Amuse-bouche

http://youtube.com/watch?v=nUDIoN-_Hxs
Image Warping and Morphing
Project 3 out today!!

project 2 how did it go?

project 3 is harder!
Image Transformations

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

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

image filtering: change range of image

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

image warping: change domain of image

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

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

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

\[ j = (0,1) \]
\[ i = (1,0) \]

\[ q = 4i + 3j = (4,3) \]

\[ p = 4u + 3v \]
Linear Transformations as Change of Basis

Any linear transformation is a basis!!!

\[ p_{uv} = (4,3) \]

\[ p_x = 4u_x + 3v_x \]
\[ p_y = 4u_y + 3v_y \]

\[ p_{ij} = ?i + ?j \]

\[ p_{ij} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} p_{uv} \]

Any linear transformation is a basis!!!
What's the inverse transform?

- How can we change from any basis to any basis?
- What if the basis are orthogonal?

\[ j = (0,1) \]
\[ i = (1,0) \]
\[ p^{ij} = (5,4) = p_x u + p_y v \]

\[ \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}^{-1} \begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}^{-1} p^{ij} \]

\[ p^{uv} = (p_x, p_y) = ? \]
Projection onto orthogonal basis

\[ j = (0, 1) \]

\[ i = (1, 0) \]

\[ p_{ij} = (5, 4) \]

\[ p_{ij} = (v_x, v_y) \]

\[ p_{uv} = (u \cdot p_{ij}, v \cdot p_{ij}) \]

\[ p_{uv} = \begin{bmatrix} u_x & u_x \\ v_y & v_y \end{bmatrix} \begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} p_{ij} \]
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

\[ \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \text{homogeneous coords} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]
Homogeneous Coordinates

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

\[
\begin{align*}
x' &= x + t_x \\
y' &= y + t_y
\end{align*}
\]

A: Using the rightmost column:

\[
\text{Translation} = \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\]
Matrix Composition

Transformations can be combined by matrix multiplication

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

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

Does the order of multiplication matter?
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

Will the last coordinate \( w \) always be 1?
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
Importance of shape and structure in evolution
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?
How many correspondences needed for translation?
How many Degrees of Freedom?
What is the transformation matrix?

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

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

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

\[ T_1^{-1} \]

Inverse change of basis

\[ T_2 \]

change of basis

Don’t forget to move the origin too!

Very useful for Project 3… (hint,hint,nudge,nudge)
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-dissolve

Align first, then cross-dissolve

- Alignment using global warp – picture still valid
Image Morphing

Morphing = warping + cross-dissolving

shape (geometric)  color (photometric)
Two-stage Morphing Procedure

Morphing procedure:

for every $t$,

1. Find the average shape (the “mean dog” 😊)
   - warping

2. Find the average color
   - Cross-dissolve the warped images
BUT: 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.
- But what to do with all the intermediate 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?
Full morphing result

(c) Ian Albuquerque Raymundo da Silva
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:
\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]
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”
   - Check out griddata in Matlab
Inverse warping

\[ f(x,y) \]

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 / Python
Bilinear Interpolation

http://en.wikipedia.org/wiki/Bilinear_interpolation
Help interp2
Forward vs. inverse warping

Q: which is better?

A: usually inverse—eliminates holes
   • however, it requires an invertible warp function—not always possible...
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.
An $O(n^3)$ Triangulation Algorithm

Repeat until impossible:

- Select two sites.
- If the edge connecting them does not intersect previous edges, keep it.
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.
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!
Project #3: morphing

1. Define corresponding points

2. Define triangulation on points
   • Use same triangulation for both images

3. For each t = 0:step:1
   a. Compute the average 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

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

© Rachel Albert, CS194-26, Fall 2015
Examples from last year

@Michael Jayasuriya

@Varun Saran

What’s the difference?
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