Yu Chuang
2005/3/9

with slides by Richard Szeliski, Steve Seitz and Alexei Efros

Announcements

• Class time: 1:30-4:20 (with a 20-minute break)
• Last call: send cyy@csie.ntu.edu.tw to subscribe vfx
• Course forum is set up (see course page)
• Scribe volunteers for today and next week
• A schedule for scribes will be posted in forum soon. Please fill in the schedule.

Outline

• Images
• Image warping
• Image morphing
• Project #1

Image fundamentals
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:
  \[
  f[i, j] = \text{Quantize}\{ f(i \cdot D, j \cdot D) \}
  \]
- The image can now be represented as a matrix of integer values

<table>
<thead>
<tr>
<th>( i )</th>
<th>62</th>
<th>79</th>
<th>23</th>
<th>119</th>
<th>120</th>
<th>105</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>78</td>
<td>34</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>35</td>
<td>5</td>
<td>188</td>
<td>191</td>
<td>68</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>0</td>
<td>89</td>
<td>144</td>
<td>147</td>
<td>187</td>
<td>182</td>
<td>62</td>
<td>208</td>
<td>0</td>
</tr>
<tr>
<td>255</td>
<td>252</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>30</td>
<td></td>
</tr>
</tbody>
</table>
Aliasing

Image processing

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

Point processing

Image enhancement

\[ h(a) = 1 - a \]
\[ h(a) = a^r \]
Contrast stretching

Histogram

Accumulated histogram

Histogram equalization
Histogram matching

It is useful for calibrating exposure.

Neighborhood Processing (filtering)

- Q: What happens if I reshuffle all pixels within the image?
- A: It’s histogram won’t change. No point processing will be affected...
- Need spatial information to capture this.

Noise

- Gaussian noise
- Salt and pepper noise
- Impulse noise
- Gaussian noise

Noise reduction

- Mean filter
- Median filter
- Gaussian filter
Comparison: salt and pepper noise

<table>
<thead>
<tr>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="3x3" /></td>
<td><img src="image2" alt="3x3" /></td>
<td><img src="image3" alt="3x3" /></td>
</tr>
<tr>
<td><img src="image4" alt="5x5" /></td>
<td><img src="image5" alt="5x5" /></td>
<td><img src="image6" alt="5x5" /></td>
</tr>
<tr>
<td><img src="image7" alt="7x7" /></td>
<td><img src="image8" alt="7x7" /></td>
<td><img src="image9" alt="7x7" /></td>
</tr>
</tbody>
</table>

Comparison: Gaussian noise

<table>
<thead>
<tr>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image10" alt="3x3" /></td>
<td><img src="image11" alt="3x3" /></td>
<td><img src="image12" alt="3x3" /></td>
</tr>
<tr>
<td><img src="image13" alt="5x5" /></td>
<td><img src="image14" alt="5x5" /></td>
<td><img src="image15" alt="5x5" /></td>
</tr>
<tr>
<td><img src="image16" alt="7x7" /></td>
<td><img src="image17" alt="7x7" /></td>
<td><img src="image18" alt="7x7" /></td>
</tr>
</tbody>
</table>

Image warping

Image filtering: change **range** of image

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

Image warping: change **domain** of image

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

image warping: change \textit{domain} of image
\[ f(x) = g(h(x)) \]

\textbf{Examples of parametric warps:}
- translation
- rotation
- similarity
- affine
- perspective
- cylindrical

\textbf{2D coordinate transformations}
- translation: \[ x' = x + t \quad x = (x,y) \]
- rotation: \[ x' = R x + t \]
- similarity: \[ x' = s R x + t \]
- affine: \[ x' = A x + t \]
- perspective: \[ x' \approx H x \quad x = (x,y,1) \]
  \textit{(x is a homogeneous coordinate)}
- These all form a nested \textit{group} (closed under composition w/ inv.)

\textbf{2D image transformations}
Image warping

- Given a coordinate transform \( x' = h(x) \) and a source image \( f(x) \), how do we compute a transformed image \( g(x') = f(h(x)) \)?

Forward warping

- Send each pixel \( f(x) \) to its corresponding location \( x' = h(x) \) in \( g(x') \)

  - What if pixel lands “between” two pixels?

  - Answer: add “contribution” to several pixels, normalize later (splatting)

Inverse warping

- Get each pixel \( g(x') \) from its corresponding location \( x = h^{-1}(x') \) in \( f(x) \)

  - What if pixel comes from “between” two pixels?
Inverse warping

- Get each pixel $g(x')$ from its corresponding location $x = h^{-1}(x')$ in $f(x)$

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

Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc / FIR

Bilinear interpolation

- A simple method for resampling images

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

Bicubic interpolation

Non-parametric image warping

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

Demo

- http://www.colonize.com/warp/
- 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.
Image morphing

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

| shape (geometric) | color (photometric) |

Morphing sequence

Artifacts of cross-dissolving

http://www.salavon.com/
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

An ideal example
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 (field warping)

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

Beier & Neely (SIGGRAPH 1992)

- Single line-pair PQ to P’Q’:

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

\[ D_i = X'_i - X_i \]

- **Destination Image**
- **Source Image**

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

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

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

Full Algorithm

For each pixel \(X\) in the destination

- \(D_{SUM} = (0,0)\)
- \(\text{weightsum} = 0\)

For each line \(P_i Q_i\)

- calculate \(u,v\) based on \(P_i Q_i\)
- calculate \(X'_i\) based on \(u,v\) and \(P_i Q'_i\)
- calculate displacement \(D_i = X'_i - X_i\) for this line

\[
\text{dist} = \text{shortest distance from } X \text{ to } P_i Q_i
\]

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

\[ DSUM += D_i \times \text{weight} \]

\[ \text{weightsum} += \text{weight} \]

\[ X' = X + DSUM / \text{weightsum} \]

destinationImage(X) = sourceImage(X')

Resulting warp

Animated sequences

- Specify keyframes and interpolate the lines for the inbetween frames
- Require a lot of tweaking
Comparison to mesh morphing

- Pros: more expressive
- Cons: speed and control

Warp interpolation

- How do we create an intermediate warp at time $t$?
- For optical flow:
  - Easy. Interpolate each flow vector
- For feature point methods:
  - linear interpolation of each feature pair
- For Beier-Neely:
  - Can do the same 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
  - Not very intuitive

Other Issues

- Beware of folding
  - Can happen in any of the methods
  - You are probably trying to do something 3D-ish
- Extrapolation can sometimes produce interesting effects
  - Caricatures

Results

*Michael Jackson’s MTV “Black or White”*
Warp specification

- How can we specify the warp
  3. Specify corresponding points
     - interpolate to a complete warping function

Solution#1: 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

Solution#2: scattered point interpolation

- RBF
- Work minimization

Transition control
Transition control

Multi-source morphing

\[ W_i(p) = \sum_{j=1}^{n} b_j W_{ij}(p) \]

\[ I_i(r) = W_i(p) \cdot b_i I_i(p) \]

\[ I(r) = \sum_{i=1}^{n} I_i(r) \]
Project #1: image morphing

- Assigned: 3/9
- Due: 11:59pm 3/29
- Work in pairs
- Handout will be online by tomorrow noon. I will send a mail to vfx when it is available.
- We will provide a generic image library, gil.
Reference software

- Morphing software review
- I used FantaMorph 30-day evaluation version. You can use any one you like.

Morphing is not only for faces

Bells and whistles

- Multi-source morphing
- Automatic morphing
- Morphing for animated sequences
Submission

- You have to turn in your complete source, the executable, a html report and an artifact.
- Report page contains:
  description of the project, what do you learn, algorithm, implementation details, results, bells and whistles...
- Artifacts must be made using your own program. artifacts voting on forum
- Submission mechanism will be announced later.