Women in Art video

http://youtube.com/watch?v=nUDIoN-__Hxs
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. *Argyropelecus Olfersi.*

Fig. 518. *Sternopyx diaphana.*

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

Slide by Durand and Freeman
What if we know $f$ and $g$ and want to recover the transform $T$?

- e.g. better align images from Project 1
- willing to let user provide correspondences
  - How many do we need?
Translation: # correspondences?

How many correspondences needed for translation?

How many Degrees of Freedom?

What is the transformation matrix?

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

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

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

How many correspondences needed for projective?
How many DOF?
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.

What kind of transformation is T?

How can we compute the transformation matrix:

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

Two ways: Algebraic and geometric
warping triangles (Barycentric Coordinates)

Don’t forget to move the origin too!

Very useful for Project 5… (hint,hint,nudge,nudge)
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

\[ f(x, y) \rightarrow g(x', y') \]

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”
  – Check out `griddata` in Matlab
Inverse warping

Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image.

Q: what if pixel comes from “between” two pixels?
Inverse warping

Get each pixel \( g(x',y') \) from its corresponding location \( (x,y) = T^{-1}(x',y') \) in the first image.

Q: what if pixel comes from “between” two pixels?

A: Interpolate color value from neighbors
   - nearest neighbor, bilinear, Gaussian, bicubic
   - Check out interp2 in Matlab
Forward vs. inverse warping

Q: which is better?

A: usually inverse—eliminates holes
   • however, it requires an invertible warp function—not always possible...
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
Averaging Points

What’s the average of P and Q?

Linear Interpolation (Affine Combination):
New point $aP + bQ$, defined only when $a+b = 1$
So $aP + bQ = aP + (1-a)Q$

$P + 0.5v$
$= P + 0.5(Q - P)$
$= 0.5P + 0.5Q$

$P + 1.5v$
$= P + 1.5(Q - P)$
$= -0.5P + 1.5Q$
(extrapolation)

P and Q can be anything:
- points on a plane (2D) or in space (3D)
- Colors in RGB or HSV (3D)
- Whole images (m-by-n D)... etc.
Idea #1: Cross-Dissolve

Interpolate whole images:

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

This is called cross-dissolve in film industry

But what is the images are not aligned?
Idea #2: Align, then cross-disolve

Align first, then cross-dissolve

- Alignment using global warp – picture still valid
Global warp not always enough!

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
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)
Warp specification -- dense

Define vector field to specify a dense warp
Warp specification - sparse

How can we specify a sparse warp?

How do we go from feature points to pixels?
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?
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.

What kind of transformation is T?
How can we compute the transformation matrix:

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

Two ways: Algebraic and geometric
A *triangulation* of a set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.

There are an exponential number of triangulations of a point set.
An $O(n^3)$ Triangulation Algorithm

Repeat until impossible:

- Select two sites.
- If the edge connecting them does not intersect previous edges, keep it.
“Quality” Triangulations

Let $\alpha(T) = (\alpha_1, \alpha_2, \ldots, \alpha_{3t})$ be the vector of angles in the triangulation $T$ in increasing order.

A triangulation $T_1$ will be “better” than $T_2$ if $\alpha(T_1) > \alpha(T_2)$ lexicographically.

The Delaunay triangulation is the “best”

- Maximizes smallest angles
Improving a Triangulation

In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.

If an edge flip improves the triangulation, the first edge is called *illegal*. 
Naïve Delaunay Algorithm

Start with an arbitrary triangulation. Flip any illegal edge until no more exist.
Could take a long time to terminate.
Delaunay Triangulation by Duality

General position assumption: There are no four co-circular points.

Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.

**Corollary:** The DT may be constructed in $O(n \log n)$ time.

This is what Matlab’s `delaunay` function uses.
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
1. Create Average Shape

How do we create an intermediate warp at time $t$?

- Assume $t = [0,1]$
- Simple linear interpolation of each feature pair
  - $p=(x,y) \rightarrow p'(x,y)$
- $(1-t)p + t p'$ for corresponding features $p$ and $p'$
2. Create Average Color

Interpolate whole images:

$\text{Image}_{\text{halfway}} = (1-t)\times\text{Image} + t\times\text{image}'$

cross-dissolve!
Morphing & matting

Extract foreground first to avoid artifacts in the background

Slide by Durand and Freeman
Other Issues

Beware of folding
  • You are probably trying to do something 3D-ish

Morphing can be generalized into 3D
  • If you have 3D data, that is!

Extrapolation can sometimes produce interesting effects
  • Caricatures
Dynamic Scene ("Black or White", MJ)

http://www.youtube.com/watch?v=R4kLKv5gtxc
Project #5: morphing

1. Define corresponding points
2. Define triangulation on points
   • Use same triangulation for both images
3. For each \( t = 0: \text{step}: 1 \)
   a. Compute the average \textbf{shape} at \( t \) (weighted average of points)
   b. For each triangle in the average shape
      – Get the affine projection to the corresponding triangles in each image
      – For each pixel in the triangle, find the corresponding points in each image and set value to weighted average (cross-dissolve each triangle)
   c. Save the image as the next frame of the sequence

Matlab hack: can be done with just two nested loops (for \( t \), and for each triangle). Hint: compute warps for all pixels first, then use \texttt{interp2}
Examples