Recovering Transformations

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?

Slides from Alexei Efros
How many correspondences needed for translation?
How many Degrees of Freedom?
What is the transformation matrix?

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

Slides from Alexei Efros
Euclidian: # correspondences?

How many correspondences needed for translation+rotation?
How many DOF?

Slides from Alexei Efros
Affine: # correspondences?

How many correspondences needed for affine?
How many DOF?

Slides from Alexei Efros
Projective: # correspondences?

How many correspondences needed for projective?
How many DOF?

Slides from Alexei Efros
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.

What kind of transformation is T?

How can we compute the transformation matrix:

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

Slides from Alexei Efros
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?

Slides from Alexei Efros
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

Slides from Alexei Efros
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?

Slides from Alexei Efros
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

Slides from Alexei Efros
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 is the images are not aligned?

Slides from Alexei Efros
Idea #2: Align, then cross-dissolve

Align first, then cross-dissolve

- Alignment using global warp – picture still valid

Slides from Alexei Efros
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

Slides from Alexei Efros
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

Slides from Alexei Efros
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

Slides from Alexei Efros
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

Slides from Alexei Efros
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?

Slides from Alexei Efros
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!
   • Just like texture mapping

Slides from Alexei Efros
Triangulations

A *triangulation* of 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.

---

Slides from Alexei Efros
An $O(n^3)$ Triangulation Algorithm

Repeat until impossible:

- Select two sites.
- If the edge connecting them does not intersect previous edges, keep it.
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
- $(1-t)p_1 + tp_0$ for corresponding features $p_0$ and $p_1$