Large Mesh Deformation Using the Volumetric Graph Laplacian

Kun Zhou¹ Jin Huang²* John Snyder³ Xinguo Liu¹ Hujun Bao² Baining Guo¹ Heung-Yeung Shum¹

¹ Microsoft Research Asia ² Zhejiang University ³ Microsoft Research

Presented by Bhaskar Kishore
Outline

- Introduction
- Related Work
- Deformation on Volumetric Graphs
- Deformation from 2D curves
- Results
- Conclusions
Outline

• Introduction
• Related Work
• Deformation on Volumetric Graphs
• Deformation from 2D curves
• Results
• Conclusions
Introduction

- Large deformations are challenging
- Existing techniques often produce implausible results
- Observation
  - Unnatural volume changes
  - Local Self Intersection
Introduction

• Volumetric Graph Laplacian
  – Represent volumetric details as difference between each point in a 3D volume and the average of its neighboring points in a graph.
  – Produces visually pleasing deformation results
  – Preserves surface details

• VGL can impose volume constraints

• Volumetric constraints are represented by a quadric energy function
Introduction

- **Volumetric Graph Laplacian**
  - Represents volumetric details as the difference between each point in a 3D volume and the average of its neighboring points.
  - Produces visually pleasing deformation results.
  - Preserves surface details.
- VGL can impose volume constraints.
- Volumetric constraints are represented by a quadric energy function.
Introduction

- To apply VGL to a triangle mesh
  - Construct a volumetric graph which includes
    - Points on the original mesh
    - Points derived from a simple lattice lying inside the mesh
  - Points are connected by graph edges which are a superset of the edges of the original mesh
- What's nice is that there is no need for volumetric tessellation.
- Deformations are specified by identifying a limited set of points – say a curve
Introduction

• This curve can then be deformed to specify destination

• A quadric energy function is generated
  – Minimum maps the points to their specified destination
  – While preserving surface detail and roughly volume too
Introduction

• Contribution
  – Demonstrate that problem of large deformation can be effectively solved by volumetric differential operator
    • Surface operators can be extended to solids by defining them on tetrahedral mesh
    • But that is difficult, constructing the tetrahedral mesh is hard
    • Existing packages remesh geometry and change connectivity
  – That a volumetric operator can be applied to the easy to build Volumetric graph without meshing int.
Outline

• Introduction
• Related Work
• Deformation on Volumetric Graphs
• Deformation from 2D curves
• Results
• Conclusions
Related Work

• Freeform modeling [Botsch Kobbelt 2004]
• Curve based FFD [Sing and Fiume 1998]
• Lattice based FFD [Sederberg and Parry 1986]
• Displacement volumes [Botsch and Kobbelt 2003]
• Poisson meshes [Yu et al 2004]
Outline

- Introduction
- Related Work
- Deformation on Volumetric Graphs
- Deformation from 2D curves
- Results
- Conclusions
Deformation on Volumetric Graphs

• Let $M = (V, K)$ be a triangular mesh
  – $V = \{p_i \in \mathbb{R}^3 \mid 1 \leq i \leq n\}$, is a set of $n$ point position
  – $K$ is a abstract simplicial complex containing three types of elements
    • Vertices \{i\}
    • Edges \{i,j\}
    • Faces \{i,j,k\}
Laplacian Deformation on Abstract Graphs

- Suppose $G = (P, E)$ is a graph
  - $P \{p_i \in \mathbb{R}^3 \mid 1 \leq i \leq N\}$, is a set of $N$ point positions
  - $E = \{(i, j) \mid p_i \text{ is connected to } p_j\}$

- Then Laplacian coordinate $\delta_i$ of a point $p_i$

\[ \delta_i = \mathcal{L}_G(p_i) = p_i - \sum_{j \in \mathcal{N}(i)} w_{ij} p_j, \quad (1) \]

where $\mathcal{N}(i) = \{j \mid \{i, j\} \in E\}$

- $L_G$ is called the Laplace operator on graph $G$
Laplacian Deformation on Abstract Graphs

- To control the deformation
  - User inputs deformed positions $q_i$, $i \in \{1, ..., m\}$ for a subset of the $N$ mesh vertices
  - Compute a new (deformed) laplacian coordinate $\delta'_i$ for each point $i$ in the graph
  - Deformed positions of the mesh vertices $p'_i$ is obtained by solving

$$
\min_{p'_i} \left( \sum_{i=1}^{N} \| \mathcal{L}_G(p'_i) - \delta'_i \|^2 + \alpha \sum_{i=1}^{m} \| p'_i - q_i \|^2 \right). \tag{2}
$$
Laplacian Deformation on Abstract Graphs

\[
\min_{p_i'} \left( \sum_{i=1}^{N} \| \mathcal{L}_G(p_i') - \delta_i' \|^2 + \alpha \sum_{i=1}^{m} \| p_i' - q_i \|^2 \right). \quad (2)
\]

- The first term represents preservation of local detail
- The second term constrains the position of those vertices directly specified by the user
- Alpha is used to balance these two objectives
Laplacian Deformation on Abstract Graphs

- Deformed Laplacian coordinates are computed via
  \[ \delta'_i = T_i \delta_i \]
  - \( \delta_i \) is the Laplacian in rest pose
  - \( \delta'_i \) is the Laplacian in the deformed pose
  - \( T_i \) is restricted to rotation and isotropic scale
  - Local transforms are propagated from the deformed region to the entire mesh
Constructing a Volumetric Graph

- Build two graphs, $G_{in}$ and $G_{out}$
- $G_{in}$ prevents large volume changes
- $G_{out}$ prevents local self-intersection
- $G_{in}$ can be obtained by tetrahedralizing the interior
  - Difficult to implement
  - Computationally expensive
  - Produces poorly shaped tetrahedra for complex models
Constructing a Volumetric Graph:

- Build two graphs, $G_{\text{in}}$ and $G_{\text{out}}$.
- $G_{\text{in}}$ prevents large volume changes.
- $G_{\text{out}}$ prevents local self-intersection.
- $G_{\text{in}}$ can be obtained by tetrahedralizing the interior.
  - Difficult to implement.
  - Computationally expensive.
  - Produces poorly shaped tetrahedra for complex models.

Figure 2: Large twist deformation.

Figure 3: Large bend deformation.

(a) Laplacian surface
(b) Poisson mesh
(c) VGL
Constructing the Volumetric Graph

• Algorithm
  – Construct inner shell \( M_{in} \) for mesh \( M \) by offsetting each vertex a distance in the direction opposite to its Normal.
  – Embed \( M_{in} \) and \( M \) in a body-centered cubic lattice. Remove lattice nodes outside of \( M_{in} \).
  – Build edge connections among \( M \), \( M_{in} \), and lattice nodes.
  – Simplify the graph using edge collapse and smooth the graph.
Constructing the Volumetric Graph

- Construct inner shell $M_{\text{in}}$ for mesh $M$ by offsetting each vertex a distance in the direction opposite to its Normal
Constructing the Volumetric Graph

- Embed $M_{in}$ and $M$ in a body-centered cubic lattice. Remove lattice nodes outside of $M_{in}$.

Figure 6: *Types of edge connections in the volumetric graph.*
Constructing the Volumetric Graph

- Build edge connections among $M$, $M_{in}$, and lattice nodes
Constructing the Volumetric Graph

- Simplify the graph using edge collapse and smooth the graph
Constructing the Volumetric Graph

- $M_{in}$ ensures that inner points are inserted even in thin features that may be missed by lattice sampling.
- Question: how much of a step should one take to construct $M_{in}$?
- Use iterative method based on simplification envelopes [Cohen et al. 1996]
Constructing the Volumetric Graph

- $M_{\text{in}}$ ensures that inner points are inserted even in thin features that may be missed by lattice sampling.

- Question: how much of a step should one take to construct $M_{\text{in}}$?
  - Use iterative method based on simplification envelopes [Cohen et al. 1996]
Constructing the Volumetric Graph

- Use iterative method based on simplification envelopes [Cohen et al. 1996]
  - At each iteration
    - Move each vertex a fraction of the average edge length
    - Test its adjacent triangles for intersection with each other and the rest of the model
    - If no intersections are found, accept step, else reject it
    - Iterations terminate when all vertices have moved desired distance or can no longer move
Constructing the Volumetric Graph

• Use iterative method based on simplification envelopes [Cohen et al. 1996]
  – At each iteration
    • Move each vertex a fraction of the average edge length
    • Test its adjacent triangles for intersection with each other and the rest of the model
    • If no intersections are found, accept step, else reject it
    • Iterations terminate when all vertices have moved desired distance or can no longer move
Constructing the Volumetric Graph

• The BCC lattice
  – Consists of nodes at every point of a Cartesian grid
  – Additionally there are nodes at cell centers
  – Node locations may be viewed as belong to two interlaced grids
  – This lattice provides desirable rigidity properties as seen in crystalline structures in nature
  – Grid interval set to average edge length
Constructing the Volumetric Graph

- Three types of edge connections for an initial graph
  - Each vertex in M is connected to its corresponding vertex in $M_{\text{in}}$. Shorter diagonal for each prism face is included as well.
  - Each inner node of the BCC lattice is connected with its eight nearest neighbors in the other interlaced grid
  - Connections are made between Min and nodes of the BCC lattice.

- For each edge in the BCC lattice that intersects Min and has at least one node inside $M_{\text{in}}$, we connect the BCC lattice node inside $M_{\text{in}}$ to the point in $M_{\text{in}}$ closest to this intersection
Constructing the Volumetric Graph

- Three types of edge connections for an initial graph:
  - Each vertex in M is connected to its corresponding vertex in M.
  - Shorter diagonal for each prism face is included as well.
  - Each inner node of the BCC lattice is connected with its eight nearest neighbors in the other interlaced grid.
  - Connections are made between Min and nodes of the BCC lattice.

- For each edge in the BCC lattice that intersects Min and has at least one node inside Min, we connect the BCC lattice node inside Min to the point in Min closest to this intersection.
Constructing the Volumetric Graph

- Simplification and Smoothing
  - Visit graph in increasing order of length
  - If length of an edge is less than a threshold, collapse it to edge's mid point
    - Threshold is half of average edge length of M
  - Apply iterative smoothing
    - Each point is moved to the average of its neighbors
    - Three smoothing operations in their implementation
  - No smoothing or simplification are applied to the vertices of original mesh M
Constructing the Volumetric Graph

- $G_{out}$ can be constructed in a similar way to $G_{in}$.
- $M_{out}$ can be obtained by moving a small step in the normal direction.
- Connections for $M_{out}$ can be made similar to $M_{in}$.
- Intersections between $M_{in}$ and $M_{out}$ can occur, especially in meshes containing regions of high curvature.
  - They claim it does not cause any difficulty in our interactive system.
Deforming the Volumetric Graph

- We modify equation (2) to include volumetric constraints

\[
\sum_{i=1}^{n} \| \mathcal{L}_M(p'_i) - \varepsilon'_i \|^2 + \alpha \sum_{i=1}^{m} \| p'_i - q_i \|^2 + \beta \sum_{i=1}^{N} \| \mathcal{L}_{G'}(p'_i) - \delta'_i \|^2
\] (3)

Where the first n points in graph G belong the mesh M

- G' is a sub-graph of G formed by removing those edges belonging to M
- \( \delta'_{i} \) (1 ≤ i ≤ N) in G' are the graph laplacians coordinates in the deformed frame.
- For points in the original mesh M, \( \varepsilon' \) (1 ≤ i ≤ n) are the mesh laplacian coordinates in the deformed coordinate frame
Deforming the Volumetric Graph

\[ \sum_{i=1}^{n} \| \mathcal{L}_M(p_i') - \epsilon_i' \|^2 + \alpha \sum_{i=1}^{m} \| p_i' - q_i \|^2 + \beta \sum_{i=1}^{N} \| \mathcal{L}_G(p_i') - \delta_i' \|^2 \] (3)

- \( \beta \) balances between surface and volumetric detail where \( \beta = n\beta'/N \).
- The \( n/N \) factor normalizes the weight so that it is insensitive to the lattice density
- \( \beta' = 1 \) works well
- \( \alpha \) is not normalized – We want constraint strength to depend on the number of constrained points relative to the total number of mesh points
- \( 0.1 < \alpha < 1 \), default is 0.2
Propagation of Local Transforms

- Local Transforms take the Laplacian coordinates in the rest frame to the deformed frame
- Use WIRE deformation method [Singh and Flume]
- Select a sequence of mesh vertices forming a curve
- Deform the curve.
Propagation of Local Transforms

- First determine where neighboring graph points deform to, then infers local transforms at the curve points, finally propagate the transforms over the whole mesh.
- Begin by finding mesh neighbors of $q_i$ and obtain their deformed positions using WIRE.
- Let $C(u)$ and $C'(u)$ be the original curve and the deformed curves parametrized by arc length $u = [0,1]$.
Propagation of Local Transforms

- Given some neighboring point \( p \in \mathbb{R}^3 \), let \( u \) be the parameter value minimizing distance between \( p \) and the curve \( c(u) \).

- The deformation mapping \( p \) to \( p' \) such that \( C \) maps to \( C' \) is given by

\[
p' = C'(u_p) + R(u_p) \left( s(u_p)(p - C(u_p)) \right).
\]

- \( R \) is a 3x3 rotation matrix taking the tangent vector \( t(u) \) on \( C \) and maps it to \( t'(u) \) on \( C' \) by rotating around \( t(u) \times t'(u) \).
Propagation of Local Transforms

\[ p' = C'(u_p) + R(u_p) \left( s(u_p)(p - C(u_p)) \right) . \]

- \( s(u) \) is a scale factor
  - Computed at each curve vertex as the ratio of the sum of lengths of its adjacent edges in \( C' \) over this length in \( C \)
  - It is then defined continuously over \( u \) by linear interpolation

- Above equation gives us deformed coordinates for each point in the curve and its 1 Ring neighborhood
Propagation of Local Transforms

\[ p' = C'(u_p) + R(u_p) (s(u_p)(p - C(u_p))) \].

- Transformations are propagated from the control curve to all graph points \( p \) via a deformation strength field \( f(p) \)
- \( f(p) \) decays away from the deformation site.
  - Constant
  - Linear
  - Gaussian
  - Based on shortest edge path from \( p \) to the curve
Propagation of Local Transforms

$$p' = C'(u_p) + R(u_p) \left( s(u_p)(p - C(u_p)) \right).$$

- A rotation is defined by
  - Computing a normal and tangent vector as the perpendicular projection of one edge vector with this normal
  - Normal is computed as a linear combination weighted by face area of face normals around mesh point i
  - Rotation is represented as a quaternion
    - Angle should be less than 180
Propagation of Local Transforms

\[ p' = C'(u_p) + R(u_p) \left( s(u_p)(p - C(u_p)) \right). \]

- The simplest propagation scheme is to assign to \( p \) a rotation and scale from the point \( q_p \) on the control curve closest to \( p \).

- Smoother results are obtained by computing a weighted average over all the vertices on the control curve instead of the closest.

\[ \exp \left( -\frac{(\| p - q_i \|_g - \| p - q_p \|_g)^2}{2\sigma^2} \right) \]
Propagation of Local Transforms

\[ p' = C'(u_p) + R(u_p) \left( s(u_p)(p - C(u_p)) \right). \]

- Weighting over multiple curves is similar, we accumulate values over multiple curves.
- Final transformation matrix is given by

\[ T_p = f(p) \tilde{T}_p + (1 - f(p)) I \]
Propagation of Local Transforms
Weighting Scheme

- They drop uniform weighting in favor of another scheme that provides better results.
- For mesh Laplacian $L_m$, use cotangent weights:
  \[ w_{ij} \propto (\cot \alpha_{ij} + \cot \beta_{ij}), \]
  where $\alpha_{ij} = \angle(p_i, p_{j-1}, p_j)$ and $\beta_{ij} = \angle(p_i, p_{j+1}, p_j)$.
- For graph Laplacian, compute weights by solving a quadratic programming problem.
Weighting Scheme

- For each graph vertex $i$, to obtain weights $w_{ij}$ solve

$$\min_{w_j} \left( \|p_i - \sum_{j \in \mathcal{N}(i)} w_j p_j\|^2 + \lambda \left( \sum_{j \in \mathcal{N}(i)} w_j \|p_i - p_j\| \right)^2 \right)$$

subject to $\sum_{j \in \mathcal{N}(i)} w_j = 1$ and $w_j > \xi$.

- The first term generates Laplacian coordinates of smallest magnitude
- Second term is based on scale dependent umbrella operator which prefers weight in proportion to inverse of edge length
- Lambda balances the two objects (set to 0.01)
- Zeta prevents small weights (set to 0.01)
Weighting Scheme

(a) Uniform  (b) Edge len. reciprocal  (c) Heat kernel  (d) Our scheme
Quadric Energy Minimization

• To minimize energy in equation (3) we solve the following equations

\[ \mathcal{L}_M(p'_i) + \beta \mathcal{L}_G(p'_i) = \varepsilon'_i + \beta \delta'_i, \quad i \in 1, ..., n, \]

\[ \beta \mathcal{L}_G(p'_i) = \beta \delta'_i, \quad i \in n + 1, ..., N, \]

\[ \alpha p'_i = \alpha q'_i, \quad i \in 1, ..., m \]

– Above equations represent a sparse linear system \( Ax = b \)
– Matrix \( A \) is only dependent on the original graph and \( A^- \) can be precomputed using LU decomposition
– \( B \) depends on current Laplacian coordinates and changes during interactive deformation
Multi resolution Methods

• Solving a the linear system of a large complex model is expensive

• Generate a simplified mesh [Guskov et al. 1999]

• Deform this mesh and then add back the details to obtain high resolution deformed mesh
Outline

● Introduction
● Related Work
● Deformation on Volumetric Graphs
● Deformation from 2D curves
● Results
● Conclusions
Deformation from 2D curves

• Method
  – User defines control curve by selecting sequence points on the mesh which are connected by the shortest edge path (dijkstra)
  – This 3D curve is projected onto one or more planes
  – Editing is done in these planes
  – The deformed curve is projected back into 3D, which then forms the basis of the deformation process
Deformation from 2D curves

• Curve Projection
  – Given a curve, the system automatically selects projection planes base on its average normal and principal vectors.
  – Principal vectors are computed as the two eigen vectors corresponding to the largest eigen values from a principal component analysis
  – In most cases, cross product of the average normal and the first principal vector provide a satisfactory plane
  – If length of average normal vector is small, then use only two principal vectors instead
Deformation from 2D curves

• Curve Editing
  – Projected 2D curves inherit geometric detail from original mesh that complicates editing
  – They use an editing method for discrete curves base on Laplacian coordinates
  – Laplacian coordinate of a curve vertex is the difference between its position and the average position of its neighbors or a single neighbor in cases of terminal vertices
  – Denote the 2D curve to be edited as $C$
  – A cubic B-Spline curve $C_b$ is first computed as a least squares fit to $C$. This represents the low frequencies of $C$
Deformation from 2D curves

• Curve Editing
  – A discrete version of $C_b$, $C_d$ is computed by mapping each vertex of $C$ onto $C_b$ using proportional arc length mapping
  – We can not edit the discrete version conveniently
  – After editing we obtain $C'_b$ and $C'_d$. These curves lack the original detail of the $C_b$
  – To restore detail, at each vertex of $C$ we find a the unique rotation and scale that maps its location from $C_d$ to $C'_d$
  – Applying these transformations to the Laplacian coordinates and solving equation (2) without the constraint term

\[
\min_{p'_i} \left( \sum_{i=1}^{N} \| \mathcal{L}_G(p'_i) - \delta'_i \|^2 + \alpha \sum_{i=1}^{m} \| p'_i - q_i \|^2 \right). \tag{2}
\]
Deformation from 2D curves

• Deformation Re-targeting from 2D Cartoons
  – An application of 2D sketch based deformation
  – Users specify one or more 3D control curves on the mesh along with their project planes and for each curves a series of 2D curves in the cartoon image sequence that drive its deformation
  – It is not necessary to generate a deformation from scratch at every time frame. Users can select a curves in a few key frames of the cartoon
  – Automatic interpolation technique based on differential coordinates is used to interpolate between key frame
Deformation from 2D curves

• Deformation Re-targeting from 2D Cartoons
  - Say we have two meshes M and M' at two different key frames
  - Compute the Laplacian coordinates for each vertex in the two meshes
  - A rotation and scale in the local neighborhood of each vertex p is computed taking the Laplacian coordinates from its location in M to M'
  - Denote the transform as $T_p$. Interpolate $T_p$ over time to transition from M to M'
  - 2D cartoon curves are deformed in a single plane, this allows for extra degrees of freedom if required by the user
Deformation from 2D curves

- Deformation Re-targeting from 2D Cartoons
Deformation from 2D curves

- Deformation Re-targeting from 2D Cartoons
Outline

- Introduction
- Related Work
- Deformation on Volumetric Graphs
- Deformation from 2D curves
- Results
- Conclusions
## Results

- **Some stats**

<table>
<thead>
<tr>
<th></th>
<th>arma</th>
<th>dino</th>
<th>cat</th>
<th>lioness</th>
<th>dog</th>
</tr>
</thead>
<tbody>
<tr>
<td># mesh vertices</td>
<td>15,002</td>
<td>10,002</td>
<td>7,207</td>
<td>5,000</td>
<td>10,002</td>
</tr>
<tr>
<td># graph points</td>
<td>28,142</td>
<td>15,895</td>
<td>14,170</td>
<td>8,409</td>
<td>17,190</td>
</tr>
<tr>
<td>graph generation</td>
<td>2.679s</td>
<td>1.456s</td>
<td>1.175s</td>
<td>1.367s</td>
<td>1.348s</td>
</tr>
<tr>
<td>LU decomposition</td>
<td>0.524s</td>
<td>0.286s</td>
<td>0.348s</td>
<td>0.197s</td>
<td>0.118s</td>
</tr>
<tr>
<td>back substitution</td>
<td>0.064s</td>
<td>0.028s</td>
<td>0.030s</td>
<td>0.019s</td>
<td>0.011s</td>
</tr>
<tr>
<td># control curves</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td># key frames</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>session time (min)</td>
<td>$\sim$120</td>
<td>$\sim$90</td>
<td>$\sim$30</td>
<td>$\sim$90</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: *Statistics and timing.*
More results

(a) Original mesh  (b) Poisson mesh editing  (c) VGL
More results

Figure 12: Deformation transfer from Goofy to armadillo. ©Disney

Figure 13: Deformation transfer from a kicking cat to dinosaur.
Outline

• Introduction
• Related Work
• Deformation on Volumetric Graphs
• Deformation from 2D curves
• Results
• Conclusions
Conclusions

• They proposed a system which would address volumetric changes and local self intersection based on the volumetric graph Laplacian

• The solution avoids the intricacies of solidly meshing complex objects

• Presented a system for retargetting 2D animations to 3D

• Note, that their system does not address global self intersections – those must addressed by the user