Advanced Multimedia

Warping & Morphing
Tamara Berg
Image Warping

http://www.jeffrey-martin.com

Slides from: Alexei Efros & Steve Seitz
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. *Argyroplecus Oljersi.*

Fig. 518. *Sternopyx diaphana.*

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

Slide by Durand and Freeman
Image Transformations

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

image warping: change *domain* of image
\[ g(x) = f(T(x)) \]
Image Transformations

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:

$$p' = Mp$$

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

Scaling a coordinate means multiplying each of its components by a scalar.

Uniform scaling means this scalar is the same for all components:
Scaling

*Non-uniform scaling*: different scalars per component:

- $X \times 2,$
- $Y \times 0.5$
Scaling

Scaling operation:

\[ x' = ax \]
\[ y' = by \]
Scaling operation:

\[ x' = ax \]
\[ y' = by \]

Or, in matrix form:

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

scaling matrix \( S \)

What’s inverse of \( S \)?
2-D Rotation

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

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

\[ x = r \cos(\phi) \]
\[ y = r \sin(\phi) \]
\[ x' = r \cos(\phi + \theta) \]
\[ y' = r \sin(\phi + \theta) \]

Trig Identity...
\[ x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta) \]
\[ y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta) \]
2-D Rotation

\[ x = r \cos(\phi) \]
\[ y = r \sin(\phi) \]

\[ x' = r \cos(\phi + \theta) \]
\[ y' = r \sin(\phi + \theta) \]

Trig Identity...
\[ x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta) \]
\[ y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta) \]

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

This is easy to capture in matrix form:

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

What is the inverse transformation?
2-D Rotation

This is easy to capture in matrix form:

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

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?

\[ x' = x \]
\[ y' = y \]
2x2 Matrices

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

2D Identity?

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

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

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

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

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  s_x & 0 \\
  0 & s_y
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
2x2 Matrices

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

2D Rotate around (0,0)?

\[ x' = \cos \Theta \cdot x - \sin \Theta \cdot y \]
\[ y' = \sin \Theta \cdot x + \cos \Theta \cdot y \]

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

2D Shear?

\[ x' = x + sh_x \cdot y \]
\[ y' = sh_y \cdot x + y \]

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} =
\begin{bmatrix}
    1 & sh_x \\
    sh_y & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]
2x2 Matrices

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

2D Mirror about Y axis?

\[ x' = -x \]
\[ y' = y \]
2x2 Matrices

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

2D Mirror about Y axis?

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

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

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

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

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix} \begin{bmatrix}
x \\
y
\end{bmatrix}
\]
2x2 Matrices

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

2D Translation?

\[ x' = x + t_x \]
\[ y' = y + t_y \]
2x2 Matrices

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

2D Translation?

\[ x' = x + t_x \quad \text{NO!} \]
\[ y' = y + t_y \]

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

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

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

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

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

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

Translate

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

Rotate

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

Scale

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

Shear

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix}
    1 & sh_x & 0 \\
    sh_y & 1 & 0 \\
    0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]
Matrix Composition

Transformations can be combined by matrix multiplication

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

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

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]
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>[ I \</td>
<td>t ]_{2\times3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>[ R \</td>
<td>t ]_{2\times3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>similarity</td>
<td>[ sR \</td>
<td>t ]_{2\times3}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>affine</td>
<td>[ A ]_{2\times3}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projective</td>
<td>[ \tilde{H} ]_{3\times3}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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?
Translation: # correspondences?

How many correspondences needed for translation?
Translation: # correspondences?

How many correspondences needed for translation?
How many Degrees of Freedom?
Translation: # correspondences?

How many correspondences needed for translation?
How many Degrees of Freedom?
What is the transformation matrix?

\[ T(x,y) \]

\[ M = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix} \]
Euclidian: # correspondences?

How many correspondences needed for translation+rotation?
How many DOF?
Affine: # correspondences?

How many correspondences needed for affine?
How many DOF?
Affine: # correspondences?

How many correspondences needed for affine?
How many DOF?

\[
\begin{pmatrix}
x' \\
y' \\
w'
\end{pmatrix} =
\begin{pmatrix}
a & b & c \\ 
d & e & f \\ 
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
w
\end{pmatrix}
\]
Projective: # correspondences?

How many correspondences needed for projective?
How many DOF?

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

Given two triangles: ABC and A’B’C’ in 2D (12 numbers)
Need to find transform T to transfer all pixels from one to the other.

How can we compute the transformation matrix:

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]
Example: warping triangles

Given two triangles: ABC and A’B’C’ in 2D (12 numbers)
Need to find transform T to transfer all pixels from one to the other.

How can we compute the transformation matrix:

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} =
\begin{bmatrix}
    a & b & c \\
    d & e & f \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

cp2transform in matlab!
input – correspondences.
output – transformation.
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”
Given a coordinate transform \((x',y') = T(x,y)\) and a source image \(f(x,y)\), how do we compute a transformed image \(g(x',y') = f(T(x,y))\)?

imtransform in matlab!
input – transform and source image
output – transformed image.
HW4 - Motivation

HP commercial
HW4 – Part 1

Target + Source
HW4 – Part 1

Target + Source

User clicks on 4 points in target, 4 points in source.
User clicks on 4 points in target, 4 points in source. Find best affine transformation between source subimg and target subimg.
User clicks on 4 points in target, 4 points in source.
Find best affine transformation between source subimg and target subimg.
Transform the source subimg accordingly.
HW4 – Part 1

User clicks on 4 points in target, 4 points in source.
Find best affine transformation between source subimg and target subimg.
Transform the source subimg accordingly.
Insert the transformed source subimg into the target image.
User clicks on 4 points in target, 4 points in source. 
Find best affine transformation between source subimg and target subimg. 
Transform the source subimg accordingly. 
Insert the transformed source subimg into the target image.

Useful matlab functions: ginput, cp2transform, imtransform. Look these up!
HW4 – Part 1

Target + Source = Result
HW4 Part 2

Do the same thing, but for a bunch of frames in a video!
Image Morphing - Women in Art video

http://youtube.com/watch?v=nUDIoN-_-_Hxs
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
Idea #1: Cross-Dissolve

Interpolate whole images:

\[ Image_{\text{halfway}} = (1-t) \cdot \text{Image}_1 + t \cdot \text{image}_2 \]

This is called **cross-dissolve** in film industry.
Idea #1: Cross-Dissolve

Interpolate whole images:

\[ \text{Image}_{\text{halfway}} = (1-t)\text{Image}_1 + t\text{image}_2 \]

This is called **cross-dissolve** in film industry

But what if 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
Idea #3: Local warp, then cross-dissolve

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 spline control points
- 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 do it?

How do we go from feature points to pixels?
1. Input correspondences at key feature points
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
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!
Image Morphing

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
- $(1-t)p + tp_0$ for corresponding features $p_0$ and $p_1$
Morphing & matting

Extract foreground first to avoid artifacts in the background

Slide by Durand and Freeman