Deformation Transfer for Triangle Meshes

Robert W. Sumner  Jovan Popović

Presented by Ryan Braud

Overview

- Mesh deformation plays a central role in computer modeling and animation
- Constructing complex deformations is a time-consuming process
- Very few techniques to help with reuse
  - Special purpose methods do exist
  - Techniques usually fail when multiple deformation techniques used simultaneously

Deformation Transfer

- Given source mesh and deformation, apply the deformation to a target mesh

Similar to...

- "Image Analogies" by A. Hertzmann, et al.

Deformation Transfer

- Compute a set of transformations induced by deformation on source mesh
- Map transformations from source polygons to target polygons
- Solve an optimization problem to consistently apply the transformations to the target mesh

Toy Example

- Given source mesh, deformed source mesh, and target mesh, compute deformed target mesh
Represent source deformation as series of affine transformations $Q+d$, one per triangle. $Q$ is a $3 \times 3$ matrix, $d$ is translation.

Triangle vertices alone do not fully determine affine transformation.

In two dimensions, need two linearly independent vectors to form a new basis.

In three dimensions, need three vectors.

Need a 4th vertex per triangle to uniquely determine $Q$.

Compute 4th undeformed vertex as:

$$v_4 = v_1 + (v_2 - v_1) \times (v_3 - v_1) \times (v_2 - v_1)$$

Note: $v_4$ is not co-planar with $v_1$, $v_2$, and $v_3$.

Given the undeformed ($v_i$) and deformed ($\tilde{v}_i$) vertices for each triangle, what is $Q$?

For $i \in 1...4$ we know:

$$Qv_i + d = \tilde{v}_i$$

Subtract the first equation from the others:

$$Q(v_i - v_1) = \tilde{v}_i - v_1$$

Let $V = [v_1 - v_1, v_1 - v_2, v_2 - v_1, v_2 - v_3]$

Then $Q = F V^{-1}$.

Define a mapping from source triangles to target triangles:

$$M = \{(s_i, t_j), (s_2, t_2), \ldots, (s_{|V|}, t_{|V|})\}$$

A pair $(s_i, t_j)$ indicates target triangle $t_j$ should deform like source triangle $s_i$.

$M$ can be an arbitrary mapping, but must contain every source and target polygon.
Polygon Correspondence

- Many-to-one and one-to-many correspondences are possible

Source Mesh

Target Mesh

First Approach

- Transfer the source transformations via the correspondence map to the target
- Apply non-translational portion $Q$ of each source affine transformation using $M$

Second Approach

- Use source displacement vectors as well as $Q$

Final Approach

- Enforce vertex consistency requirements by solving optimization problem

Adding Constraints

- Shared vertices must be transformed to the same place:
  $$T_j v_i + d_j = T_k v_{i+1} + d_{k+1}, \forall j, k \in p(v_i)$$

Adding Constraints

- Define unknown target transformations in terms of the triangles' vertices
- Let $T$ be a target transformation
  - Recall $T=FJ^{-1}$
  - $J^{-1}$ depends on known undeformed target vertices
  - $F$ - coordinates of unknown deformed vertices
- Note: elements of $T$ are linear combinations of coordinates of unknown deformed vertices
Finding Transformed Vertices

- Want to minimize the difference between non-translational components of the source and target deformations

\[
\min_{v_j,v_t} \sum_{j=1}^{M} \| S_{v_j} - T_{v_j} \|_F^2.
\]

- \( s_j \) - index of source transformation for correspondence entry \( j \)
- \( t_j \) - index of target transformation for correspondence entry \( j \)
- \( S_{v_j} \) - source transformation for correspondence entry \( j \)
- \( T_{v_j} \) - target transformation for correspondence entry \( j \)

Finding Transformed Vertices

- Rewrite minimization equation:

\[
\min_{v_1,...,v_n} \sum_{j=1}^{M} \| S_{v_j} - T_{v_j} \|_F^2.
\]

- Minimization is over deformed vertices themselves
- Continuity constraints implicitly satisfied by solving for deformed vertex positions directly

Matrix Formulation

- Solution to optimization problem is a solution to a system of linear equations:

\[
\min_{\bar{x}} \| \bar{x} - c \|_2^2
\]

- \( \bar{x} \) - vector of unknown deformed vertices
- \( c \) - entries from source transformations
- \( A \) - matrix relating \( \bar{x} \) to \( c \)

Matrix Formulation

- Solve in the least squares sense (pseudoinverse):

\[
A^T A \bar{x} = A^T c \\
\bar{x} = (A^T A)^+ A^T c
\]

Correspondence Revisited

- So, how do we compute \( M \) anyways?
- Could define it by hand, one polygon at a time
  - This is too tedious
- Instead, let user select a few control points and “learn” the rest
- Solve a minimization problem that attempts to deform the source mesh into the target mesh
- Iterate the deformation, refining the process

Correspondence Revisited

- \( A \) - target mesh
- \( B \) - source mesh
- \( C \) - source mesh after first phase of deformation
- \( D \) - final deformed mesh
Results

Questions?