

Laplacian-Based Graphical Editing, Synthesis and Simulation

Yizhou Yu

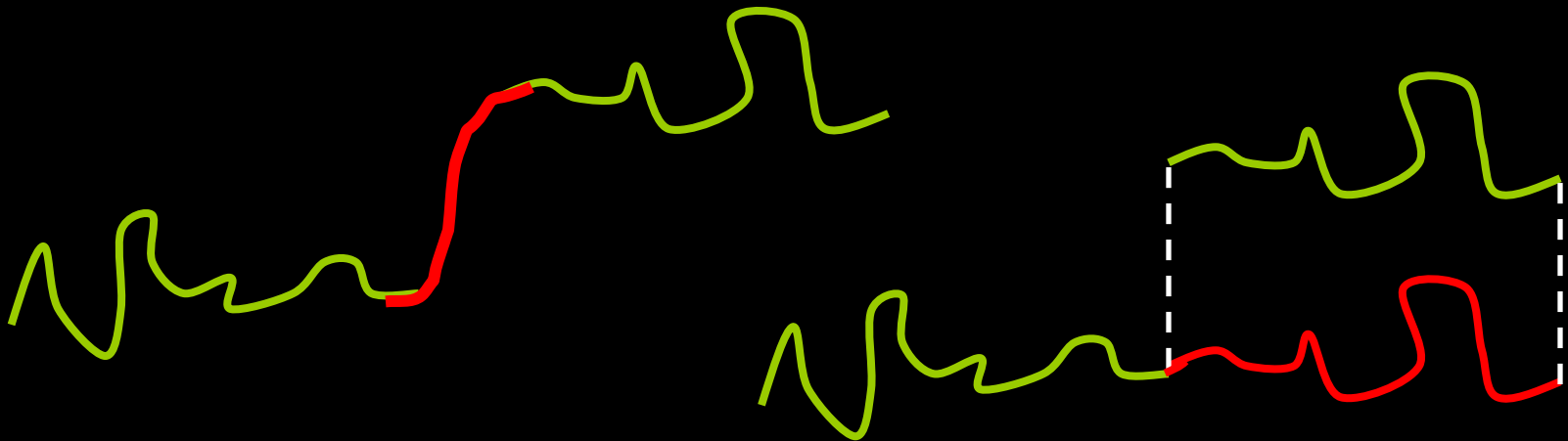
University of Illinois at Urbana-Champaign

Introduction

- The Laplacian has played a critical role in the history of science and engineering, including mathematics, mechanics, electromagnetics, wave theory etc.
 - Laplace's equation, Poisson's equation, etc.
- There has been a recent surge of applying the Laplacian to computer graphics.
 - Discrete Laplacian Operators, [Pinkall & Polthier 1993, Desbrun et al. 1999]
 - Image editing, [Fattal et al. 2002], [Perez et al. 2003]
 - Geometry compression, editing and signal processing [Kobbelt et al. 1998, Karni & Gotsman 2000, Sorkine et al. 2004, Yu et al. 2004, Shi & Yu 2006]
 - Flow simulation, [Stam 1999, Shi & Yu 2004]

Motivation for Laplacian-Based Editing

- Detail Preservation
 - It is more important to preserve high-frequency details than smoothly varying low-frequency changes.
 - *Neither create nor remove details*
 - Details can be measured by differential properties, such as gradients, Laplacian or even higher-order derivatives



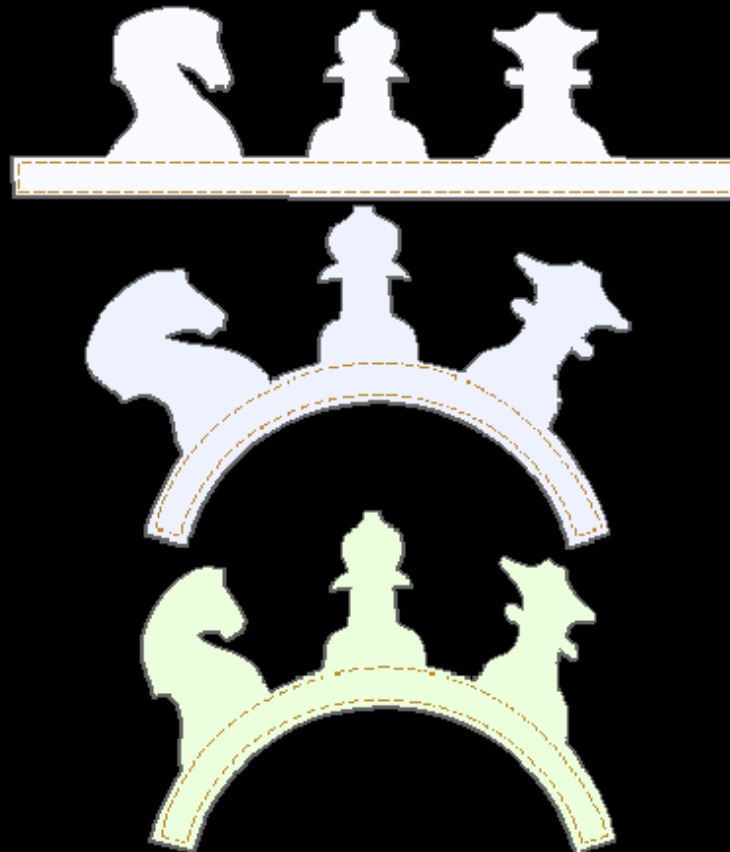
A Detail-Preserving Image Editing Example



[Perez et al. 2003]

Detail-Preserving Geometry Editing

- What is detail preservation meant for geometry?
 - Local shape preservation



Local Shape Preservation

- What is right metric for local shape preservation?
 - From a shape matching perspective
 - *Local geometric transformations need to be as close as possible to rigid body transformations + isotropic scaling*
 - From a differential geometry perspective
 - *Surfaces are 2D manifolds in a Euclidean space*
 - *Assume local tangent spaces undergo rotations only*
 - *The rotation field, \mathbf{R} , for optimal local shape preservation [Lipmann et al. 2006] :*

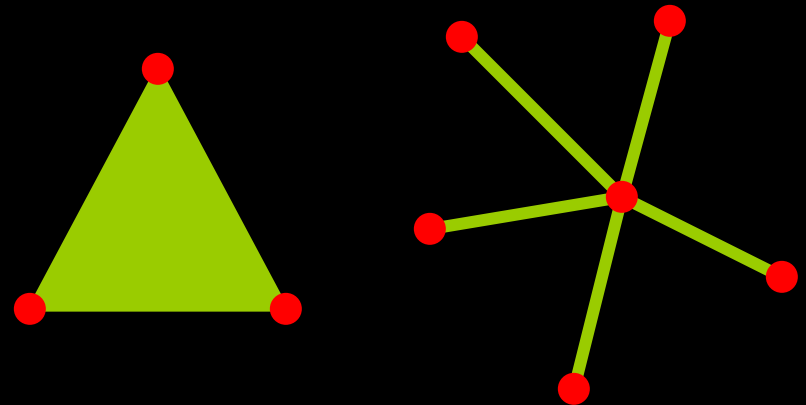
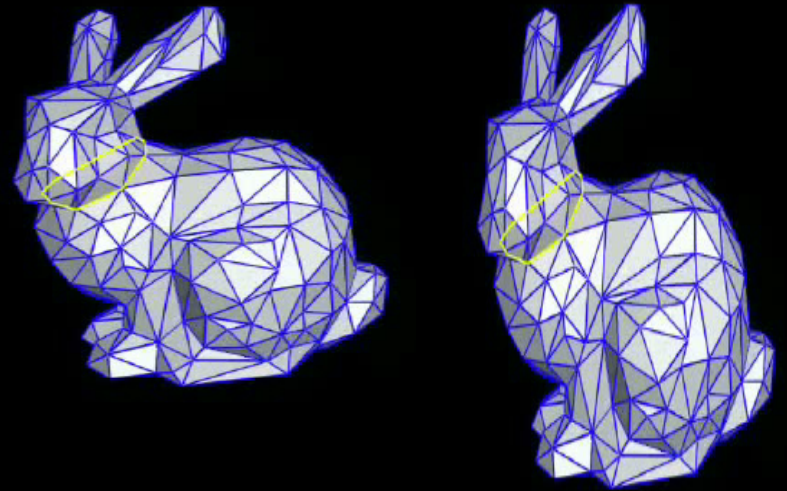
$$\operatorname{argmin}_{\mathbf{R}} \int_M \|\nabla \mathbf{R}\|_F^2 d\sigma$$

Shape-Matching vs. Elasticity

- Traditional elasticity-based models are also intended for shape preservation
- Major Differences
 - Geometric vs. Physically Based Deformations
 - Shape-matching explicitly preserves both length and angles while elasticity-based models explicitly preserve length and indirectly preserve angles
 - Shape-matching often has efficient linear or closed-form solutions while elasticity-based models give rise to expensive nonlinear systems

Local Shape Matching Criteria for Meshes

- **Face-Centric Criterion**
 - The shape of the faces should undergo as-rigid-as-possible transformations
- **Vertex-Centric Criterion**
 - The shape of the 1-ring neighborhoods should undergo as-rigid-as-possible transformations



Vector Fields and Poisson Equation

- Given a vector field \mathbf{w} , how can we approximate it using the gradient field of a scalar function?
 - Mathematically, we want to solve this minimization

$$\min_{\phi} \iint_{\Omega} \|\nabla \phi - \mathbf{w}\|^2 dA$$

- The Poisson equation solves the same problem.
 - It recovers an unknown scalar function from a given vector (guidance) field and a boundary condition.

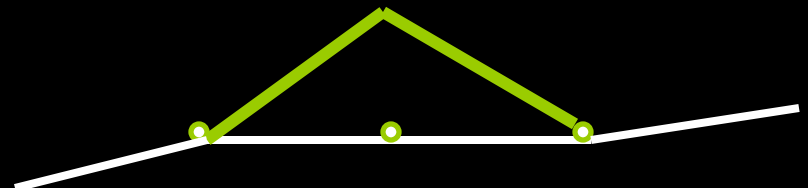
$$\Delta \phi = \nabla \cdot (\nabla \phi) = \nabla \cdot \mathbf{w}, \quad \phi|_{\partial\Omega} = f^*|_{\partial\Omega}$$

$$\Delta \phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \frac{\partial^2 \phi}{\partial z^2}, \quad \nabla \cdot \mathbf{w} = \frac{\partial w_x}{\partial x} + \frac{\partial w_y}{\partial y} + \frac{\partial w_z}{\partial z}$$

Discrete Fields on Triangle Meshes

[Polthier and Preuss 2000]

- Field definitions over discrete irregular grids
- Discrete Vector Fields
 - Piecewise constant vector fields, i.e. a constant vector within each triangle. The vector is coplanar with the triangle.
- Discrete Potential Fields
 - Piecewise linear potential fields, i.e. the potential is a linear combination of piecewise-linear basis functions.
 - $$\phi(\mathbf{x}) = \sum_i \phi_i B_i(x)$$
 - where the weights for the bases are defined at the vertices of the grid.



Poisson Equation on Triangle Meshes

[Tong *et al.* 2003]

- A Poisson equation for discrete fields on triangle meshes can be defined.

$$\text{Div} (\nabla \phi) = \text{Div} \mathbf{w},$$

$$(\text{Div} \mathbf{w})(\mathbf{v}_i) = \sum_{T_k \in N(i)} \nabla B_{ik} \cdot \mathbf{w} |T_k|$$

- It essentially has the same properties as the original Poisson equation.
- It is actually a sparse linear system:

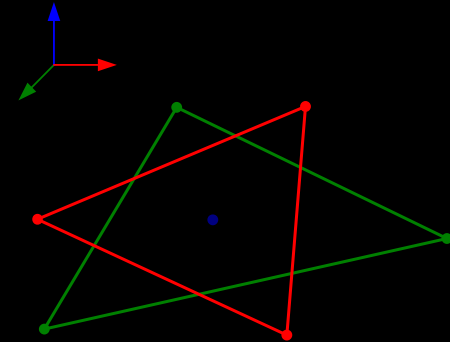
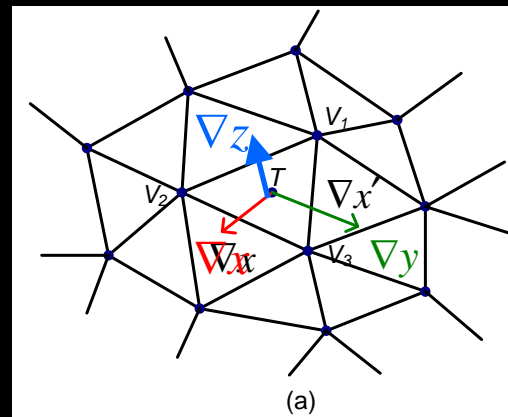
$$\mathbf{Ax} = \mathbf{b}$$

- How can we apply this Poisson equation to mesh geometry?

A Basic Poisson Mesh Solver

- Each of the x , y or z coordinates over a mesh is a piecewise-linear function defined over itself.
 - The respective coordinate of the vertices are the weights for the basis functions.
- The gradient of such functions are piecewise constant vector fields.

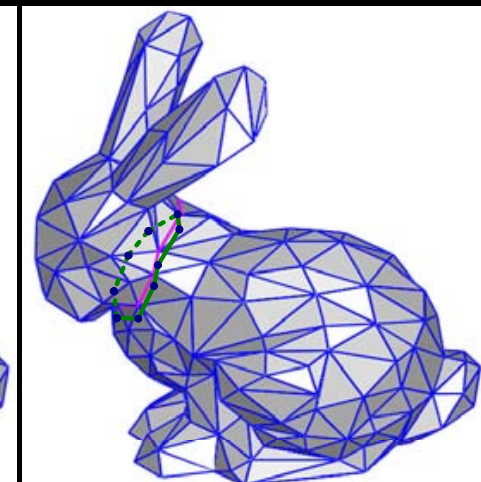
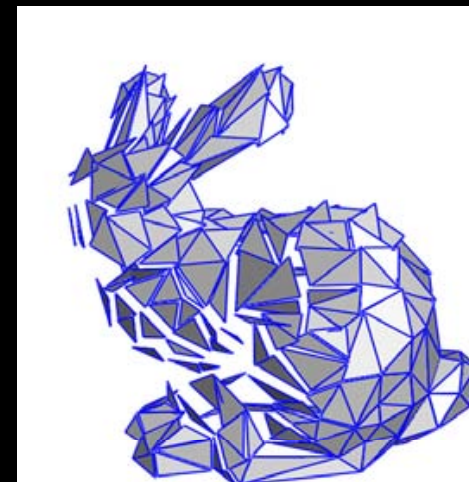
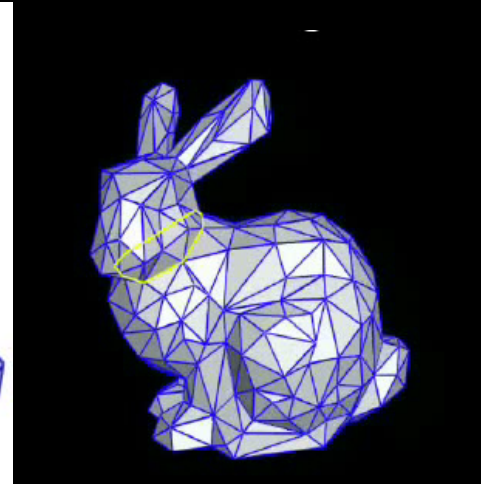
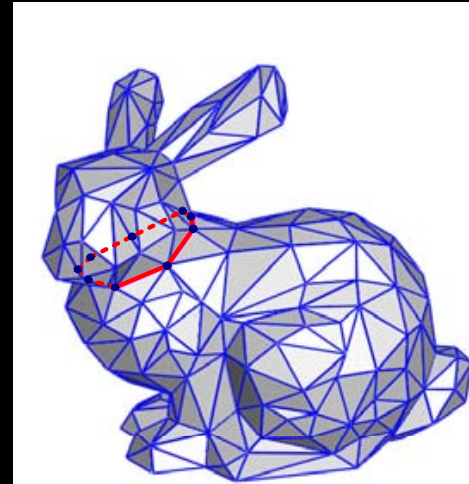
3 vectors per triangle
They are coplanar
with the triangle



- **Key observation:** If we modify these vector fields, new potential fields (vertex coordinates) can be reconstructed using the Poisson equation. That is, a new mesh is generated!
How to modify the vector fields?

Editing Gradient Fields Using Local Transforms

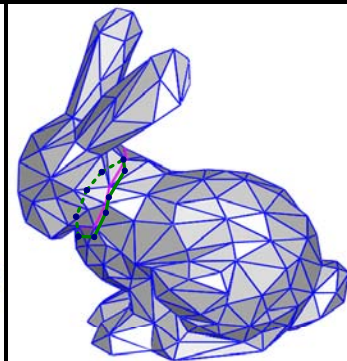
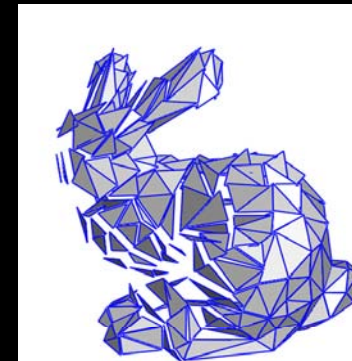
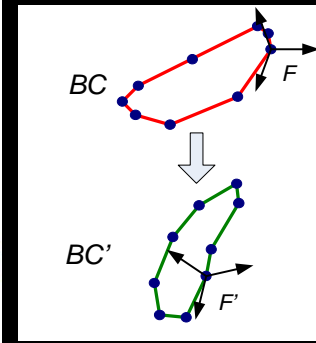
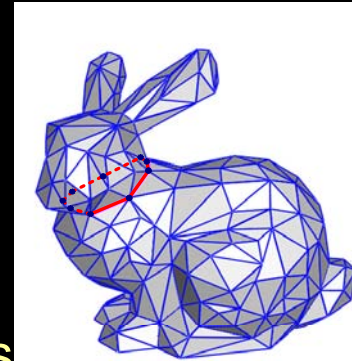
- The 3 vectors on each triangle should be modified at the same time to produce geometric meanings.
- Local transforms are applied to the triangles whose gradients produce new vectors
 - Local transforms include rotation and isotropic scaling
- Where do the local transforms come from?



Propagating Local Transforms from Boundaries

- A single curve:

- At each vertex on the curve, obtain a local 3D transform.
- Simplest scheme: for every free vertex in the editable region of the mesh, find the nearest vertex on the curve and “steal” its local transform.
- Filtering schemes: *Uniform, Linear or Gaussian*
- The required distance transform is computed by the Level Set method [Sethian 1999].

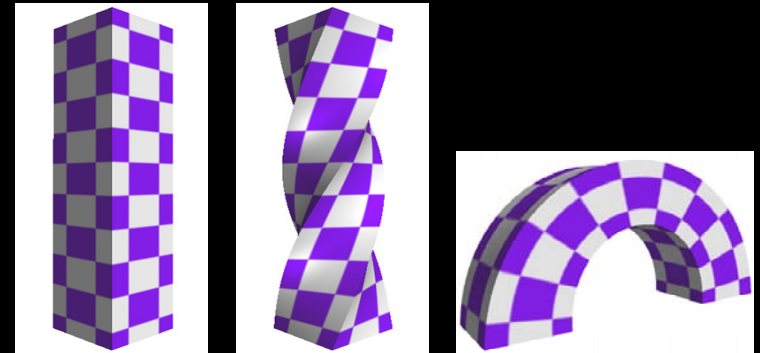


- Multiple Curves

- Weighted average of the transforms from all curves. Rotations are represented by quaternions.
- Weights based on distance: *constant, linearly decreasing, or a cosine wave*

Mesh Editing: Deformation

- Curve- or vertex-driven boundary condition editing
- Simultaneous editing with minimal user interaction
 - Translation, rotation and/or scaling applies to all vertices on the same curve.
 - Simultaneous rotation of all the vertex normals around their respective tangents
- Individual editing
 - Local transforms apply to individual vertices to achieve small scale changes.



4000 triangles, 578ms & 609ms



2480 triangles, 230ms & 240ms



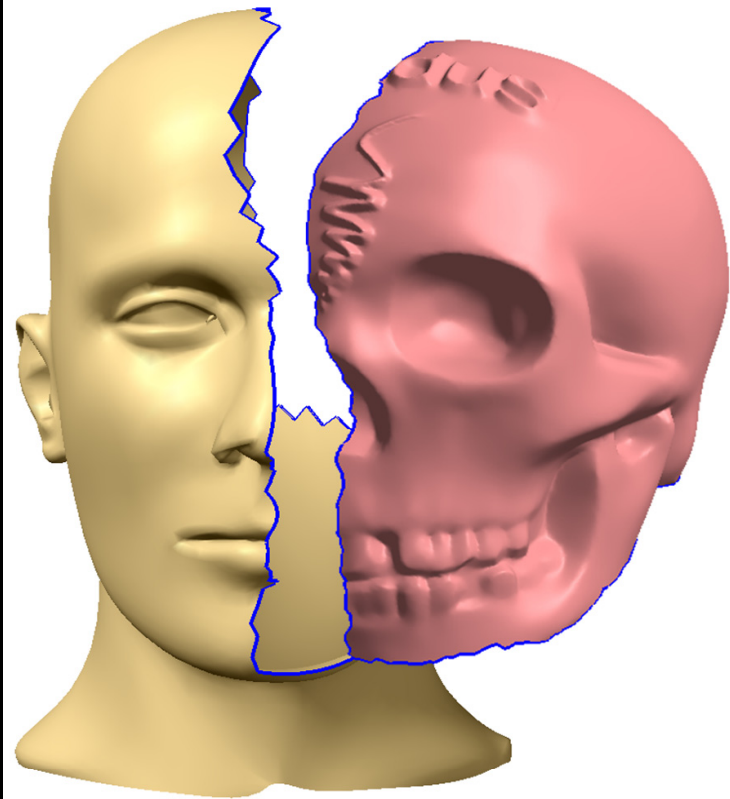
Mesh Editing: Object Merging

- Steps for merging two partial meshes at their open boundaries
 - Define local frames everywhere on the two boundaries;
 - Define vertex correspondence between them;
 - Generate an intermediate mesh boundary including interpolated vertex positions and local frames;
 - Consider the intermediate mesh boundary as the new boundary condition for both meshes, and perform boundary condition editing on both.
- Advantages of this Poisson-based method
 - The original mesh boundaries can be very different.
 - Local transform propagation globally adjusts the meshes to make them compatible with each other.

An Object Assembled from Parts



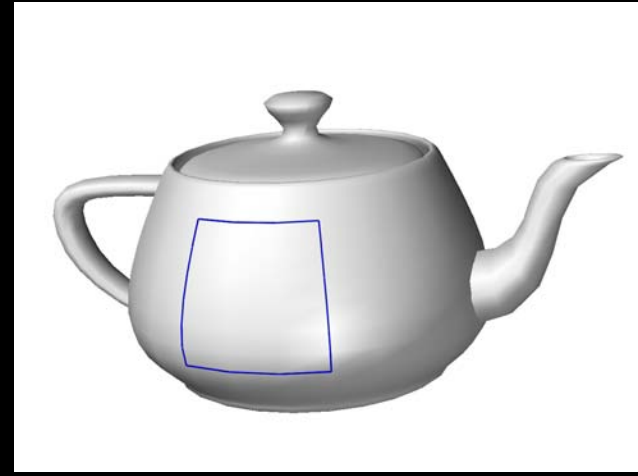
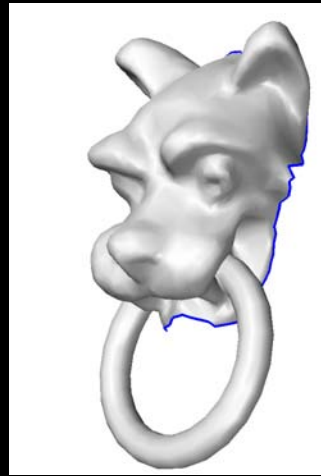
Object Merging Using Sparse Correspondences



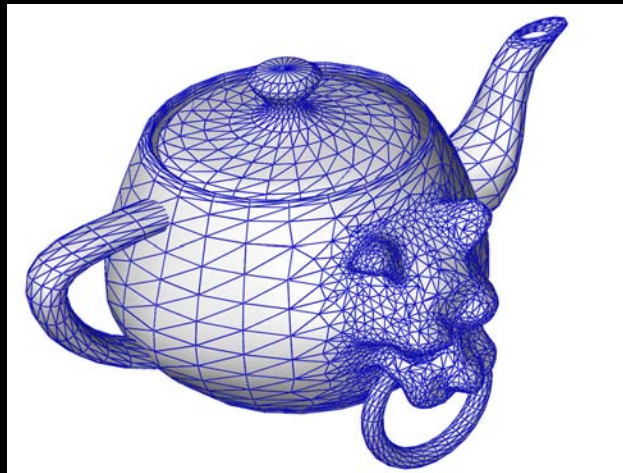
Dense correspondences are obtained from interpolation based on the arc length on the boundaries.

Merging High Genus Meshes

Input



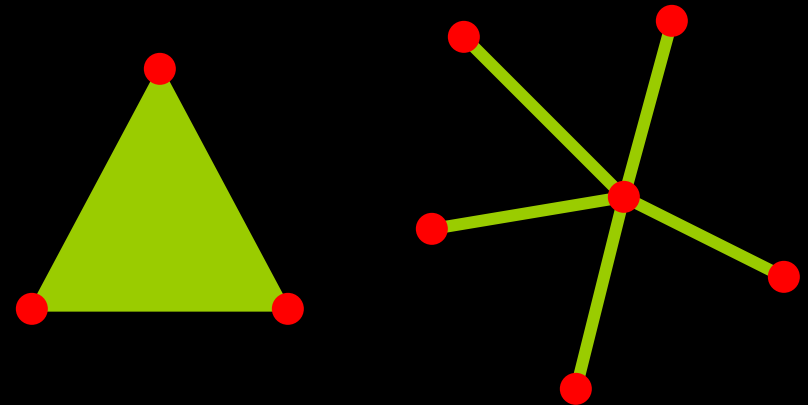
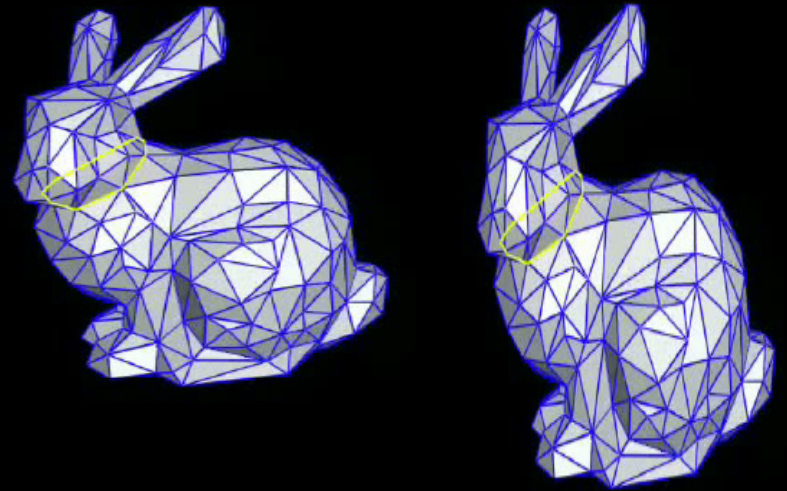
Output



GARGOYLE: 4000 triangles, TEAPOT: 2000 triangles, Running Time: 890ms

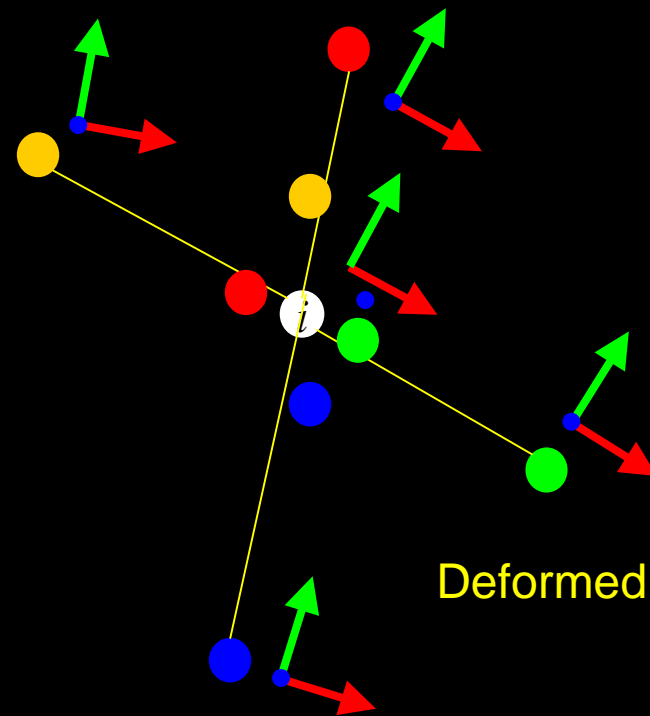
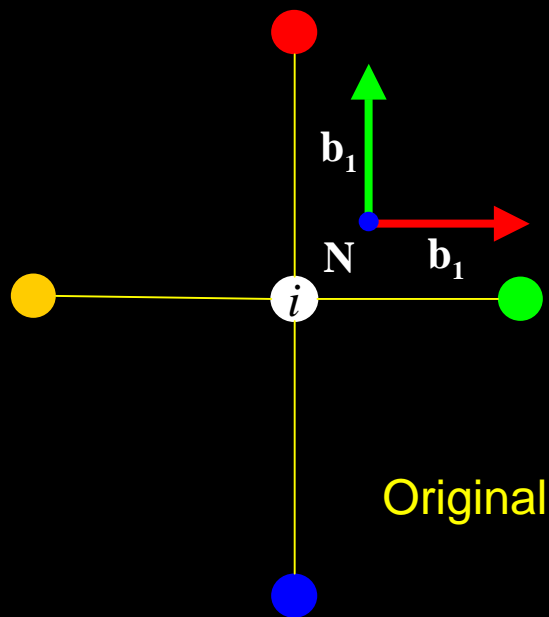
Local Shape Matching Criteria for Meshes

- **Face-Centric Criterion**
 - The shape of the faces should undergo as-rigid-as-possible transformations
- **Vertex-Centric Criterion**
 - The shape of the 1-ring neighborhoods should undergo as-rigid-as-possible transformations



Vertex-Based Mesh Deformation

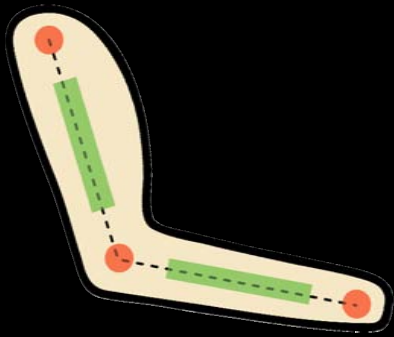
- Set up a local frame for every vertex, compute rotated local frames, and preserve the local coordinates of the edges in these new local frames [Lipman et al. 2005]



$$\sum_{j \in N(i)} (\mathbf{x}_i - \mathbf{x}_j - \mathbf{d}_{ji}) = 0$$

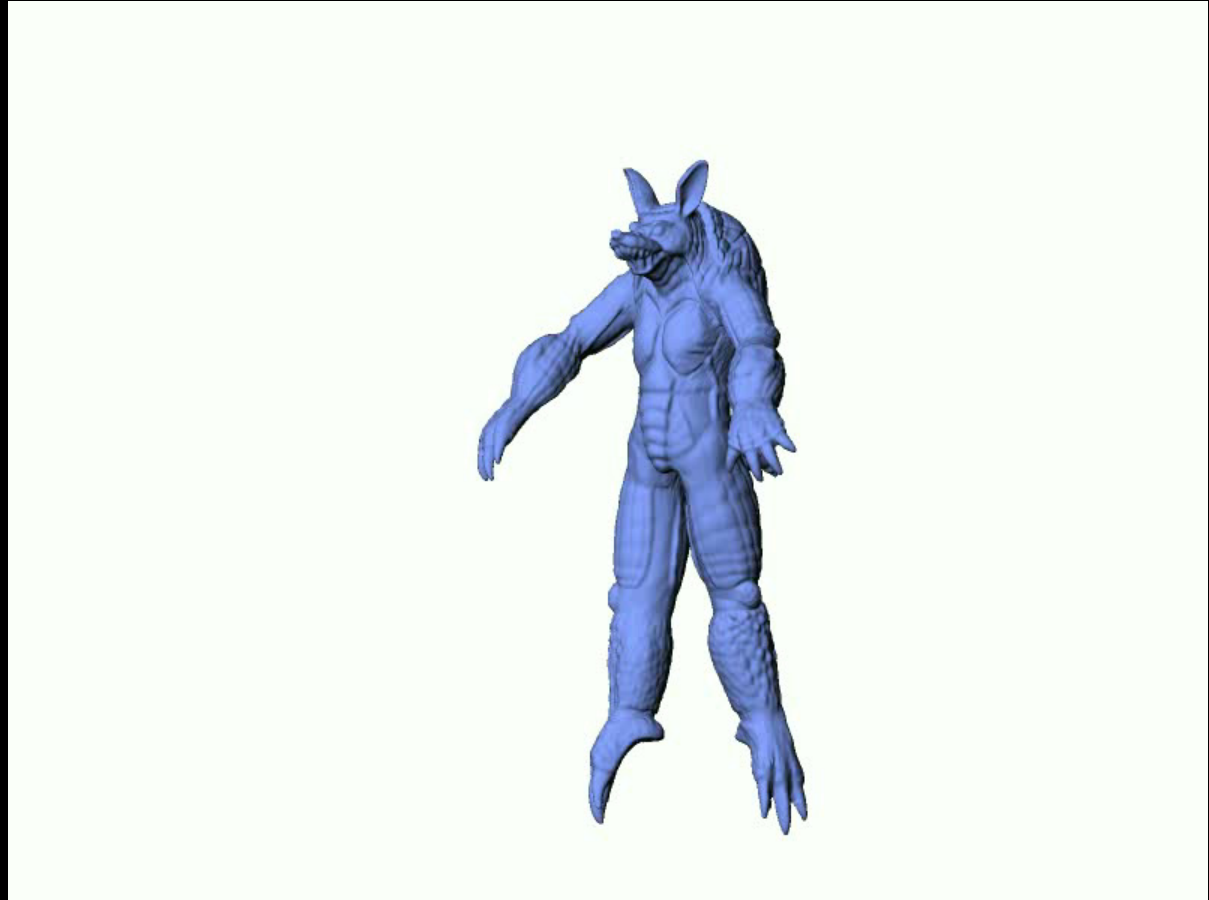
$$\mathbf{d}_{ji} = \frac{1}{2} (\mathbf{R}_i + \mathbf{R}_j) (\mathbf{v}_i^0 - \mathbf{v}_j^0)$$

A Deformation Example Driven by MOCAP Data



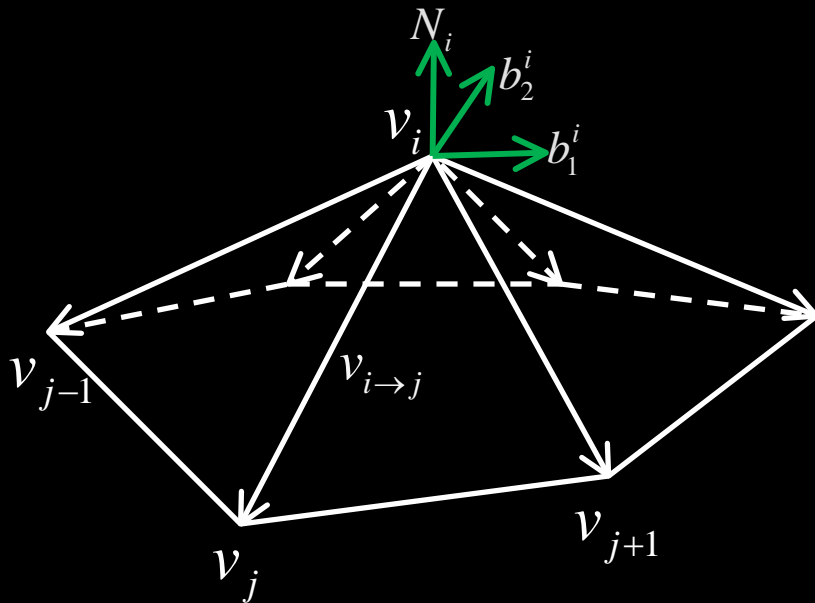
Green: rotation
constraints

Red: position
constraints



Alternating Least-Squares Mesh Deformation Solver

- High quality mesh deformation based on rotation-invariant coordinates:



$v_{i \rightarrow j}$

Rotation invariant
coordinates

The local coordinates of v_j in the local frame $\mathbf{R}_i = \{b_1^i, b_2^i, N^i\}$ of v_i in the original mesh

Alternating Least-Squares Mesh Deformation Solver

- To preserve the rotation invariant coordinates, we desire that the deformed vertex \hat{v}_i satisfies:

$$\hat{v}_j - \hat{v}_i = \hat{\mathbf{R}}_i v_{i \rightarrow j}, j \in N_i$$



Collect it from each vertex and consider position constraints

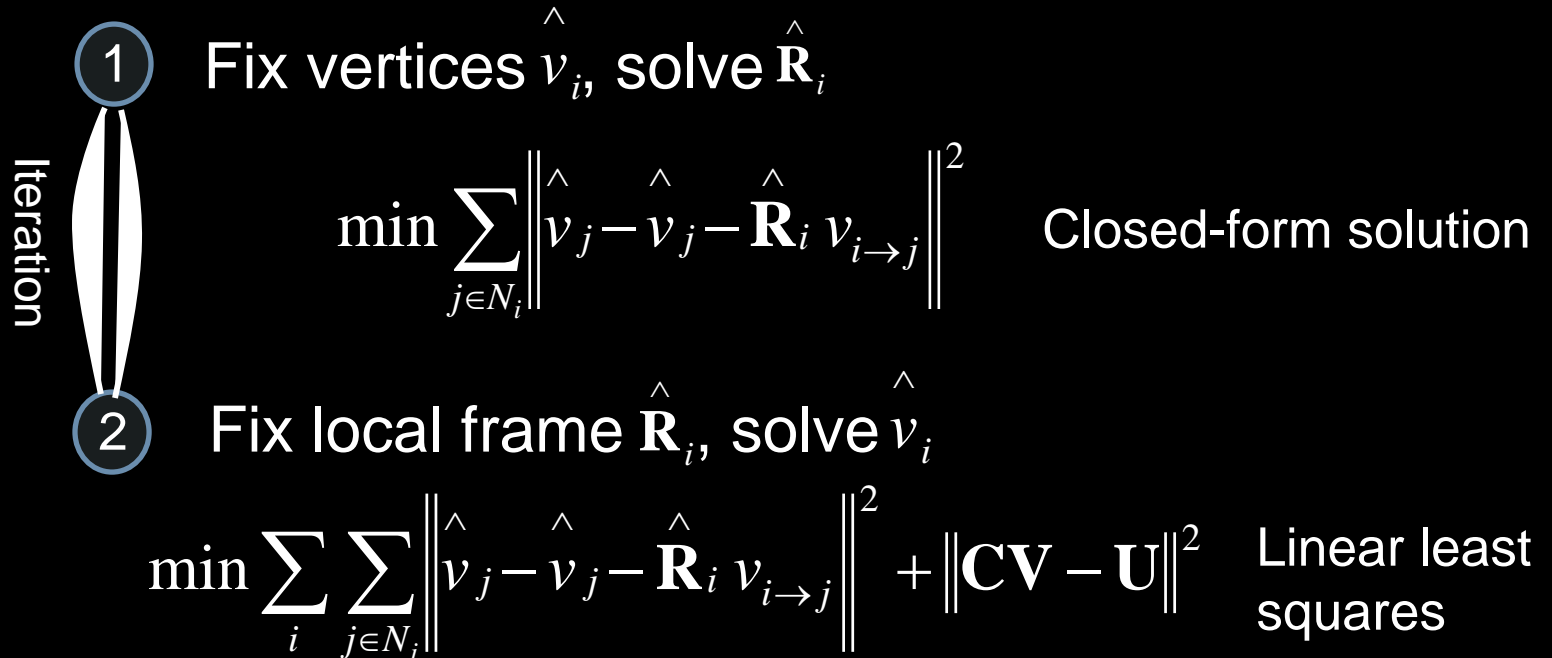
$$\min \sum_i \sum_{j \in N_i} \left\| \hat{v}_j - \hat{v}_i - \hat{\mathbf{R}}_i v_{i \rightarrow j} \right\|^2 + \left\| \mathbf{C}\mathbf{V} - \mathbf{U} \right\|^2$$

\mathbf{U} : position constraints

nonlinear least-squares
deformation energy function

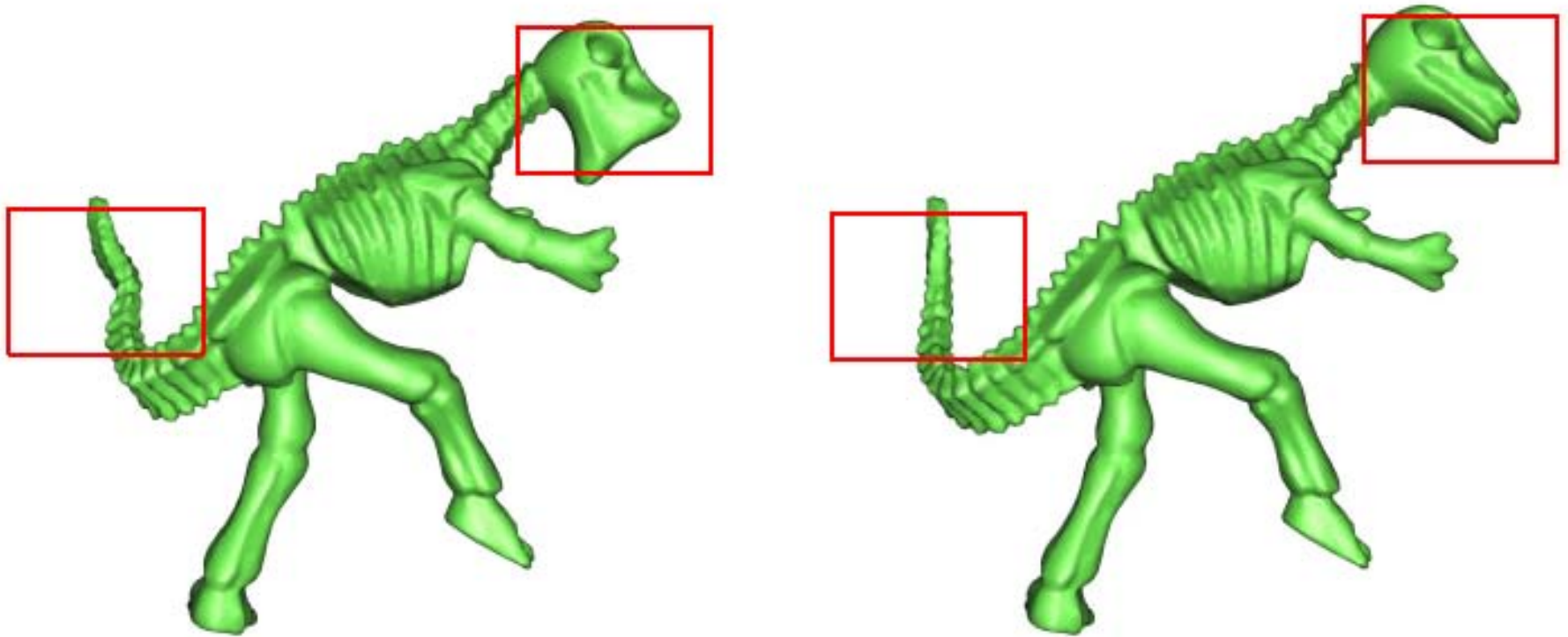
Alternating Least-Squares Mesh Deformation Solver

- Alternative solver – two simpler steps



Initial solution from subspace mesh deformation solver

Alternating Least-Squares Mesh Deformation Solver



Subspace[Huang et.al]

Alternating Least Square Solver

Editing Deforming Mesh Sequences

Online

Key-frame editing



Offline

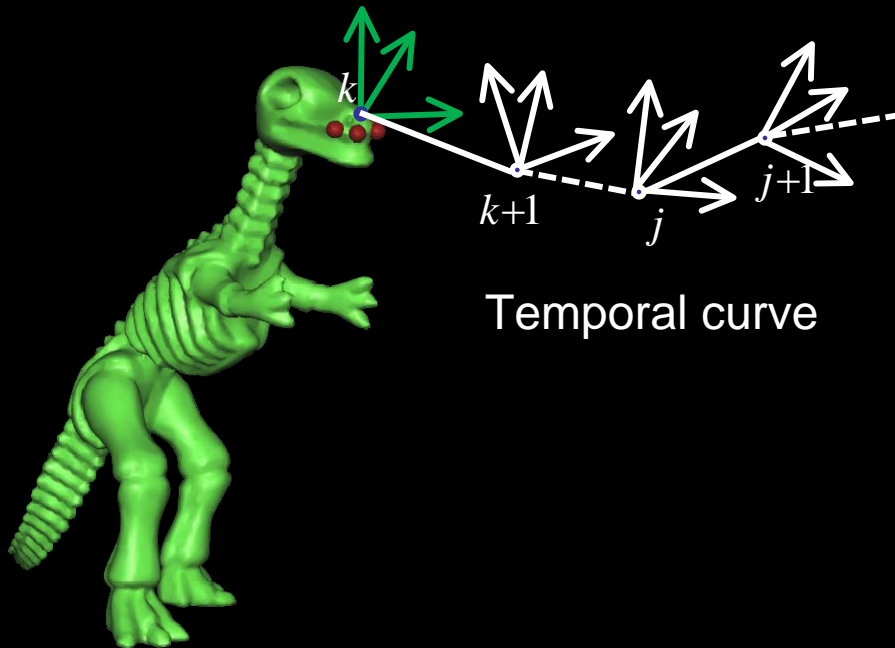
Handle trajectory editing



Novel mesh sequence
generation

Handle Trajectory Editing

- The propagated handles serve as new boundary conditions at intermediate frames for the solver
- Handle trajectory is actually a temporal curve:



Temporal curve

$$\mathbf{H}_i^k = \left\{ \mathbf{c}_i^k, \mathbf{F}_i^k, \left\{ \mathbf{v}_{i_m}^k \mid m=0, \dots, n_p - 1 \right\} \right\}$$

$\left\{ \mathbf{v}_{i_m}^k \mid m=0, \dots, n_p - 1 \right\}$: the set of vertices

\mathbf{c}_i^k : frame origin

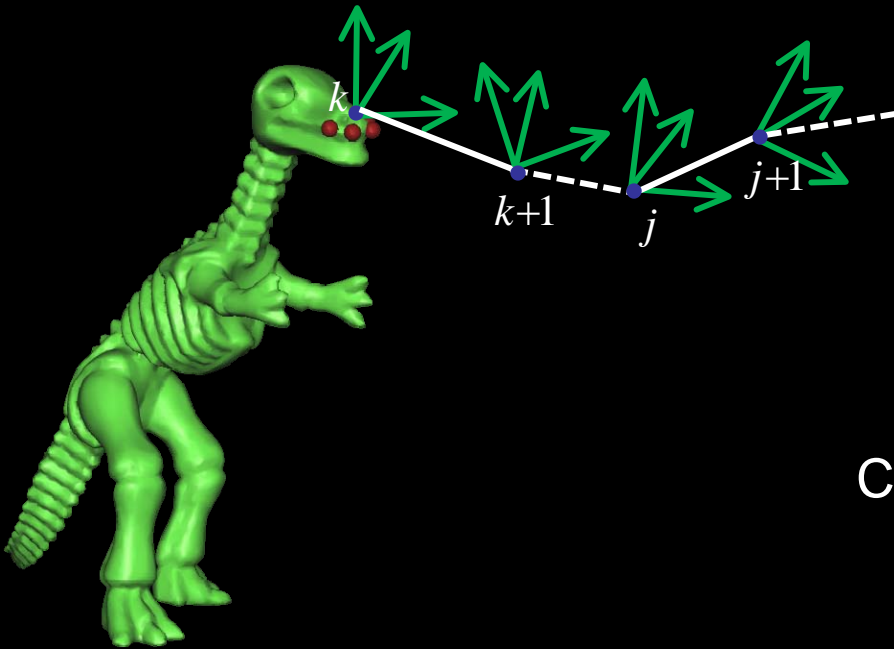
\mathbf{F}_i^k : frame axes

i : handle number

k : frame number

Handle Trajectory Editing

- Rotation Invariant Representation of a Temporal Curve



$$c_i^l - c_i^k = \mathbf{F}_i^k c_i^{k \rightarrow l} \quad l \in N_k$$

Rotation invariant



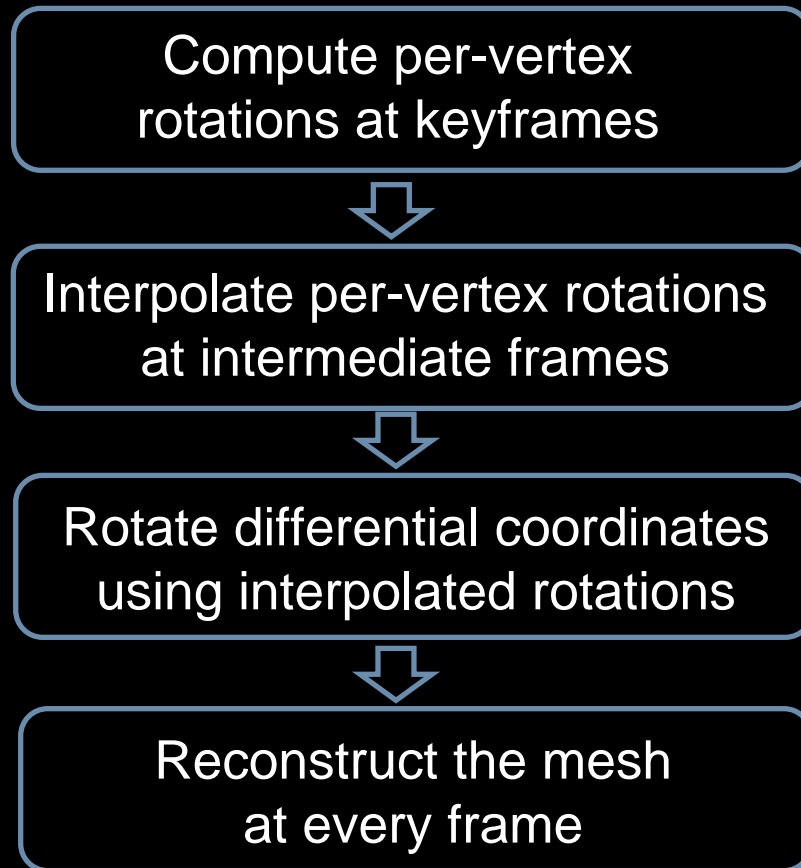
Curve editing energy function:

$$\sum_k \sum_{l \in N_k} \left\| c_i^{\wedge l} - c_i^{\wedge k} - \mathbf{F}_i^{\wedge k} c_i^{k \rightarrow l} \right\|^2$$

Where \wedge : means edited frame origin or axes

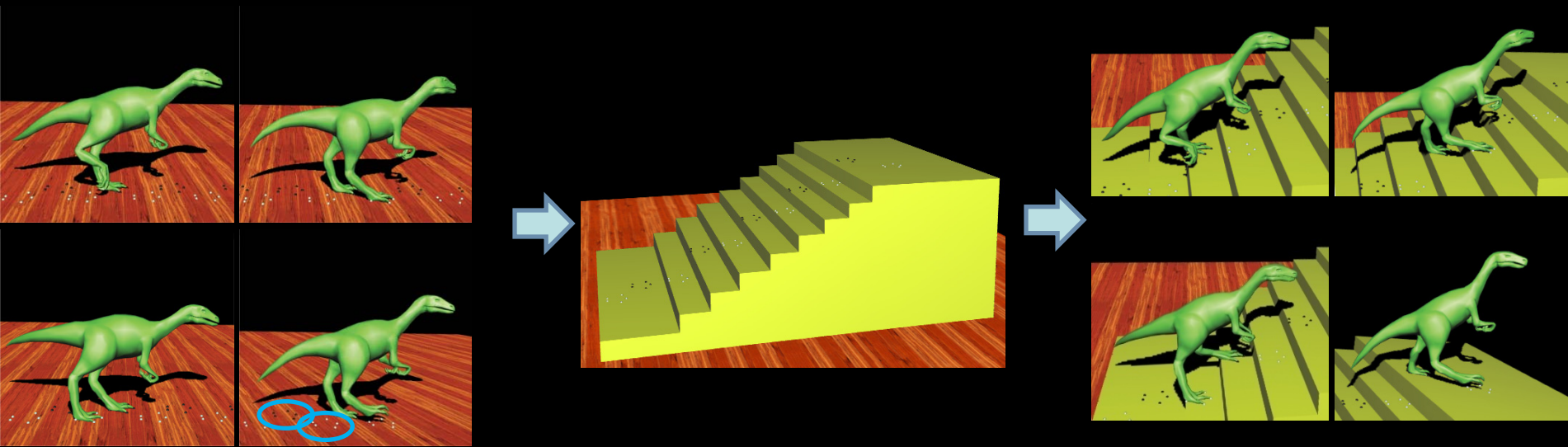
Novel Mesh Sequence Generation

- Offline process



Application: Footprint Editing

- Steps



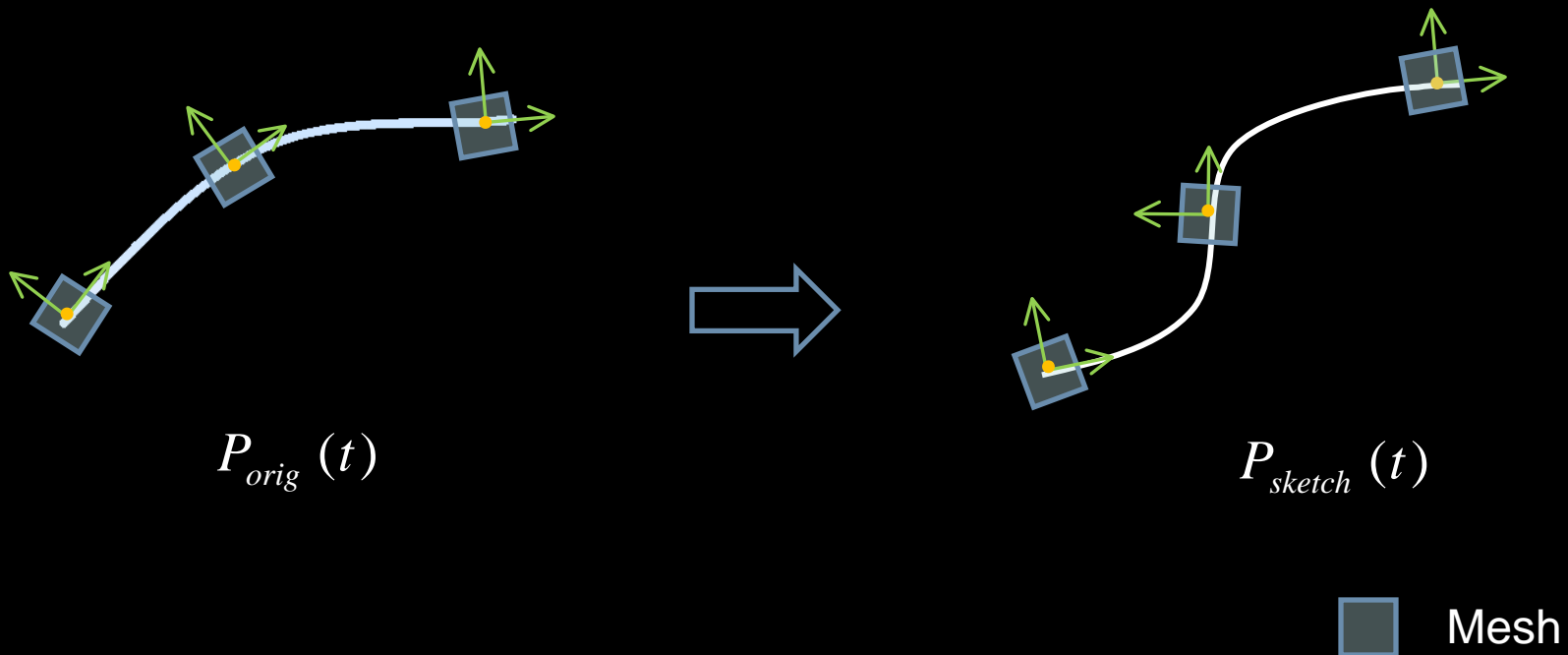
- Dedicated to walking or running mesh animations
- Use footprints to automatically capture the constraints that should be satisfied during walking or running

Demo: Footprint Editing

Footprint Editing

Application: Path Editing

- A sketch user interface for editing the motion paths of deforming mesh sequences



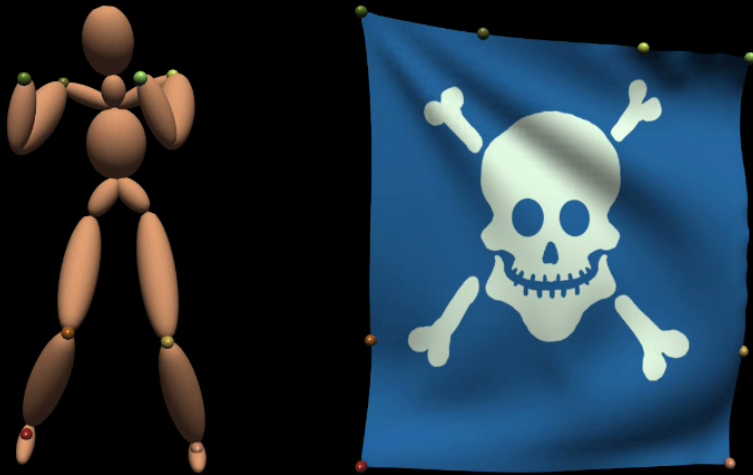
Demo: Path Editing

Application: Deformation Mixing

- Different from deformation transfer, deformation mixing mixes the large-scale deformations of a first deforming mesh sequence with the small-scale deformations of a second deforming mesh sequence to create a novel sequence.

Handle Based Deformation Mixing

- Steps



Specify corresponding handles



Transfer the motion trajectories of handles from the first sequence to second



Set the transferred trajectory as constraints of the second sequence and invoke the solver

Demo: Deformation Mixing

A cloth animation is mixed
with a running motion

Application: Mesh Deformation Filtering

- Edit the deforming mesh sequence in frequency domain
- The filter is applied to the temporal trajectories of handle centers, **not** the trajectory of every vertex, for the purpose of shape detail preservation

Mesh Deformation Filtering

Specify handles and their hierarchy



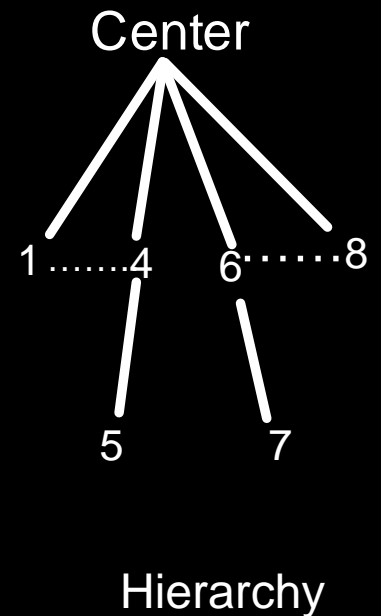
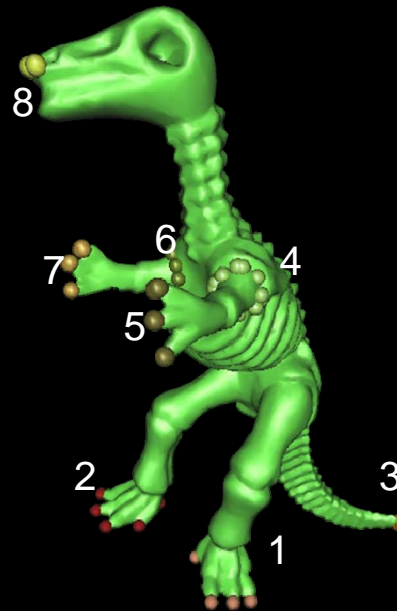
Transform handle trajectory to the local frame of its parent



Apply a filter to the trajectory and compute the new handle positions



Invoke the solver to get filtered mesh sequence

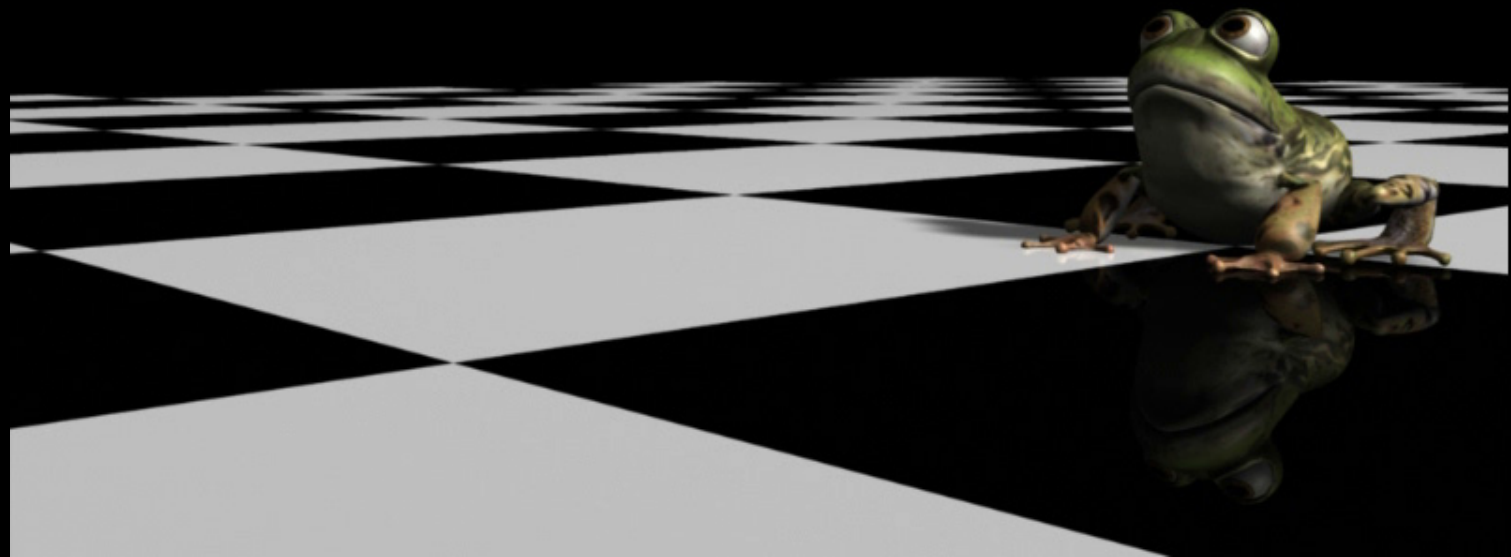


Mesh Deformation Filtering

- Demo – Cartoon animation filter

Mesh Deformation Filter

Demo: Revisiting Bunny



One Problem in Surface-Based Mesh Deformation

- The numerical solver becomes the bottleneck of the entire system when processing large meshes.
- Existing solutions restrict the scope of editing operations.
- **Our solution:** a fast multigrid algorithm
 - Good performance and scalability
 - No significant setup time
 - *user constraints can be updated on the fly*
 - Low memory usage
 - Related work:
 - *[Aksoylu et al. 2005, Clarenz et al. 2004, Ray and Levy 2003]*

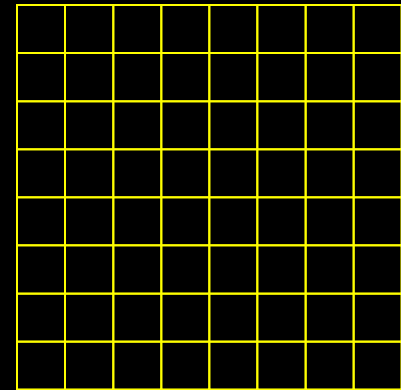
Brief Review: Basic Multigrid I

Original Equation: $L_h u_h = f_h$ $\xrightarrow{\text{dashed}} \text{RHS}$

Discretized differential operator Unknowns

Error smoothing: Jacobi or Gauss-Seidel Relaxation

$$u_h^m \xrightarrow{\text{Smooth}} u_h^{m+1}$$

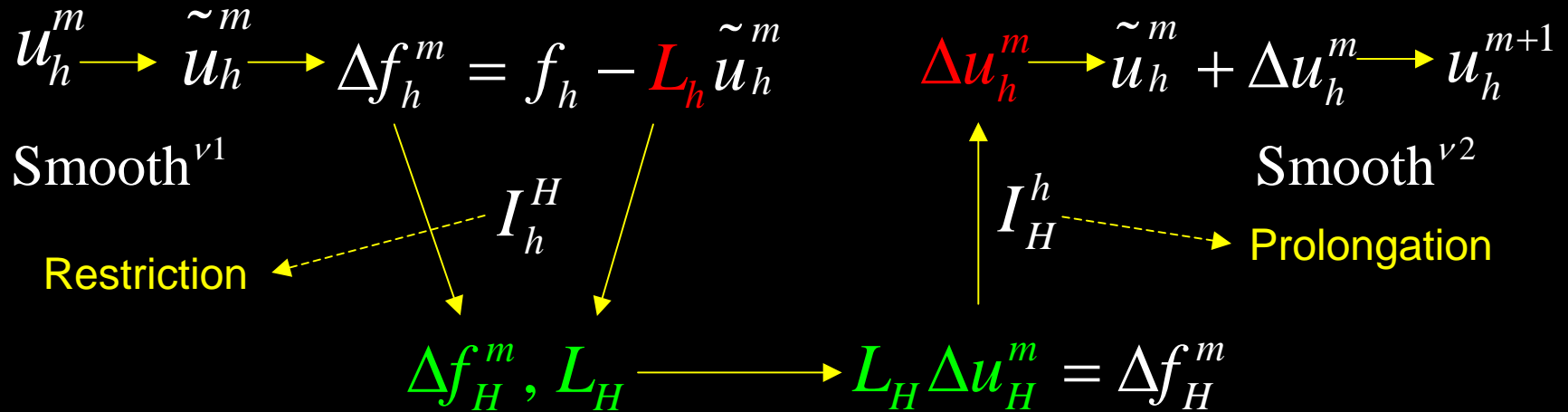
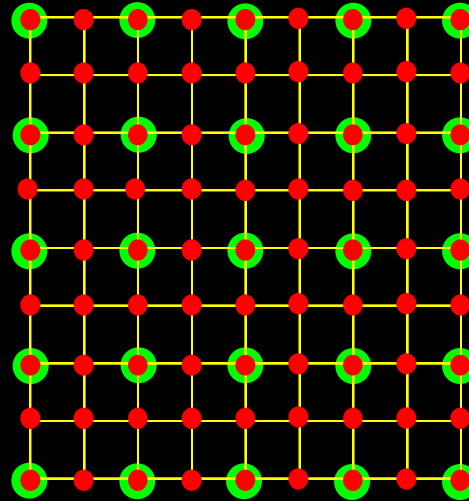


Defect Equation:

$$L_h \Delta u_h = \Delta f_h \left(= f_h - L_h u_h \right)$$

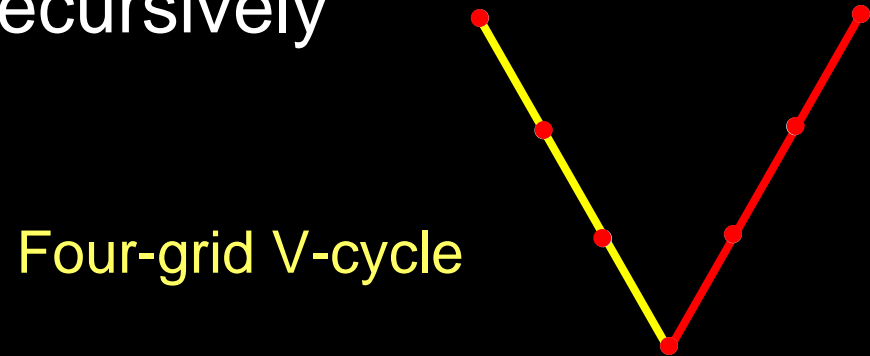
Brief Review: Basic Multigrid II

- Two-grid cycle

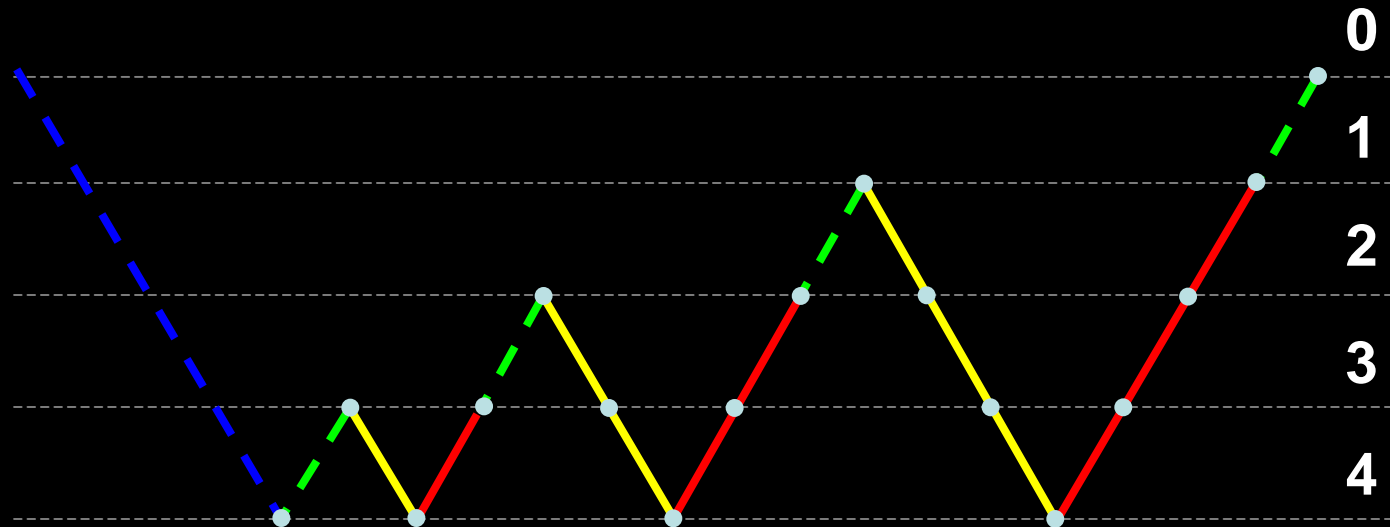


Brief Review: MG vs FMG

- Apply the same idea recursively



- An initial approximation can be obtained by nested iterations.



Multigrid for Unstructured Mesh Deformation

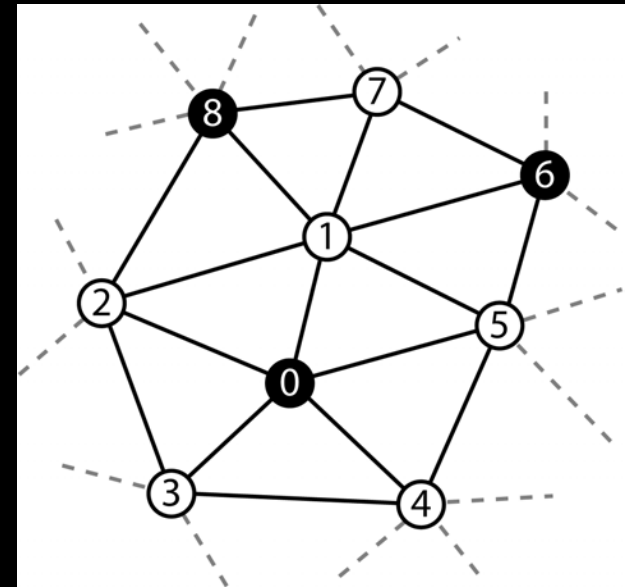
- Solver at the Coarsest Level
 - SuperLU
 - The number of vertices at the coarsest level is ~2500.
- Smoothing
 - Gauss-Seidel Relaxation
- **Graph Coarsening**
- **Efficient Prolongation/Restriction Operators on Graph Structures**

Graph Coarsening

- Vertices in coarser levels are chosen using *maximal δ -independent vertex set*:
 - A subset of vertices, V_{ind}^l , is δ -independent iff

$$\forall \mathbf{v}_i^l, \mathbf{v}_j^l \in V_{ind}^l, e_{ij}^l \notin E \quad \text{or} \quad \|\mathbf{v}_i^l - \mathbf{v}_j^l\| > \delta$$

- \mathbf{v}_i^{l+1} and \mathbf{v}_j^{l+1} are connected if \mathbf{v}_j^l is within the 2-ring neighborhood of \mathbf{v}_i^l
- The sparsity structure remains the same across different levels



Deformation Equation with Constraints and the Smoothing Operator

$$\sum_{j \in N(i)} \mathbf{x}_i - \mathbf{x}_j - \mathbf{d}_{ji} = 0 \quad \longrightarrow \quad \sum_{j \in N(i) \setminus C(i)} w_{ji} (\mathbf{x}_i - \mathbf{x}_j - \mathbf{d}_{ji}) + \alpha_i \mathbf{x}_i = \boldsymbol{\beta}_i$$

$$\mathbf{d}_{ji} = \frac{1}{2} (R_i + R_j) (\mathbf{v}_i^0 - \mathbf{v}_j^0) \quad \alpha_i = \sum_{j \in C(i)} w_{ji} \quad \boldsymbol{\beta}_i = \sum_{j \in C(i)} w_{ji} (\mathbf{x}_j + \mathbf{d}_{ji})$$

- Smoothing Operator:

$$\mathbf{x}_i = \frac{\sum_{j \in N(i) \setminus C(i)} w_{ji} (\mathbf{x}_j + \mathbf{d}_{ji}) + \boldsymbol{\beta}_i}{\sum_{j \in N(i) \setminus C(i)} w_{ji} + \alpha_i}$$

Operators for Original Equations I

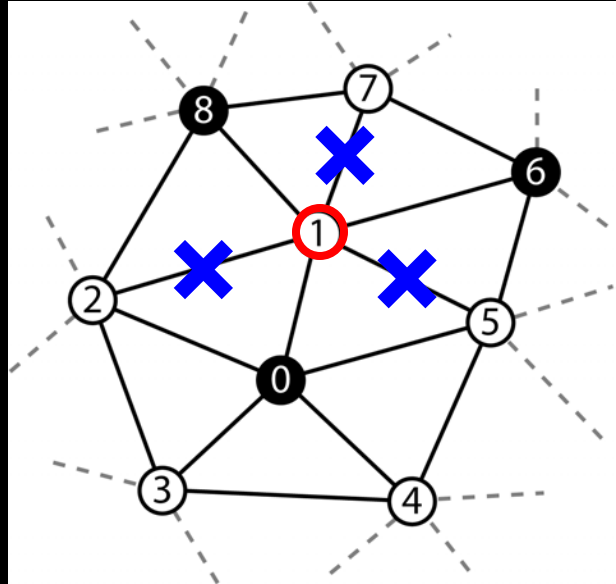
- Two distinct pairs of prolongation/restriction operators for original equations and defect equations
- Prolongation operator for the original equation (pruning):

$$\mathbf{x}_i^l = \mathbf{x}_i^{l+1}$$

If vertex i is in level $l+1$

$$\mathbf{x}_i^l = \frac{\sum_{j \in R^l(i)} w_{ji}^l (\mathbf{x}_j^{l+1} + \mathbf{d}_{ji}^l) + \boldsymbol{\beta}_i^l}{\sum_{j \in R^l(i)} w_{ji}^l + \alpha_i^l}$$

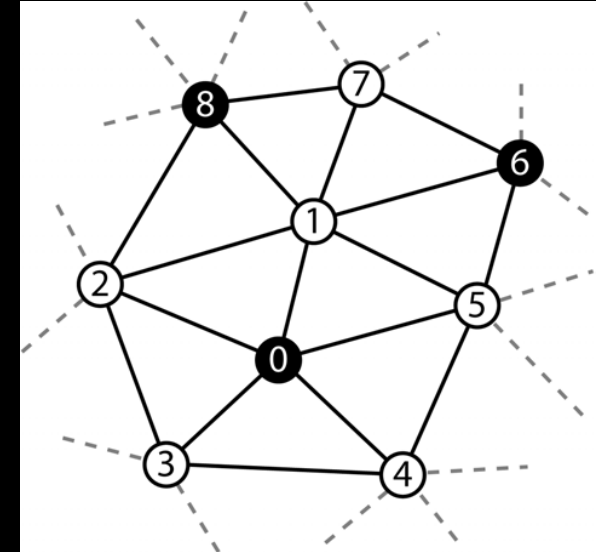
otherwise



Operators for Original Equations II

- Restriction operator for original equations:

$$\sum_{i \in N^l(j)} w_{ij}^l (\mathbf{x}_j^l - \mathbf{x}_i^l - \mathbf{d}_{ij}^l) + \alpha_j^l \mathbf{x}_j^l = \boldsymbol{\beta}_j^l$$



$$\sum_{i \in R^l(j)} w_{ij}^l (\mathbf{x}_j^{l+1} - \mathbf{x}_i^{l+1} - \mathbf{d}_{ij}^l) + \sum_{i \in K^l(j)} w_{ij}^l \left(\mathbf{x}_j^{l+1} - \left(\frac{\sum_{k \in R^l(i)} w_{ki}^l (\mathbf{x}_k^{l+1} + \mathbf{d}_{ki}^l) + \boldsymbol{\beta}_i^l}{\sum_{k \in R^l(i)} w_{ki}^l + \alpha_i^l} \right) - \mathbf{d}_{ij}^l \right) + \alpha_j^l \mathbf{x}_j^{l+1} = \boldsymbol{\beta}_j^l$$

$$\sum_{i \in N^{l+1}(j)} w_{ij}^{l+1} (\mathbf{x}_j^{l+1} - \mathbf{x}_i^{l+1} - \mathbf{d}_{ij}^{l+1}) + \alpha_j^{l+1} \mathbf{x}_j^{l+1} = \boldsymbol{\beta}_j^{l+1}$$

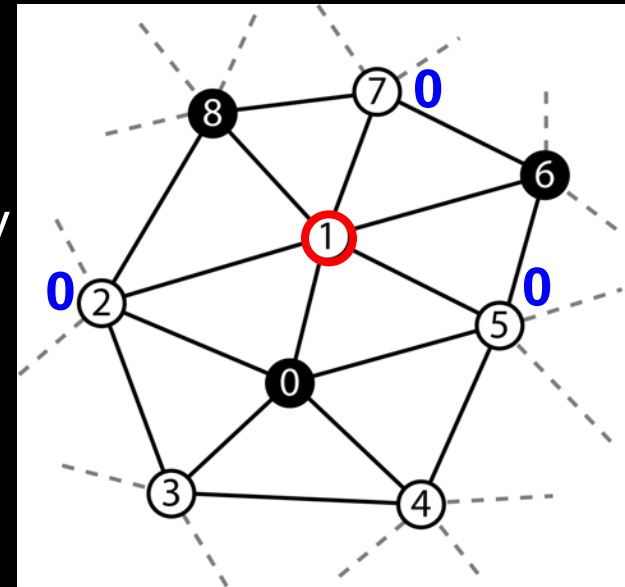
Operators for Defect Equations

- Defect equation:
$$\sum_{j \in N(i)} w_{ji}^l (\Delta \mathbf{x}_i^l - \Delta \mathbf{x}_j^l) + \alpha_i^l \Delta \mathbf{x}_i^l = \xi_i^l$$

- Prolongation operator: set zero instead of pruning

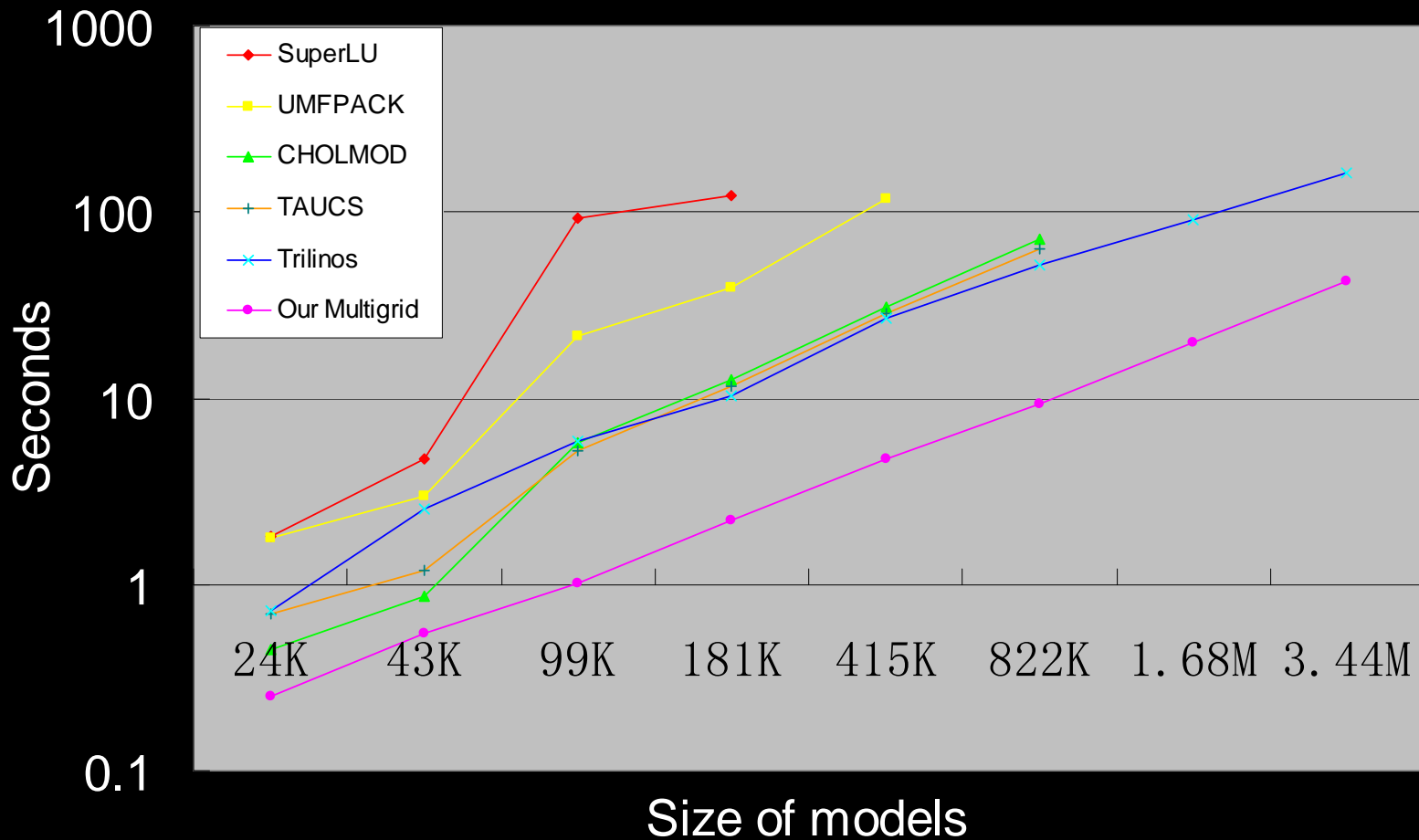
$$\Delta \mathbf{x}_i^l = \frac{\sum_{j \in R^l(i)} w_{ji}^l \Delta \mathbf{x}_j^{l+1} + \xi_i^l}{\sum_{j \in N^l(i)} w_{ji}^l + \alpha_i^l}$$

- Restriction operator can be obtained similarly



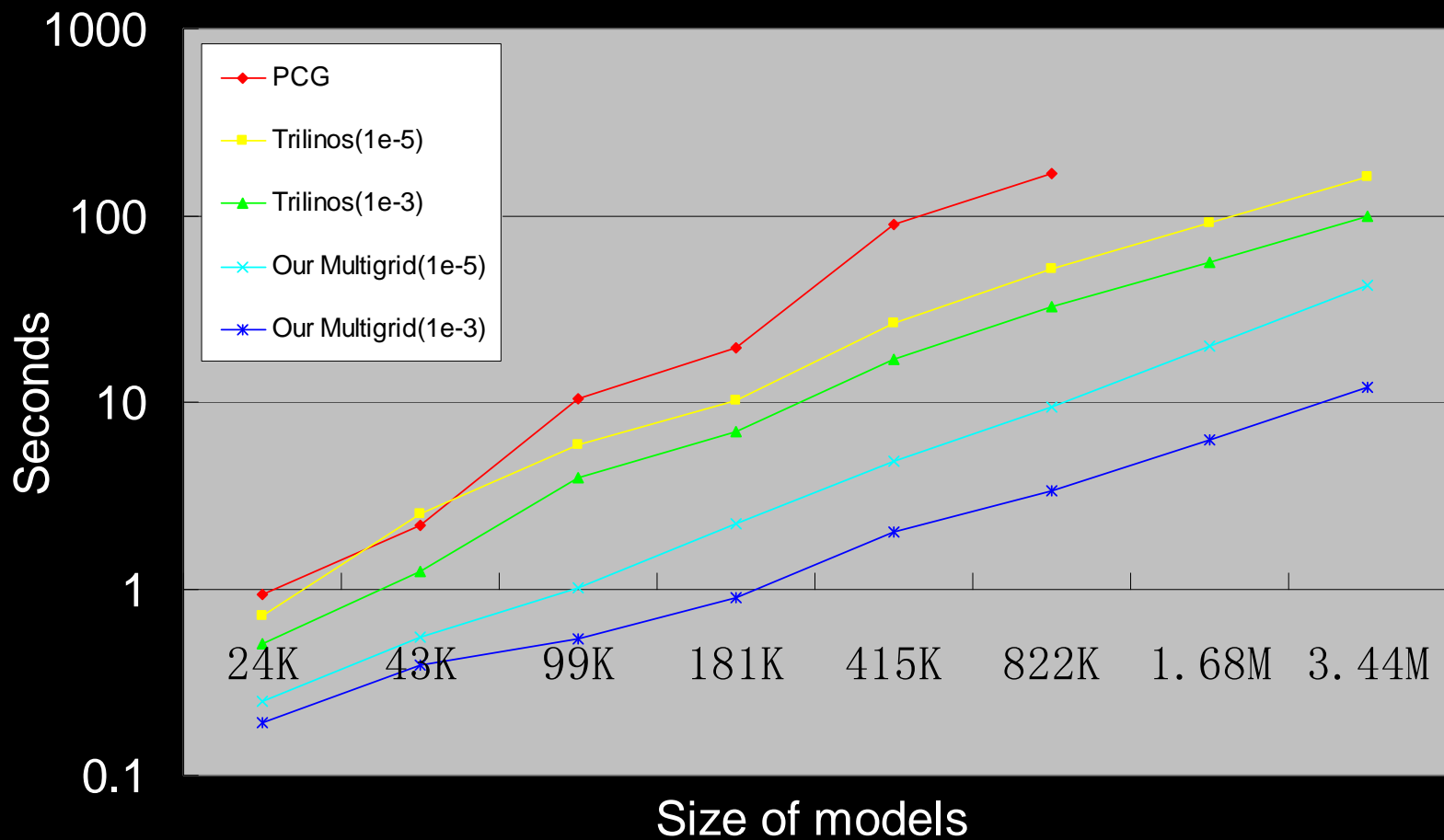
Experimental Results: Performance and Scalability

Performance statistics reported by popular solvers



Experimental Results: Tradeoff Precision for Speed

Performance statistics reported by popular solvers



Experimental Results: Large-Scale Composite Mesh Editing





Armadillo: 525k

Female: 415k

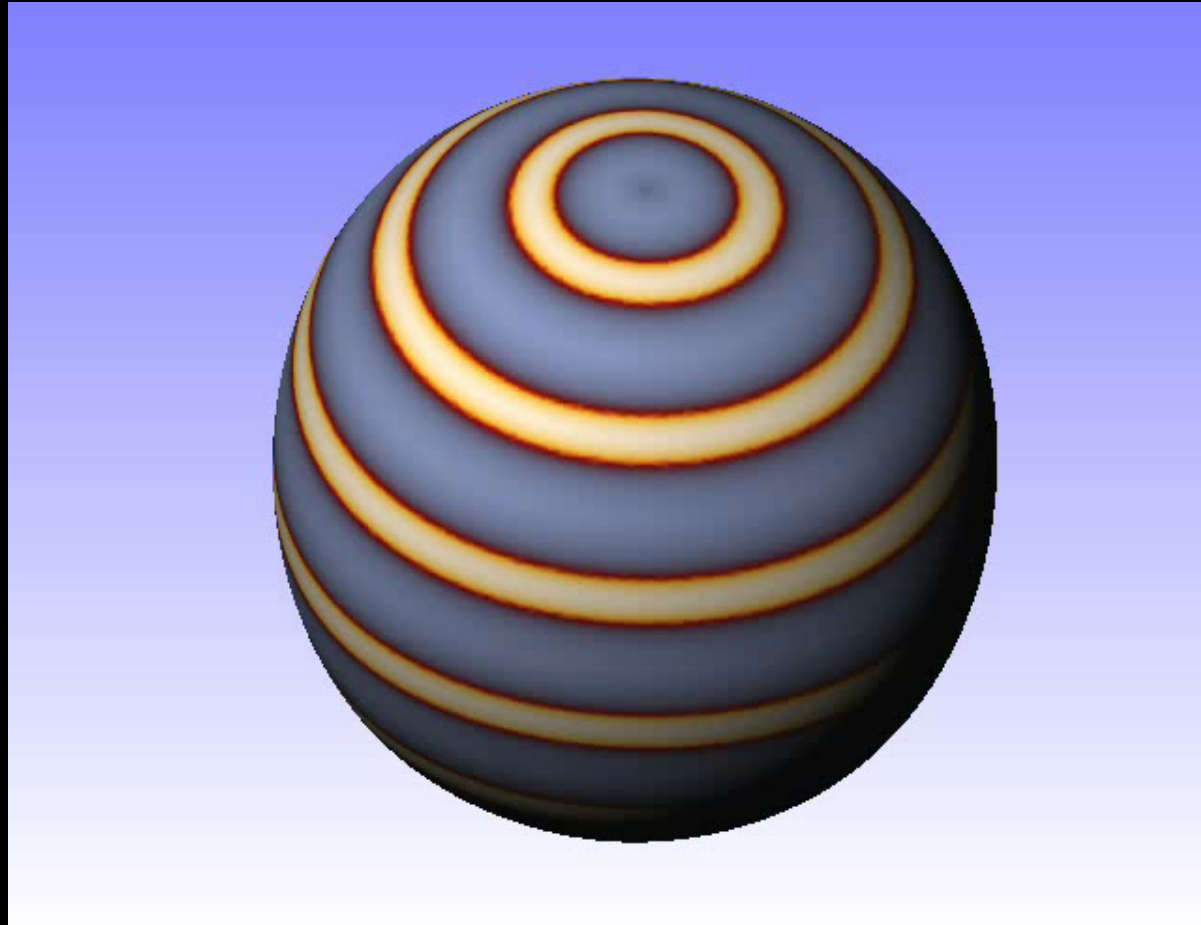
Camel: 100k

Mesh Animations

Generated from
MOCAP data



Dynamic Textures: Surface Flow Simulation



Navier-Stokes Equations

Inviscid, incompressible Navier-Stokes equations:

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0$$

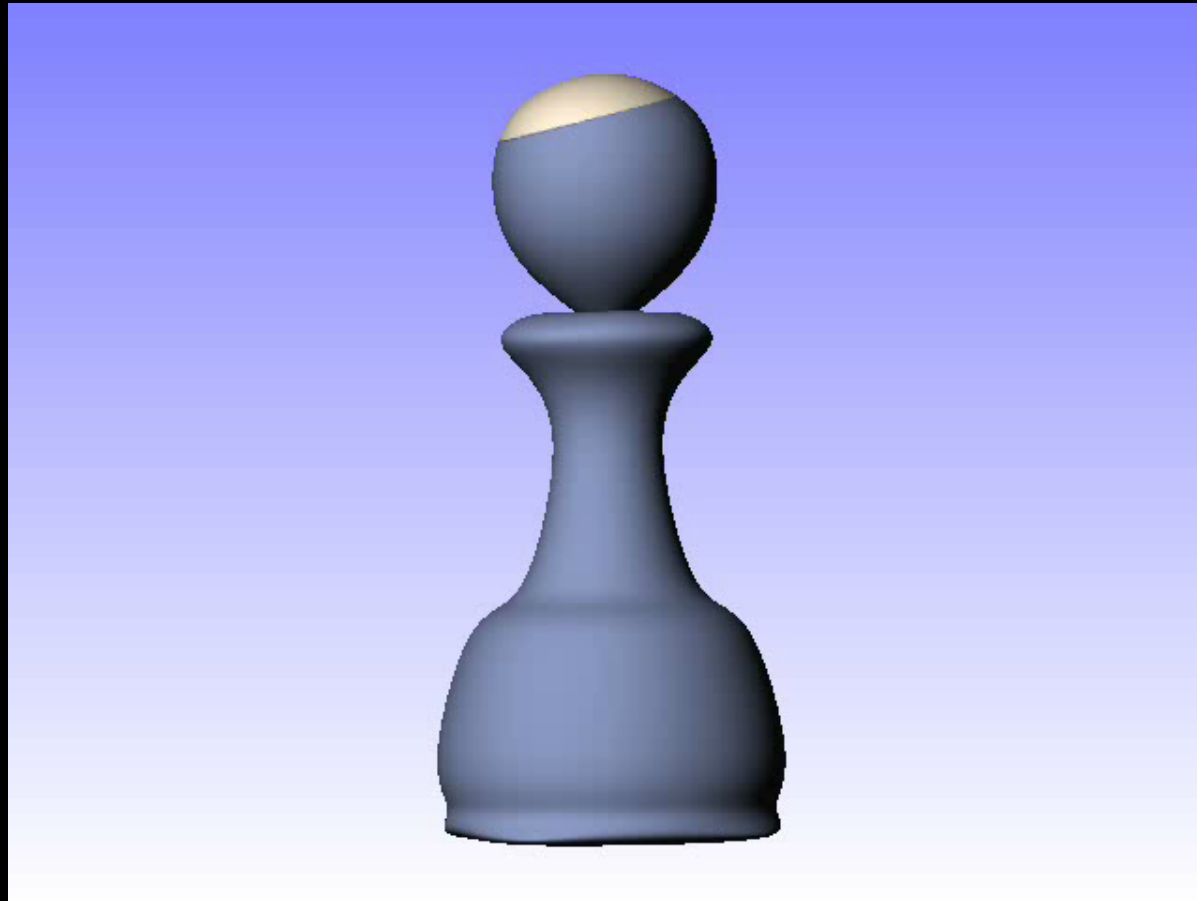
The iteration steps are [Stam 99, Fedkiw *et. al.* 01]:

- Compute intermediate velocity field \mathbf{u}^* ignoring the pressure term: i) adding forces; ii) perform semi-Lagrangian advection
- Projection (Decomposition):
 - i) define pressure as the solution to the Poisson equation,

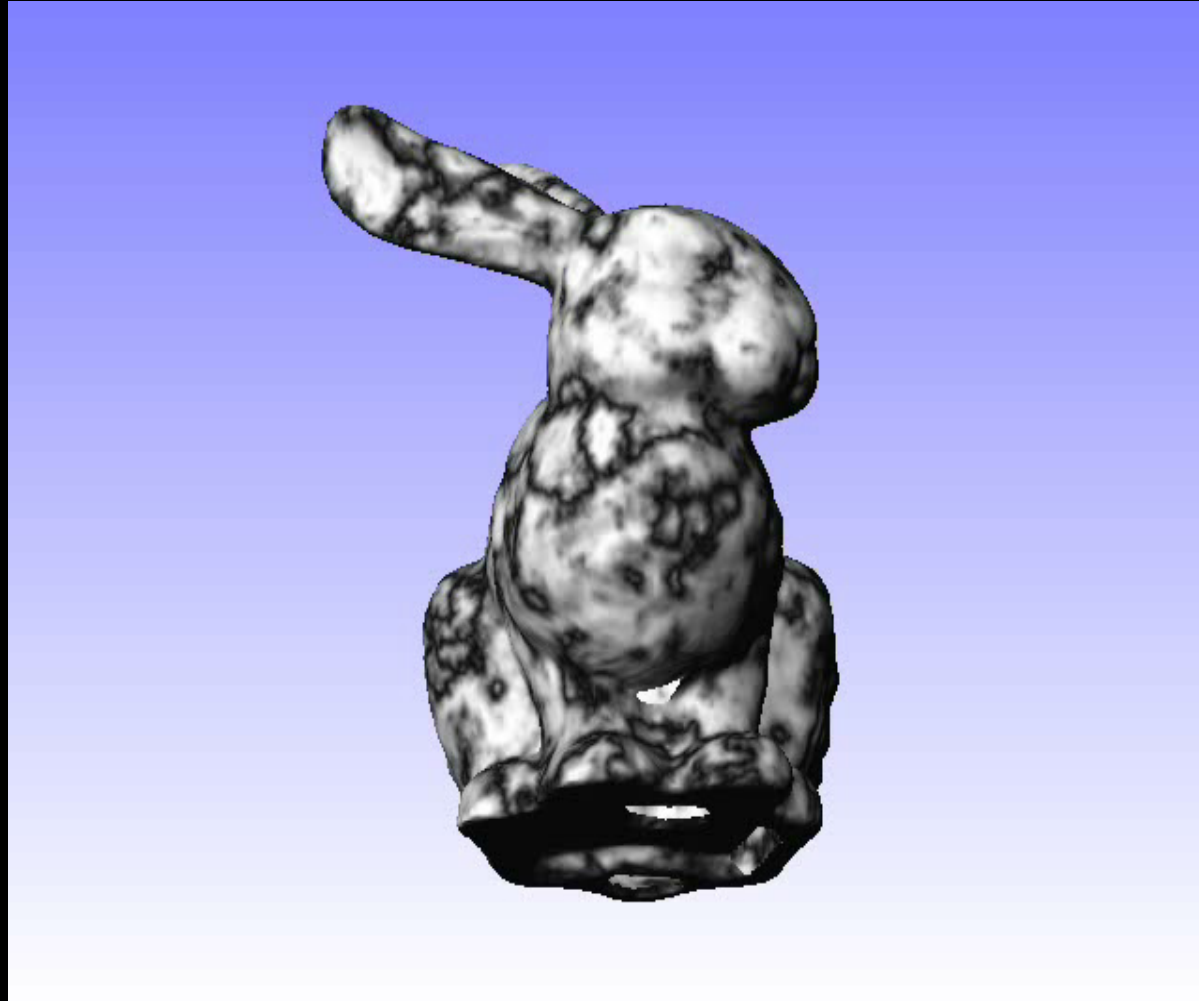
$$\nabla^2 p = \nabla \cdot \mathbf{u}^* / \Delta t$$

- ii) compute the velocity update via $\mathbf{u} = \mathbf{u}^* - \Delta t \nabla p$

Additional Surface Flow Results I



Additional Surface Flow Results II



Acknowledgments

- Collaborators: Lin Shi, Nathan Bell, Kun Zhou, Stephen Bond, Weiwei Xu, Qing Wu, Wei-Wen Feng, Baining Guo, Luke Olson, Harry Shum
- CMU motion capture database
- Stanford 3D model library
- NSF and UIUC research board

Thank You!