



Computer Graphics

# Mesh Processing

---

Teacher: Dr. Zhuo SU (苏卓)

E-mail: [suzhuo3@mail.sysu.edu.cn](mailto:suzhuo3@mail.sysu.edu.cn)

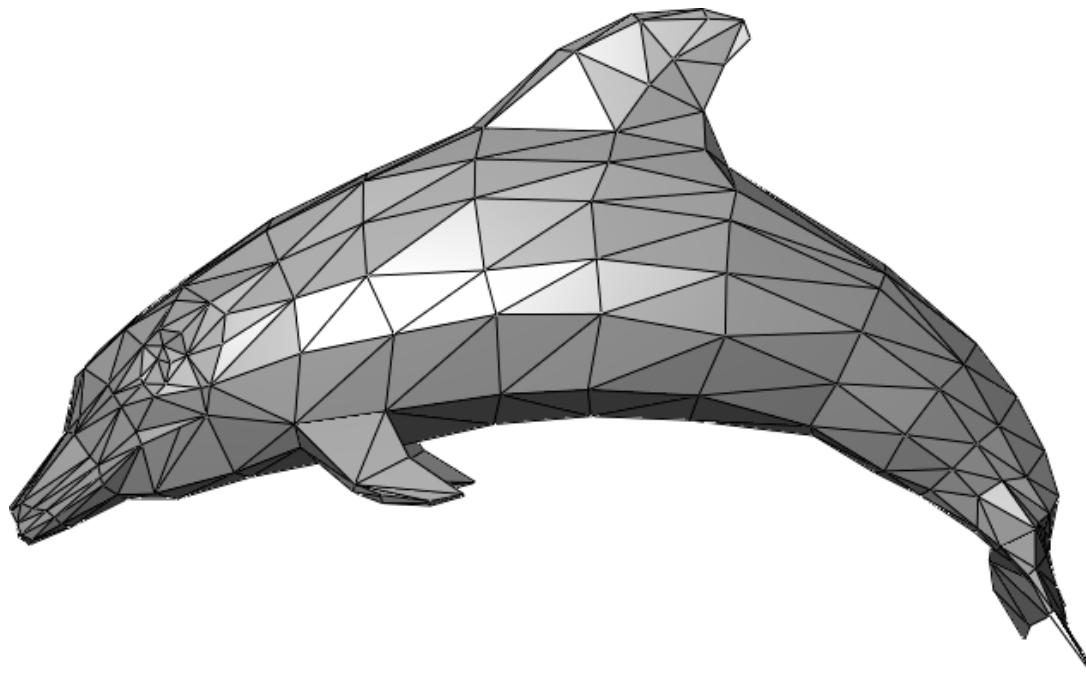
School of Data and Computer Science



# What is Polygon Mesh?

---

- A polygon mesh is a collection of vertices, edges, and faces that defines the shape of a polyhedral object in 3D computer graphics and solid modeling.



# Example – Polyhedral widgeon

---

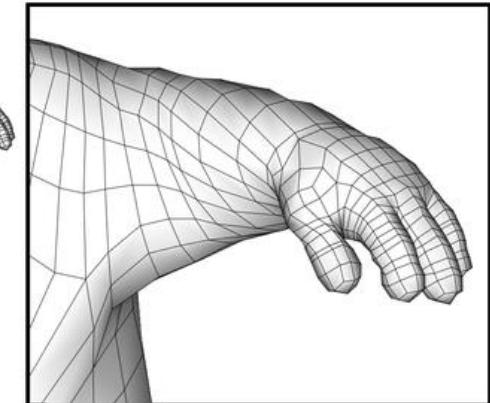
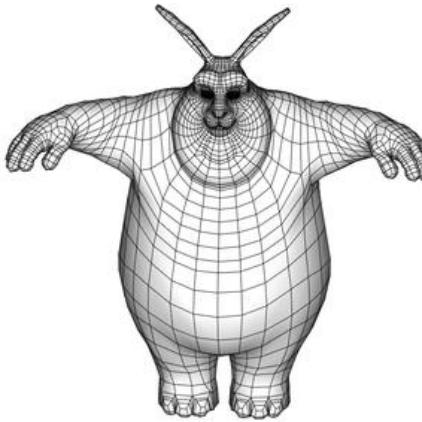
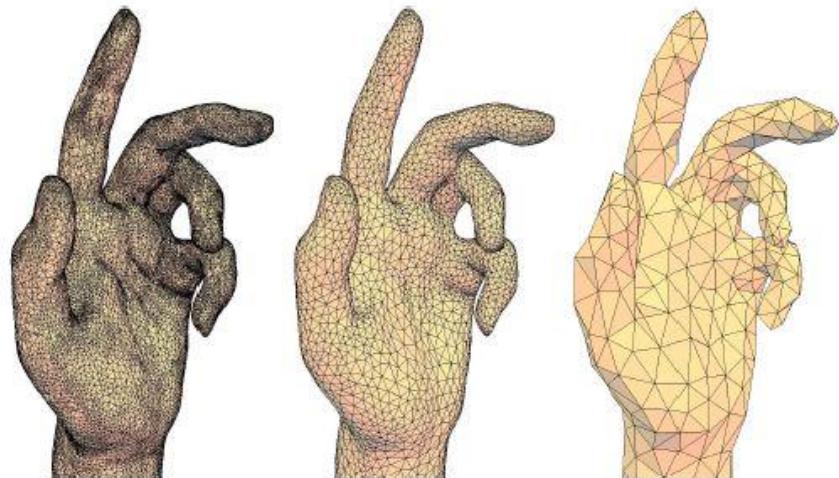
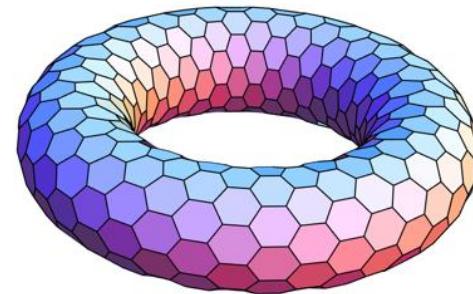


6656 faces (面), 3474 vertices (顶点)



# Categories of Polyhedron (多面体)

- Polyhedron are essentially linear approximation
  - Triangular meshes (三角网格)
  - Quadrilateral meshes (四边形网格)
  - Polygonal meshes (多边形网格)



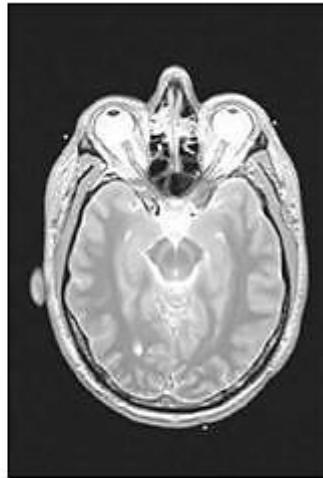
# Volumetric Scanning

---

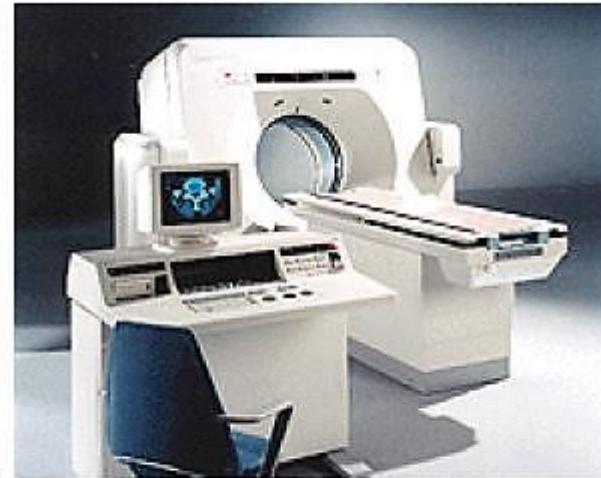
- Build voxel structure by scanning slices



CT

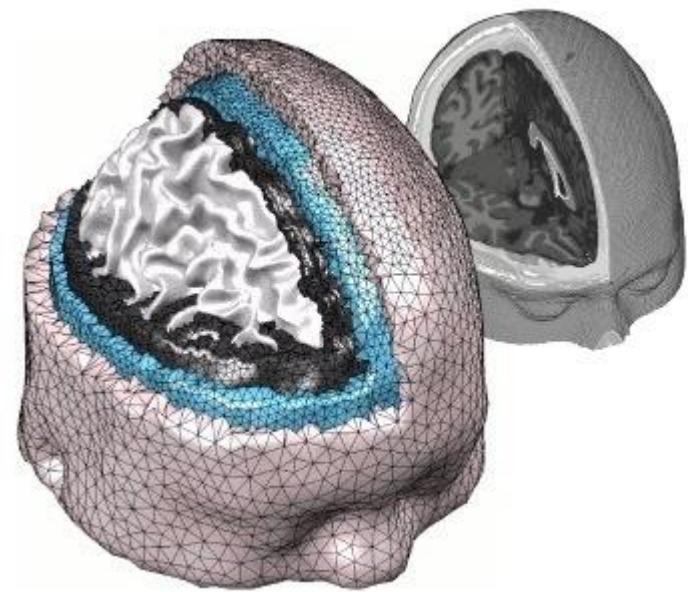
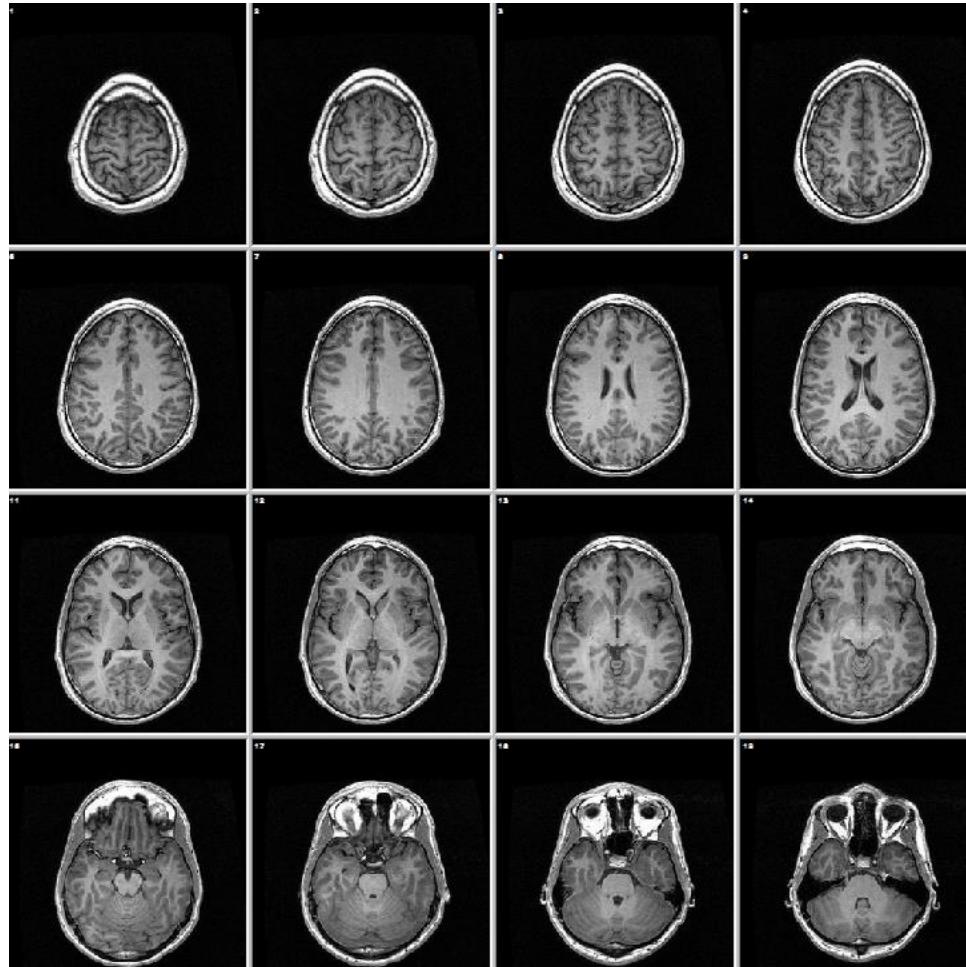


MRI



# Volumetric Scanning

- Build voxel structure by scanning slices



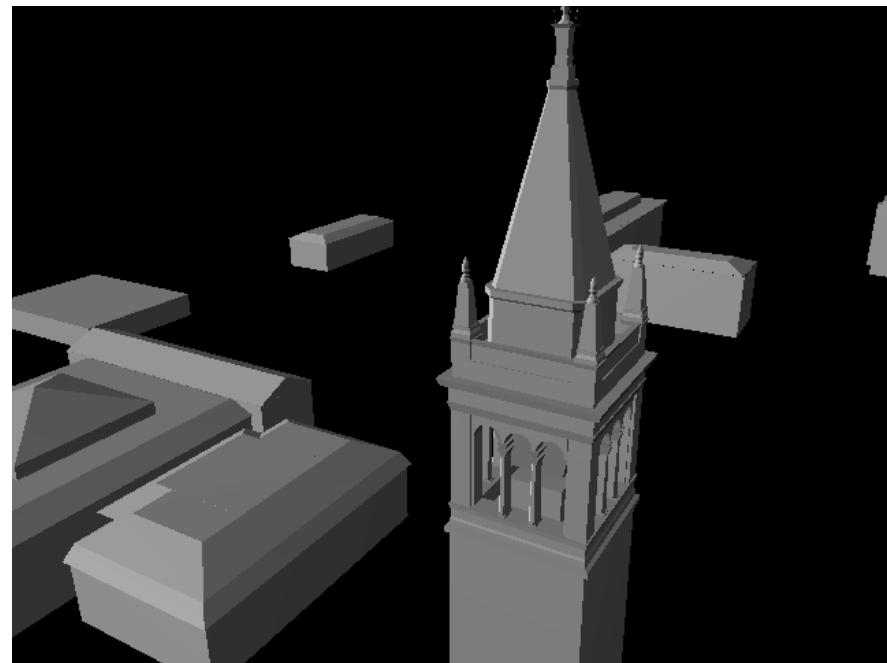
# Photogrammetry

---

- Reconstruction from photographs

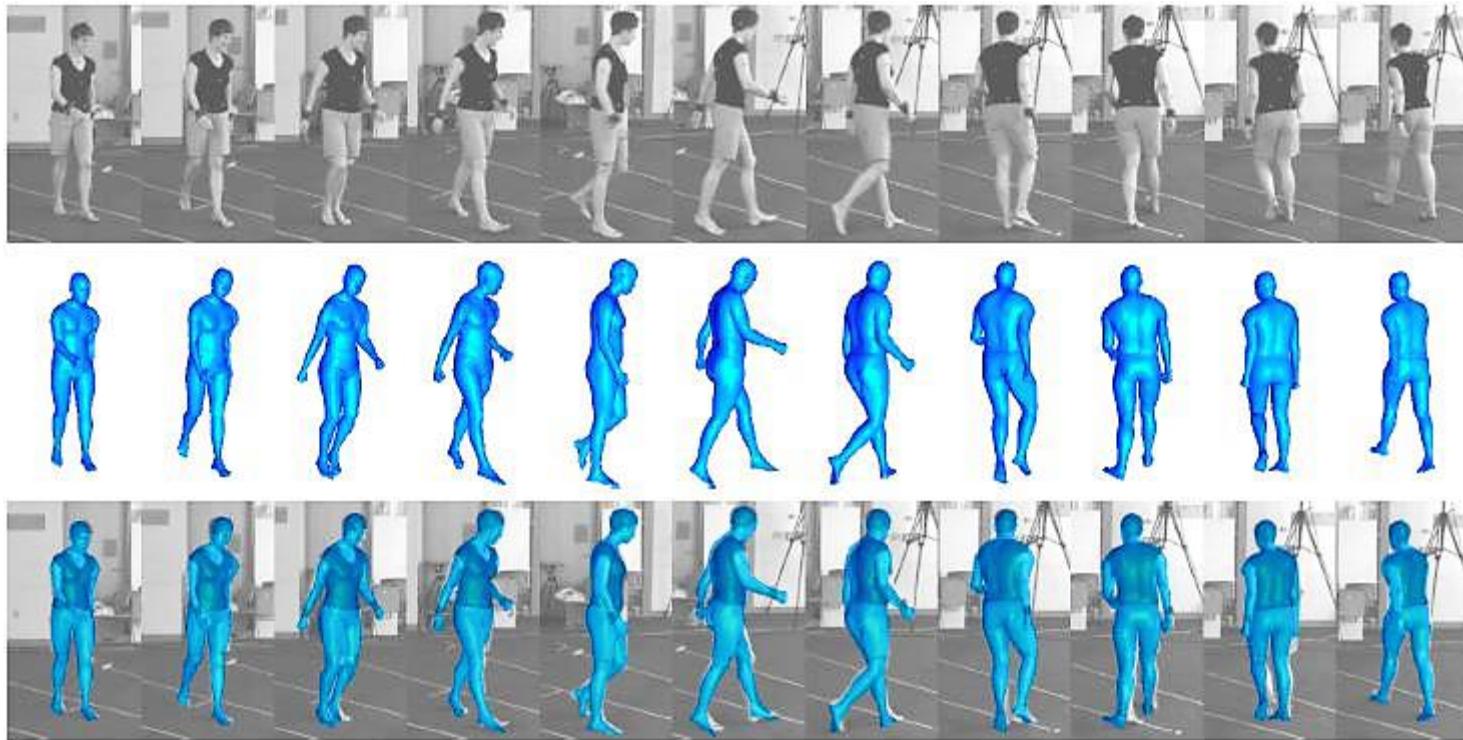


Tower Photographs



# Photogrammetry

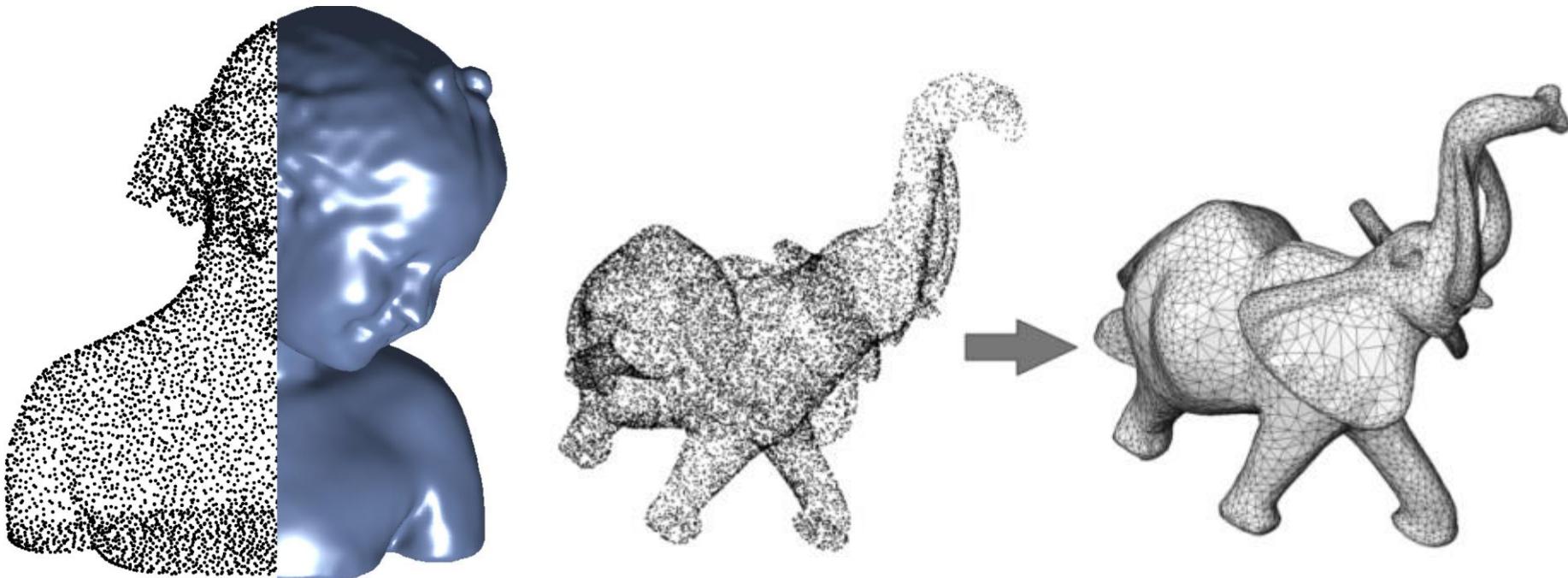
- Reconstruction from a series of photos (video)



# Range Scanning

---

- Reconstruction from point cloud



# Getting Meshes from Real Objects

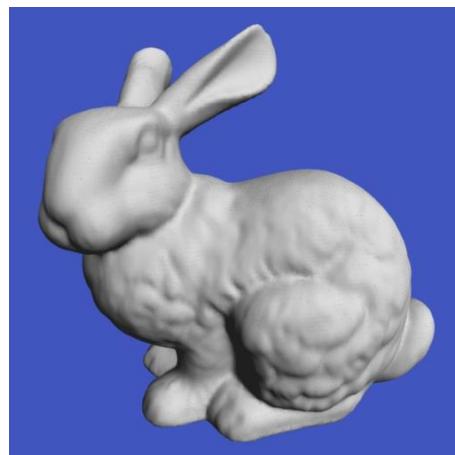
---

- Many models used in Graphics are obtained from real objects

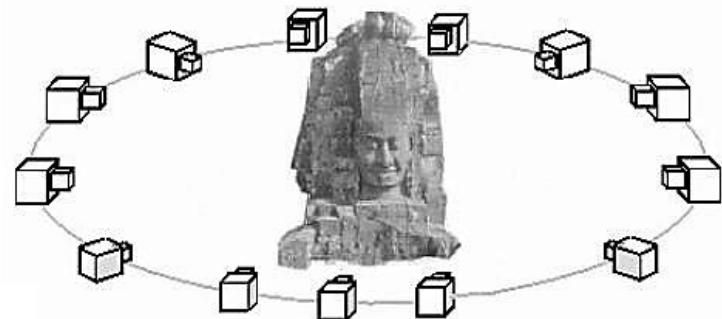


**Stanford dragon**

- Faces : 871414
- Vertices: 437645
- Compressed: 8.2 MB  
in PLY format



# Getting Meshes from Real Objects



# Range Scanning

---

- Accurate calibration is crucial
- Multiple scans required for complex objects
  - scan path planning
  - scan registration
- Scans are incomplete and noisy
  - model repair, hole filling
  - smoothing for noise removal



Set of raw scans



Reconstructed model



# General Used Mesh Files

---

- General used mesh files
  - Wavefront OBJ (\*.obj)
  - 3D Max (\*.max, \*.3ds)
  - VRML(\*.vrl)
  - Inventor (\*.iv)
  - PLY (\*.ply, \*.ply2)
  - User-defined(\*.m, \*.liu)
- Storage
  - Text – (Recommended)
  - Binary



# Wavefront OBJ File Format

---

- Vertices
  - Start with char ‘v’
  - (x,y,z) coordinates
- Faces
  - Start with char ‘f’
  - Indices of its vertices in the file
- Other properties
  - Normal, texture coordinates, material, etc.

```
v 1.0 0.0 0.0
v 0.0 1.0 0.0
v 0.0 -1.0 0.0
v 0.0 0.0 1.0
f 1 2 3
f 1 4 2
f 3 2 4
f 1 3 4
```



# Wavefront .obj file

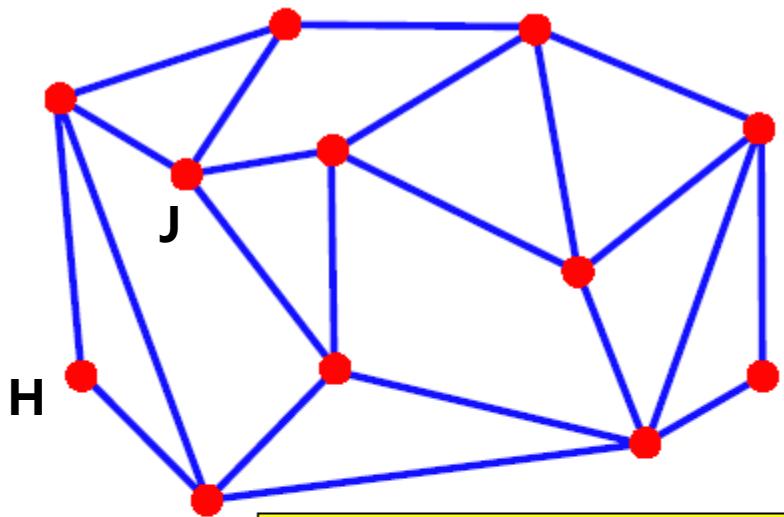
```
# List of Vertices, with (x,y,z[,w]) coordinates, w is optional and defaults to 1.0.  
v 0.123 0.234 0.345 1.0  
v ...  
...  
# Texture coordinates, in (u ,v [,w]) coordinates, these will vary between 0 and 1, w is optional and  
default to 0.  
vt 0.500 1 [0]  
vt ...  
...  
# Normals in (x,y,z) form; normals might not be unit.  
.vn 0.707 0.000 0.707  
vn ...  
...  
# Parameter space vertices in ( u [v] [,w] ) form; free form geometry statement ( see below )  
vp 0.310000 3.210000 2.100000  
vp ...  
...  
# Face Definitions (see below)  
f 1 2 3  
f 3/1 4/2 5/3  
f 6/4/1 3/5/3 7/6/5  
f ...  
...
```



# Meshes: Definitions & Terminologies



# Standard Graph Definition



**G = <V,E>**

**V = vertices =**

{A,B,C,D,E,F,G,H,I,J,K,L}

**E = edges =**

{(A,B),(B,C),(C,D),(D,E),(E,F),(F,G),  
(G,H),(H,A),(A,J),(A,G),(B,J),(K,F),  
(C,L),(C,I),(D,I),(D,F),(F,I),(G,K),  
(J,L),(J,K),(K,L),(L,I)}

**Vertex degree (valence)** = number of edges incident on vertex

$$\deg(J) = 4, \deg(H) = 2$$

**k-regular** graph = graph whose vertices all have degree  $k$

**Face:** cycle of vertices/edges which cannot be shortened

**F = faces =**

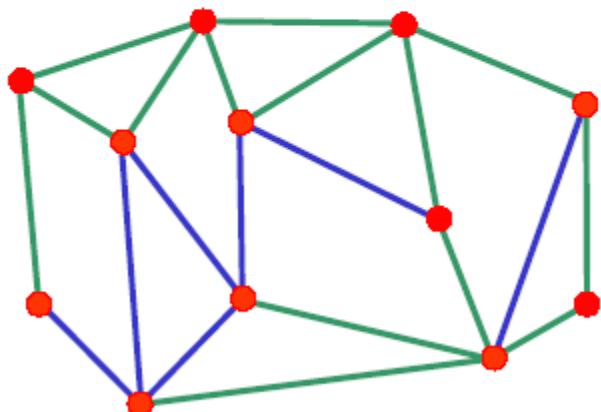
{(A,H,G),(A,J,K,G),(B,A,J),(B,C,L,J),(C,I,J),(C,D,I),  
(D,E,F),(D,I,F),(L,I,F,K),(L,J,K),(K,F,G)}



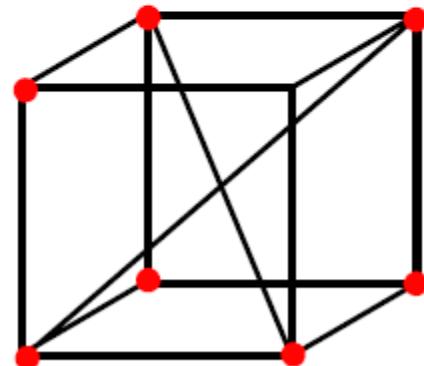
# Graph Embedding

---

Graph is *embedded* in  $\mathbb{R}^d$  if each vertex is assigned a position in  $\mathbb{R}^d$



Embedding in  $\mathbb{R}^2$



Embedding in  $\mathbb{R}^3$



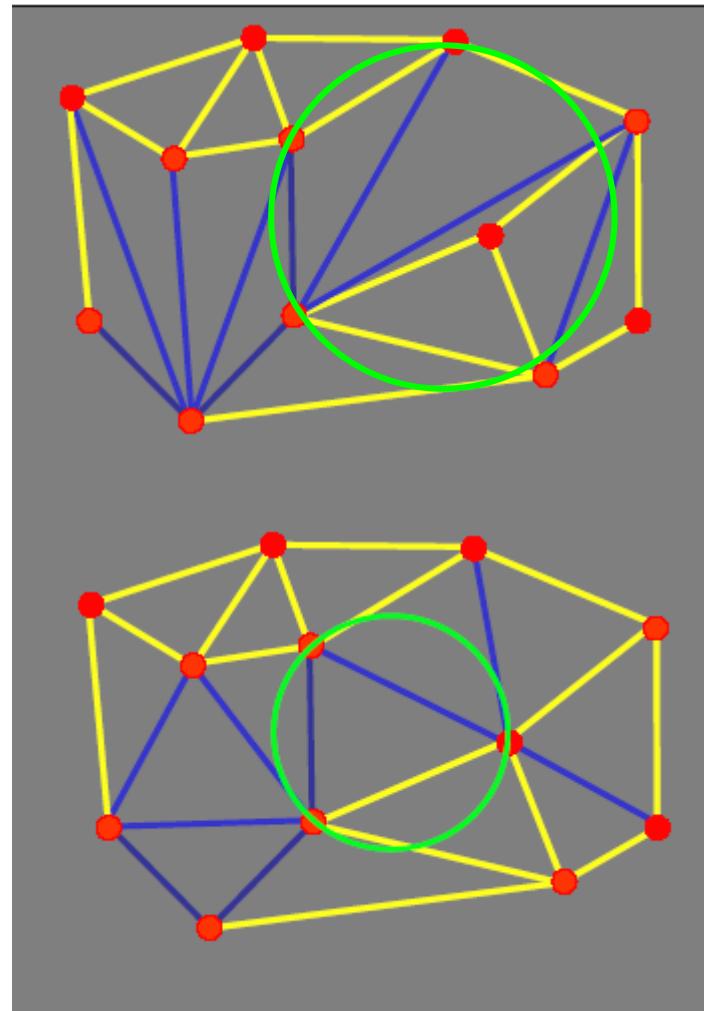
# Triangulation

**Triangulation:** straight line plane graph all of whose faces are triangles

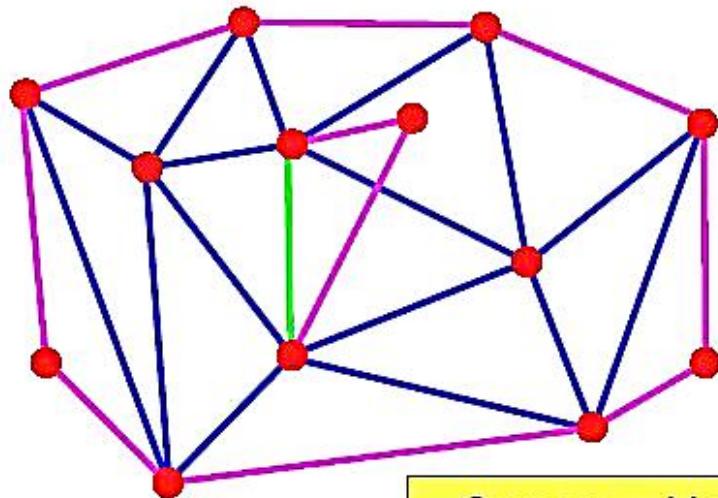
**Delaunay triangulation** of a set of points: unique set of triangles such that the circumcircle of any triangle does not contain any other point

Delaunay triangulation avoids long and skinny triangles

平面上的点集  $P$  的 德劳内三角化 是一种 三 角剖分  $DT(P)$  , 使得在  $P$  中没有点严格处于  $DT(P)$  中任意一个三角形 外接圆 的内部。



# Meshes



**Mesh**: straight-line graph embedded in  $\mathbb{R}^3$

**Boundary** edge: adjacent to exactly one face

**Regular** edge: adjacent to exactly two faces

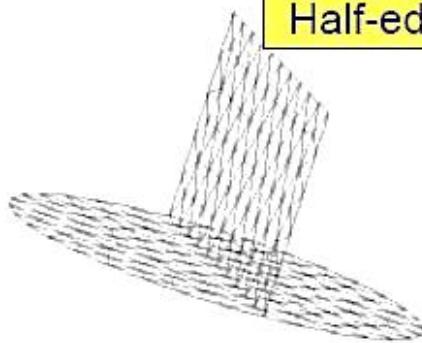
**Singular** edge: adjacent to more than two faces

**Closed** mesh: mesh with no boundary edges

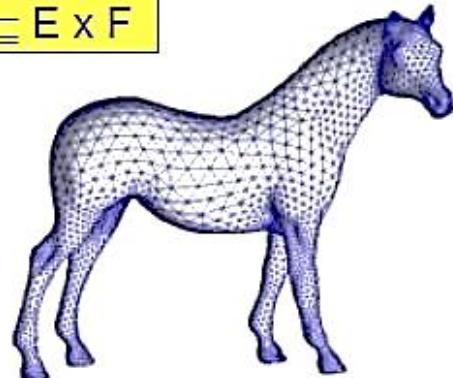
**Manifold** mesh: mesh with no singular edges

$$\text{Corners} \subseteq V \times F$$

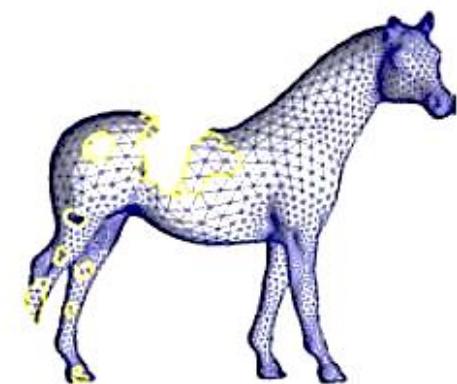
$$\text{Half-edges} \subseteq E \times F$$



Non-Manifold



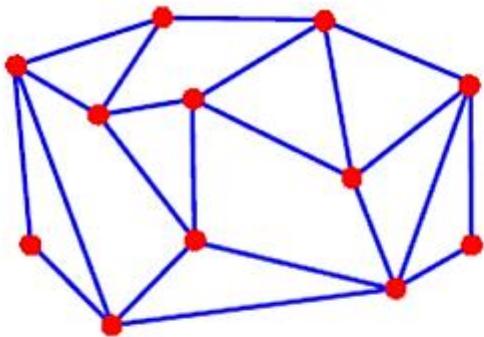
Closed Manifold



Open Manifold



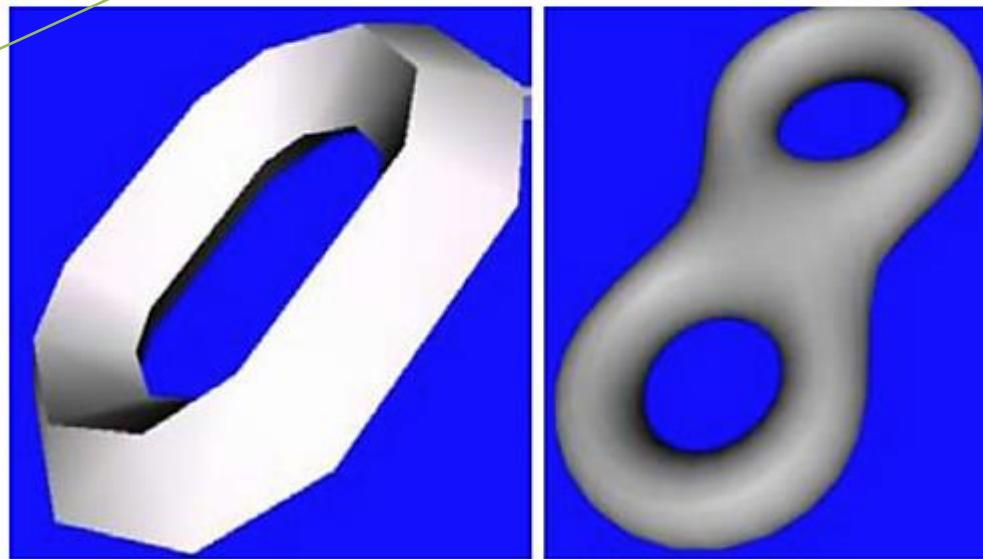
# Topology



**亏格** Genus of graph: *half of maximal number of closed paths that do not disconnect the graph (number of “holes”)*

$$\begin{aligned}\text{Genus(sphere)} &= 0 \\ \text{Genus(torus)} &= 1\end{aligned}$$

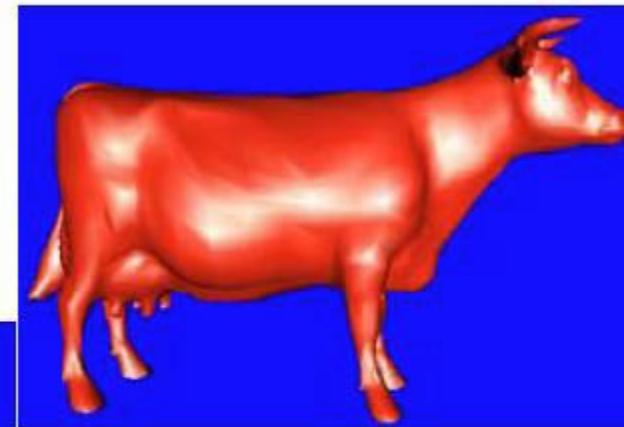
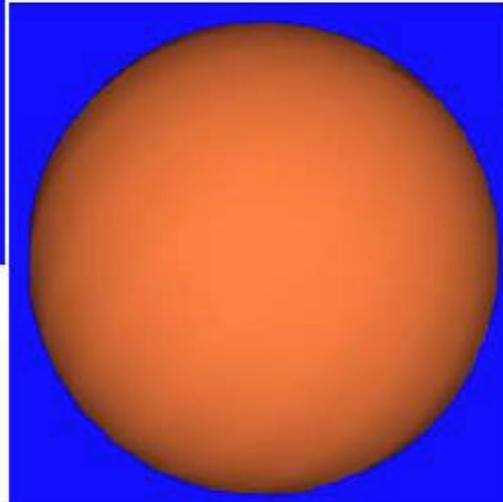
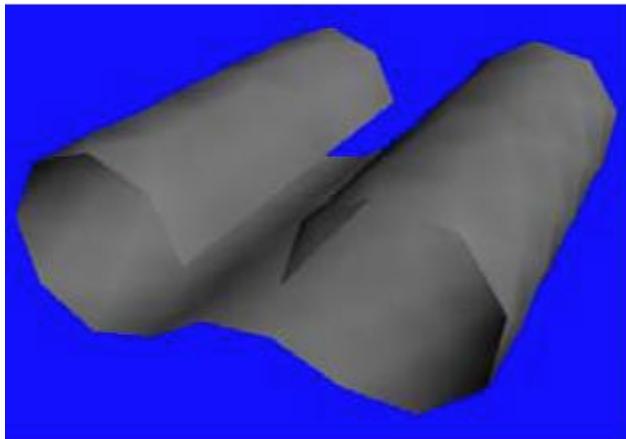
连通，可定向曲面的亏格是一个整数，代表沿闭简单曲线切开但不切断曲面的最大曲线条数。这和柄的个数是相同的



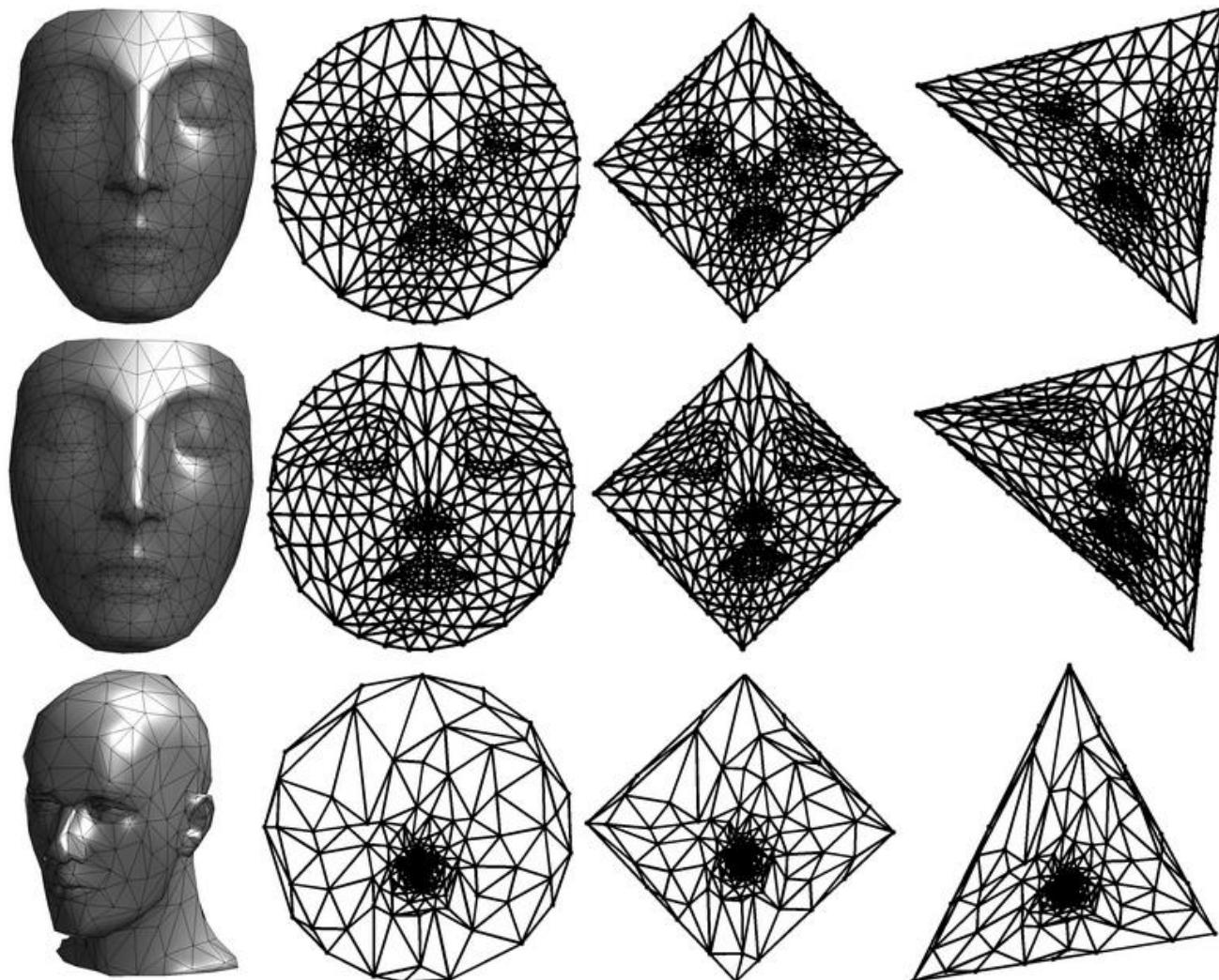
# Developability (可展性)

Mesh is **developable** if it may be embedded in  $\mathbb{R}^2$  without distortion

失真



# Developability (可展性)



# Mesh Data Structure

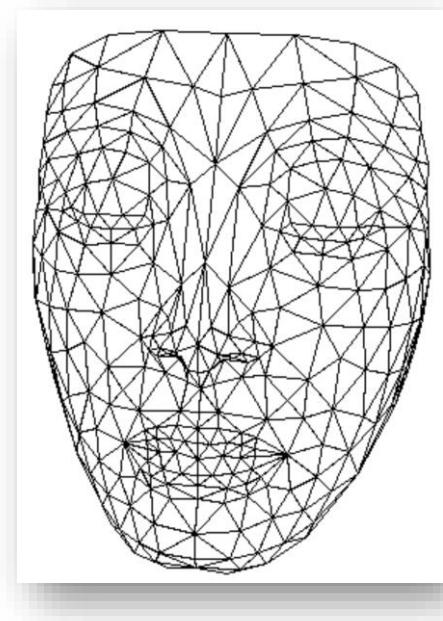
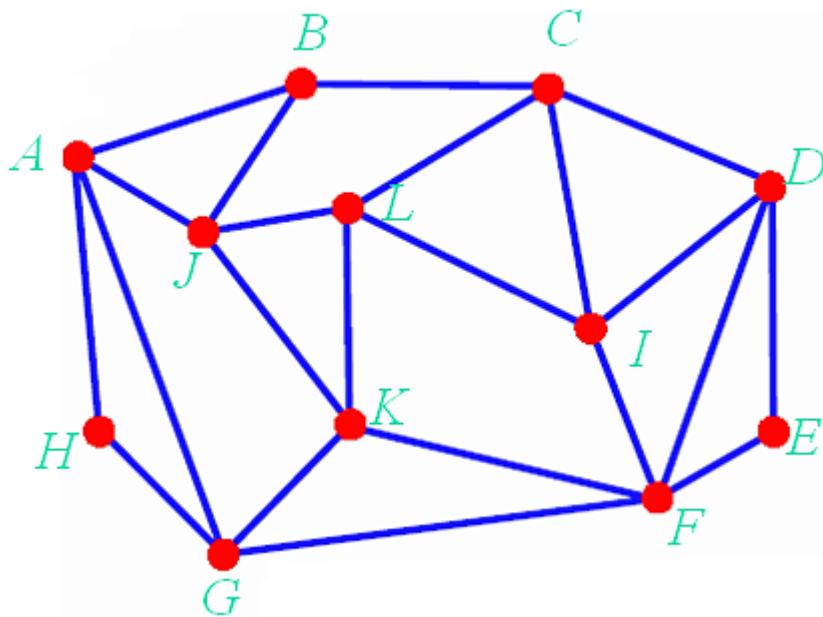
---

- How to store geometry and connectivity?
- Geometry queries
  - What are the vertices of face #k?
  - Are vertices #i and #j adjacent?
  - Which faces are adjacent face #k?
- Geometry operations
  - Remove/add a vertex/face
  - Mesh simplification
  - Vertex split, edge collapse



# Define a mesh

- Geometry
  - Vertex coordinates
- Connectivity
  - How do vertices connected?



- List of Edge
- Vertex-Edge
- Vertex-Face
- Combined



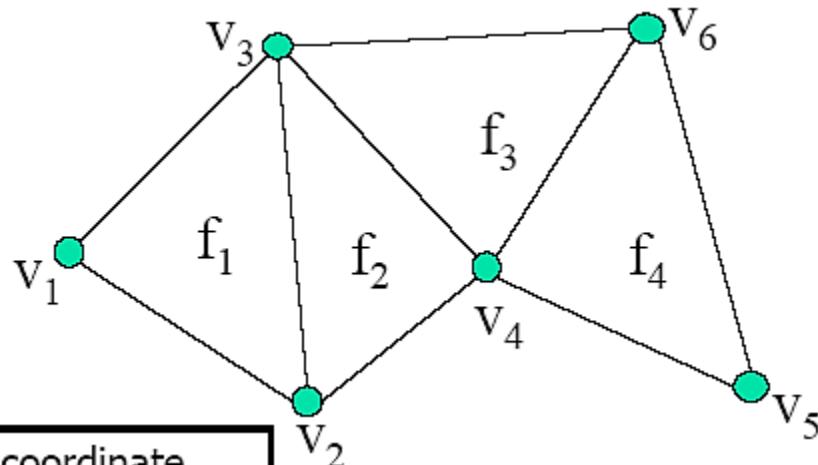
# List of Faces

---

- List of vertices
  - Position coordinates
- List of faces
  - Triplets of pointers to face vertices ( $c_1, c_2, c_3$ )
- Queries:
  - What are the vertices of face #3?
    - Answered in  $O(1)$  - checking third triplet
  - Are vertices  $i$  and  $j$  adjacent?
    - A pass over all faces is necessary – NOT GOOD



# List of Faces – Example



vertex	coordinate
v <sub>1</sub>	(x <sub>1</sub> ,y <sub>1</sub> ,z <sub>1</sub> )
v <sub>2</sub>	(x <sub>2</sub> ,y <sub>2</sub> ,z <sub>2</sub> )
v <sub>3</sub>	(x <sub>3</sub> ,y <sub>3</sub> ,z <sub>3</sub> )
v <sub>4</sub>	(x <sub>4</sub> ,y <sub>4</sub> ,z <sub>4</sub> )
v <sub>5</sub>	(x <sub>5</sub> ,y <sub>5</sub> ,z <sub>5</sub> )
v <sub>6</sub>	(x <sub>6</sub> ,y <sub>6</sub> ,z <sub>6</sub> )

face	vertices (ccw)
f <sub>1</sub>	(v <sub>1</sub> , v <sub>2</sub> , v <sub>3</sub> )
f <sub>2</sub>	(v <sub>2</sub> , v <sub>4</sub> , v <sub>3</sub> )
f <sub>3</sub>	(v <sub>3</sub> , v <sub>4</sub> , v <sub>6</sub> )
f <sub>4</sub>	(v <sub>4</sub> , v <sub>5</sub> , v <sub>6</sub> )



# List of Faces – Analysis

---

- Pros:
  - Convenient and efficient (memory wise)
  - Can represent non-manifold meshes
- Cons:
  - Too simple - not enough information on relations between vertices & faces



# Adjacency Matrix – Definition

---

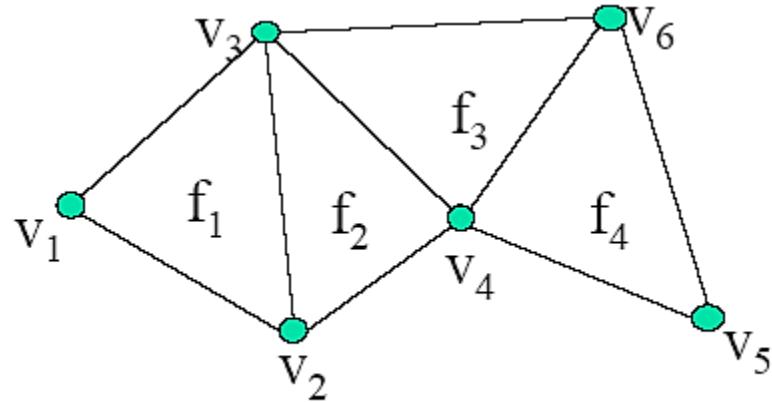
- View mesh as connected graph
- Given  $n$  vertices build  $n \times n$  matrix of adjacency information
  - Entry  $(i,j)$  is TRUE value if vertices  $i$  and  $j$  are adjacent
- Geometric info
  - list of vertex coordinates
- Add faces
  - list of triplets of vertex indices  $(v_1, v_2, v_3)$



# Adjacency Matrix – Example

vertex	coordinate
$v_1$	$(x_1, y_1, z_1)$
$v_2$	$(x_2, y_2, z_2)$
$v_3$	$(x_3, y_3, z_3)$
$v_4$	$(x_4, y_4, z_4)$
$v_5$	$(x_5, y_5, z_5)$
$v_6$	$(x_6, y_6, z_6)$

face	vertices (ccw)
$f_1$	$(v_1, v_2, v_3)$
$f_2$	$(v_2, v_4, v_3)$
$f_3$	$(v_3, v_4, v_6)$
$f_4$	$(v_4, v_5, v_6)$



	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$
$v_1$		1	1			
$v_2$	1			1	1	
$v_3$	1	1			1	1
$v_4$		1	1		1	1
$v_5$				1		1
$v_6$				1	1	1



# Adjacency Matrix – Queries

---

- What are the vertices of face #3?
  - $O(1)$  – checking third triplet of faces
- Are vertices i and j adjacent?
  - $O(1)$  - checking adjacency matrix at location  $(i,j)$ .
- Which faces are adjacent to vertex j?
  - Full pass on all faces is necessary



# Adjacency Matrix – Analysis

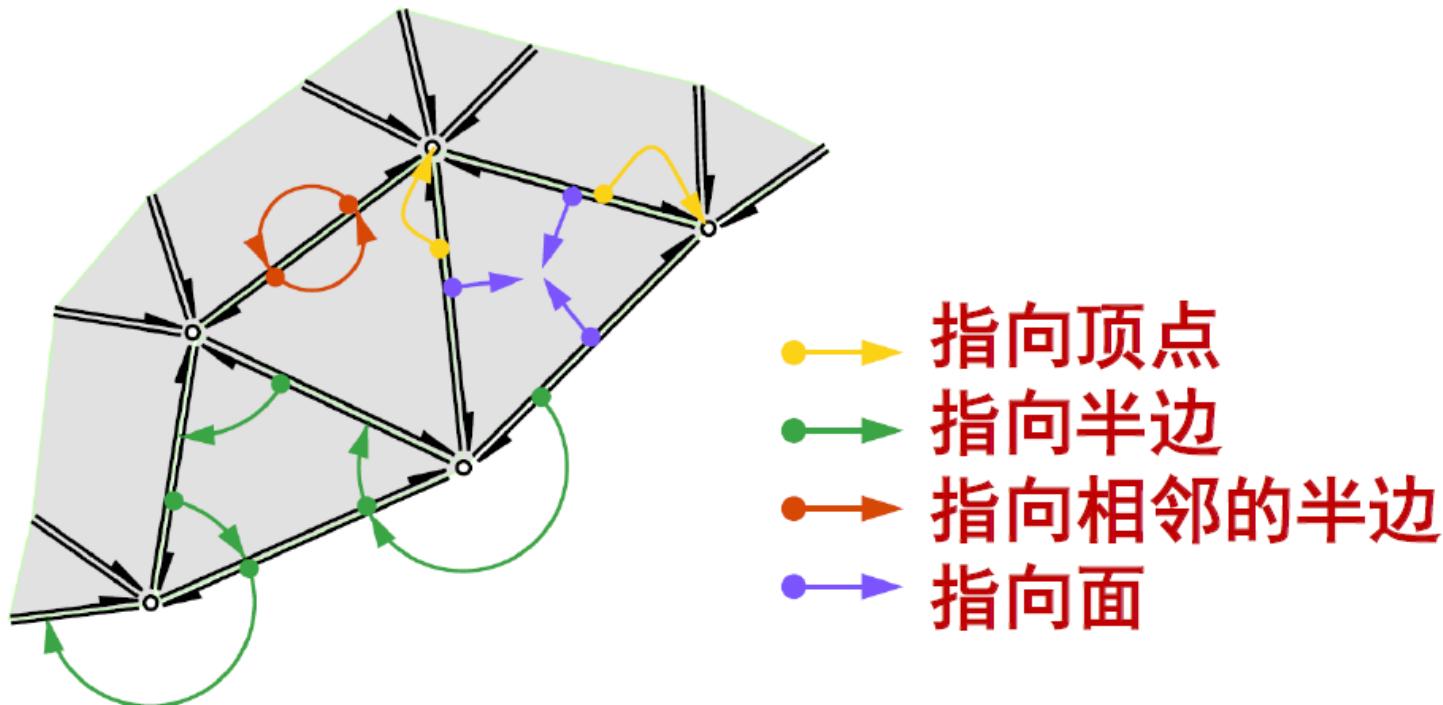
---

- Pros:
  - Information on vertices adjacency
  - Stores non-manifold meshes
- Cons:
  - Connects faces to their vertices, BUT NO connection between vertex and its face



# Half-Edge Structure

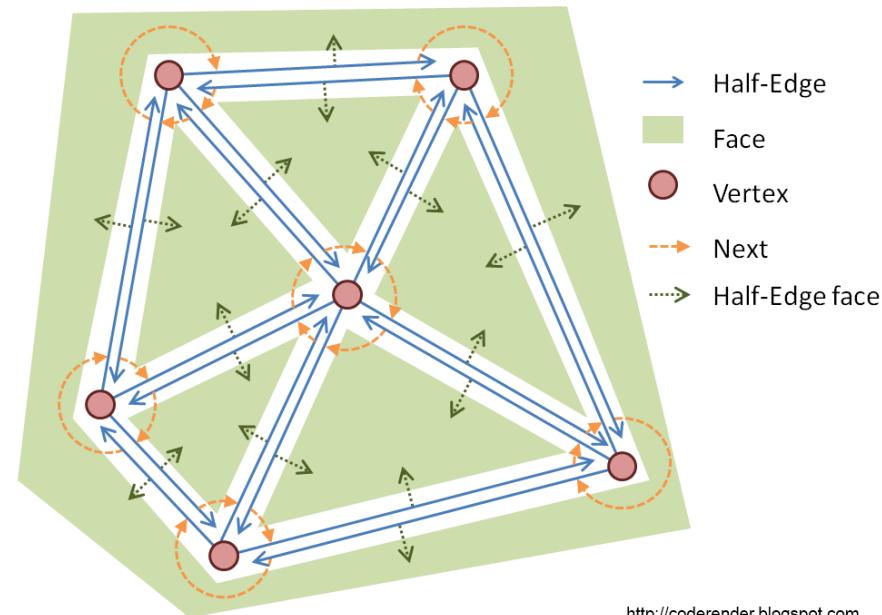
- Orientable 2D manifolds and its sub set: special polygonal meshes (适用于有向的二维流形)



# Half-Edge Structure

- Half-edge (each edge corresponds to two half-edges)
  - Pointer to the first vertices
  - To adjacent face
  - To next half-edge (逆时针方向)
  - To the other half-edge of the same edge
  - To previous half-edge (opt.)

```
struct HE_edge {  
    HE_vert* vert; // vertex at the start of the half-edge  
    HE_face* face; // face the half-edge borders  
    HE_edge* pair; // oppositely oriented adjacent half-edge  
    HE_edge* next; // next half-edge around the face  
    HE_edge* prev; // prev half-edge around the face  
};
```

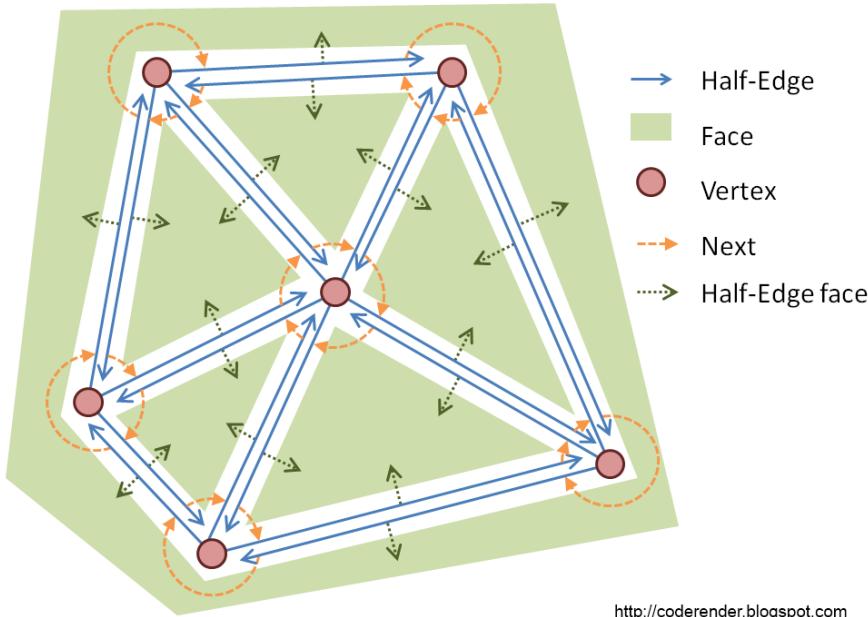


<http://coderender.blogspot.com>



# Half-Edge Structure

- Face : we only need a pointer to one of its half-edge



```
struct HE_face {  
    HE_edge* edge; // one of the half-edges bordering the face  
};
```



# Half-Edge Structure

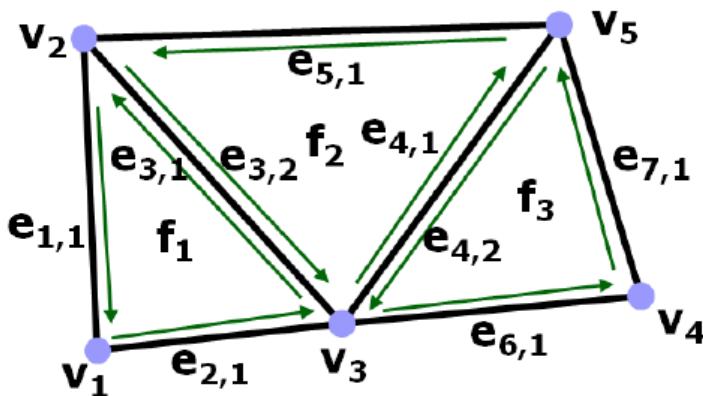
---

- Vertices
  - 3D coordinates
  - Pointer to the half-edge starting from it

```
struct HE_vert {  
    float x;  
    float y;  
    float z;  
    HE_edge* edge; // one of the half-edges  
                    // emanating from the vertex  
};
```



# Example: half-edge structure

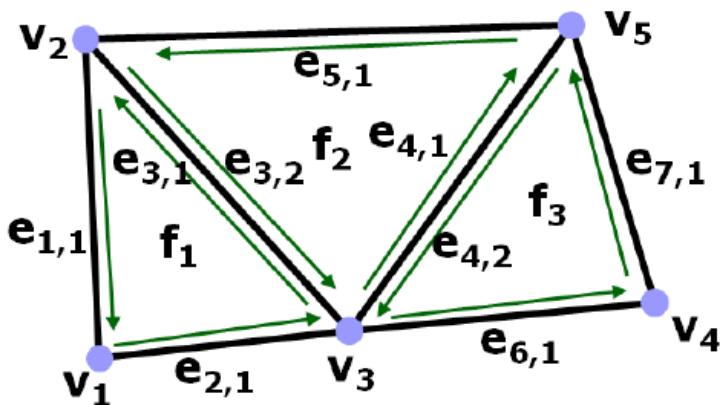


顶点	坐标	以此为起点的半边
$v_1$	$(x_1, y_1, z_1)$	$e_{2,1}$
$v_2$	$(x_2, y_2, z_2)$	$e_{1,1}$
$v_3$	$(x_3, y_3, z_3)$	$e_{4,1}$
$v_4$	$(x_4, y_4, z_4)$	$e_{7,1}$
$v_5$	$(x_5, y_5, z_5)$	$e_{5,1}$

面	半边
$f_1$	$e_{1,1}$
$f_2$	$e_{3,2}$
$f_3$	$e_{4,2}$



# Example (continued)

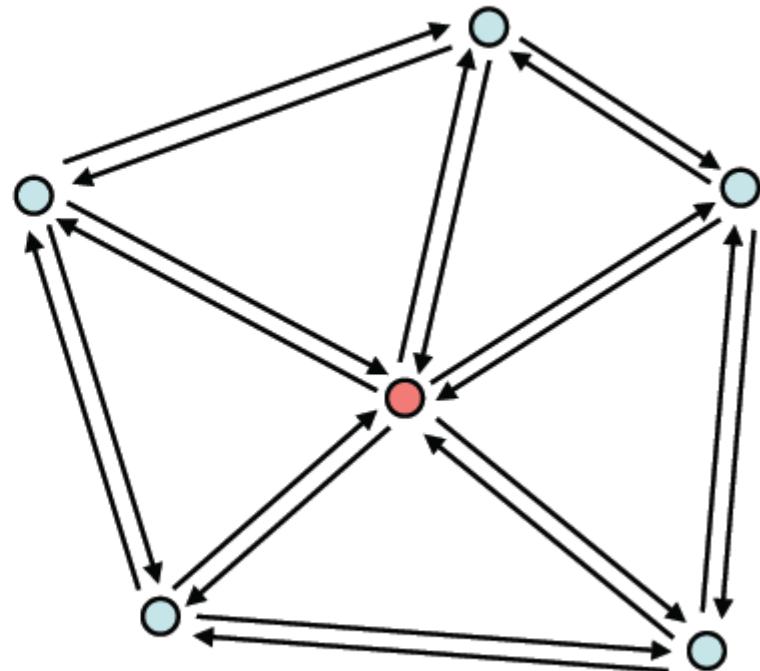


半边	起点	相邻半边	面	下条半边	前条半边
$e_{3,1}$	$v_3$	$e_{3,2}$	$f_1$	$e_{1,1}$	$e_{2,1}$
$e_{3,2}$	$v_2$	$e_{3,1}$	$f_2$	$e_{4,1}$	$e_{5,1}$
$e_{4,1}$	$v_3$	$e_{4,2}$	$f_2$	$e_{5,1}$	$e_{3,2}$
$e_{4,2}$	$v_5$	$e_{4,1}$	$f_3$	$e_{6,1}$	$e_{7,1}$

# One-Ring Traversal

---

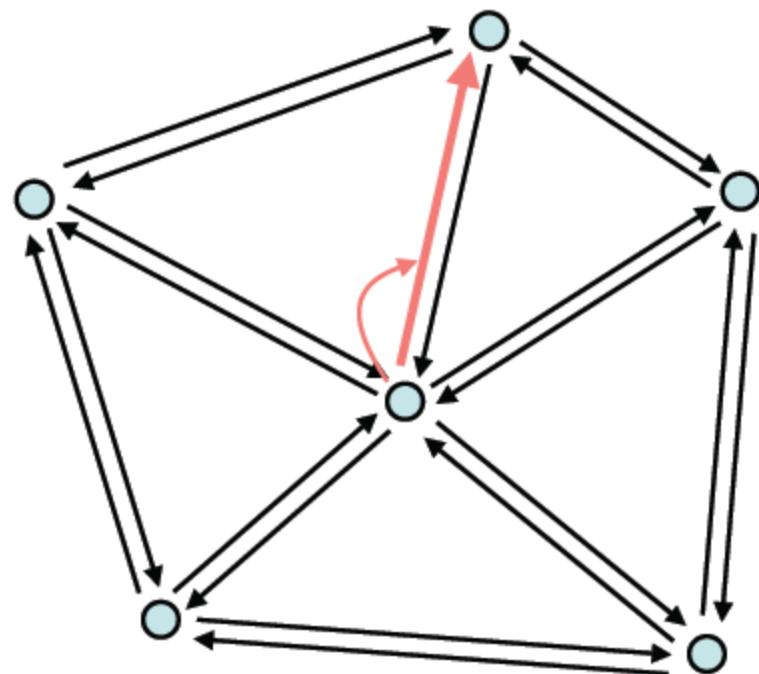
## 1. Start at vertex



# One-Ring Traversal

---

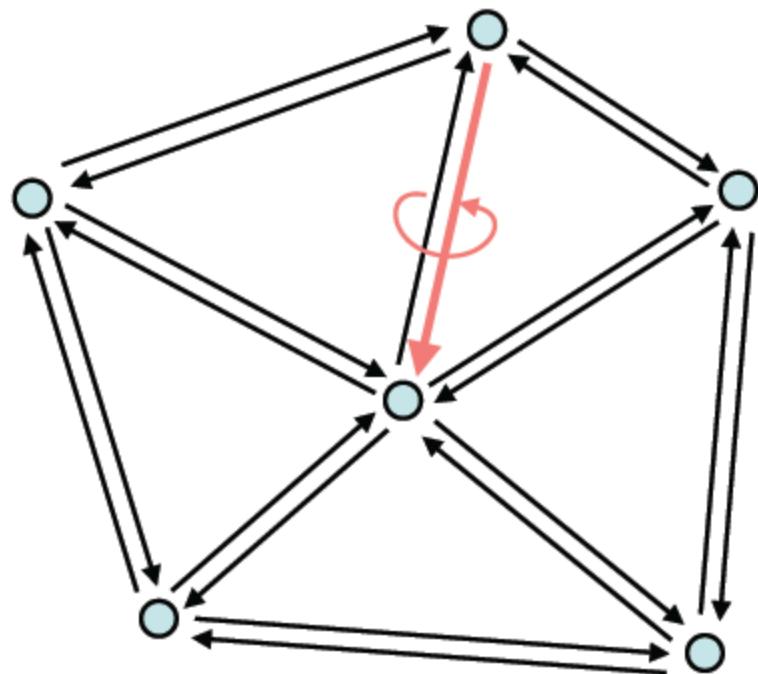
1. Start at vertex
2. Outgoing halfedge



# One-Ring Traversal

---

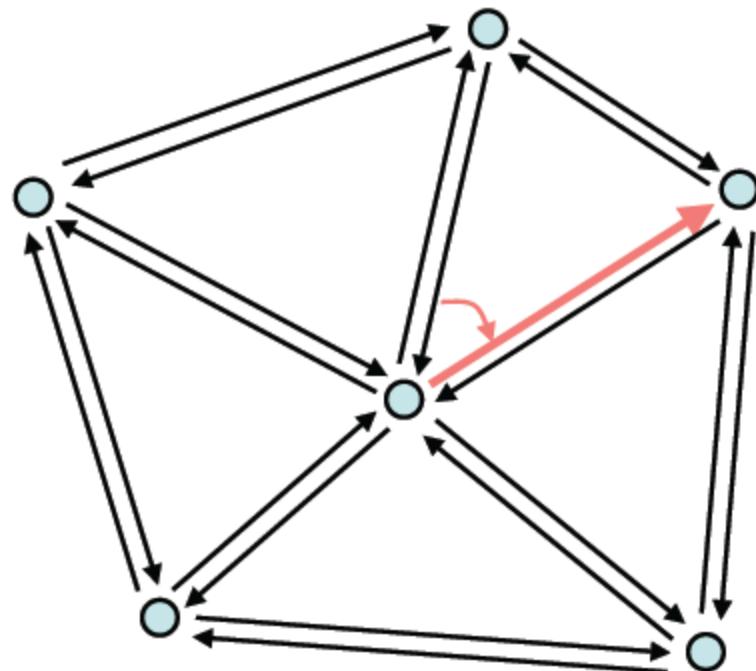
1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



# One-Ring Traversal

---

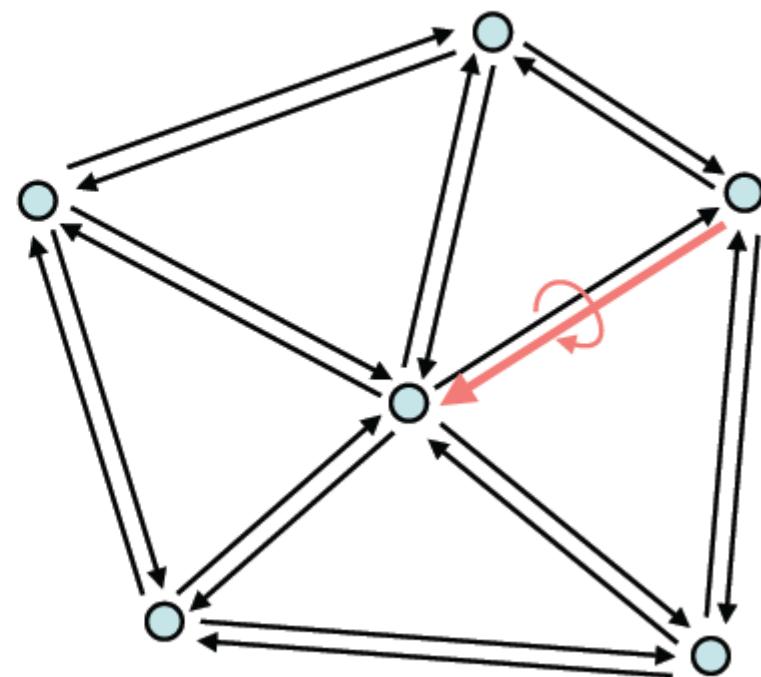
1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



# One-Ring Traversal

---

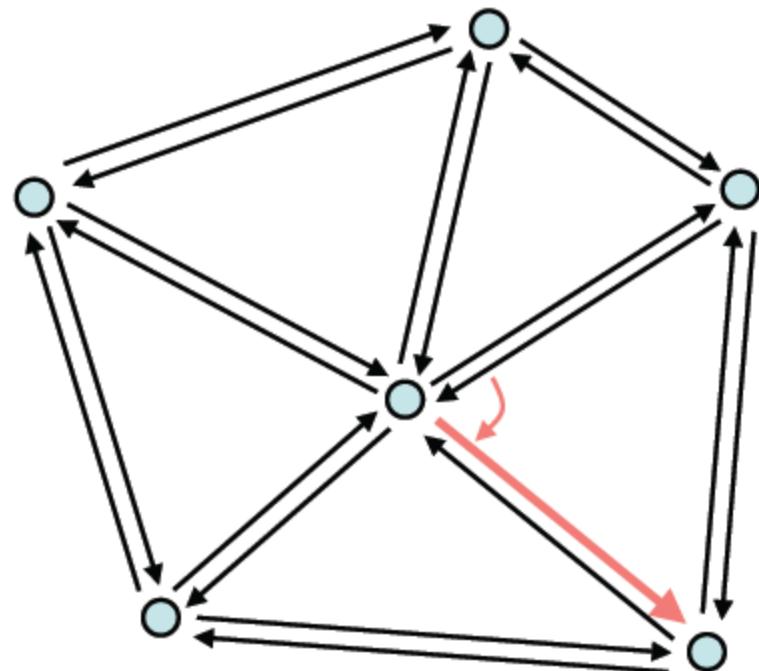
1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite



# One-Ring Traversal

---

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...



# Traversal operations

---

Vertices adjacent to a vertex v, mesh without boundary

```
he = v->halfedge;
do {
    he = he->sym->next;
    ... // perform operations with
         // he->vertex
} while (he != v->halfedge)
```

No “if” statements.



# Basic operations

---

- Mark mesh boundary (标记边界点)
- Create edge adjacency (创建邻接边)
- Add vertex (增加顶点)
- Add edge (增加边)
- Add polygonal face (增加面)
- Delete polygonal face (删除面)
- Delete edge (删除边)
- Delete vertex (删除顶点)



# Discussion

---

- Advantage and disadvantage(优缺点) :
  - Adv. : Query time  $O(1)$  , operation time  $O(1)$
  - Dis. : redundancy & only applicable to 2D manifolds
- For more information refer to
  - CGAL :
    - the Computational Geometry Algorithms Library , <http://www.cgal.org/>
    - Free for non-commercial use
  - OpenMesh : <http://www.openmesh.org/>
    - Mesh processing
    - Free, LGPL licence
  - Meshlab: <http://meshlab.sourceforge.net/>



# Advantage and disadvantage in polygon representation

---

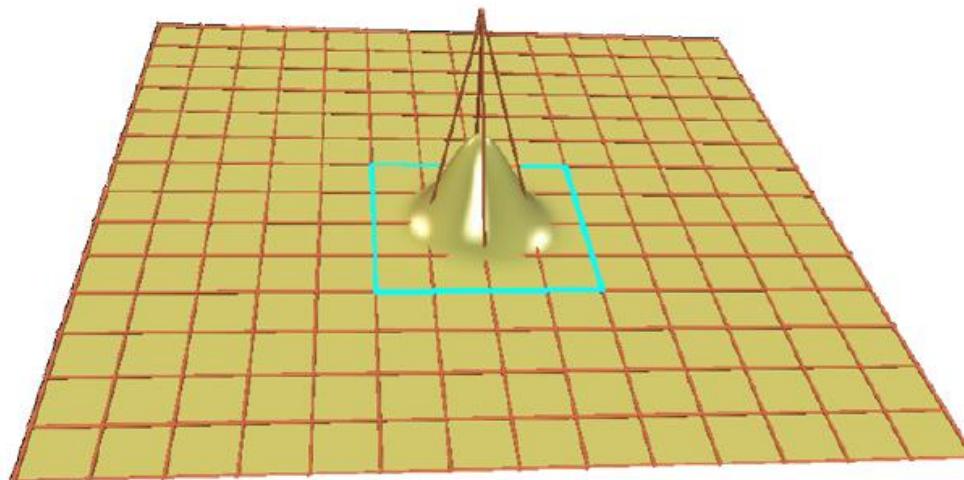
- Advantage
  - Simplicity - ease of description
  - Based data for rendering software/hardware
  - Input to most simulation/analysis tools
  - Output of most acquisition tools
  - laser scanner, CT, MRI, etc...
- Disadvantage
  - Approximation, it is hard to satisfy real time interaction
  - It is hard to edit mesh with traditional method.
  - Without analytical form, geometric attribute is hard to compute
  - When expressed object with complex topology and rich details, modeling/editing/rendering/storing will have more burden.



# Spline Surfaces (样条曲面)

- Tensor product surfaces (“curves of curves”)
  - Rectangular grid of control points

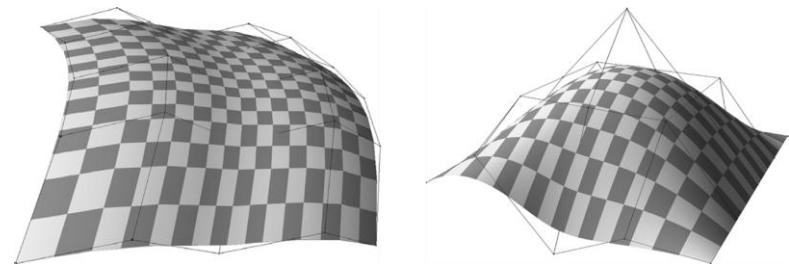
$$\mathbf{p}(u, v) = \sum_{i=0}^k \sum_{j=0}^l \mathbf{p}_{ij} N_i^n(u) N_j^n(v)$$



# Spline Surfaces (样条曲面)

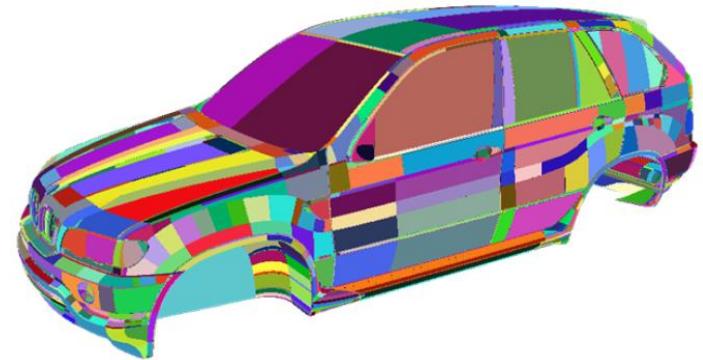
- Tensor product surfaces (“curves of curves”)

- Rectangular grid of control points
- Rectangular surface patch



- Problems:

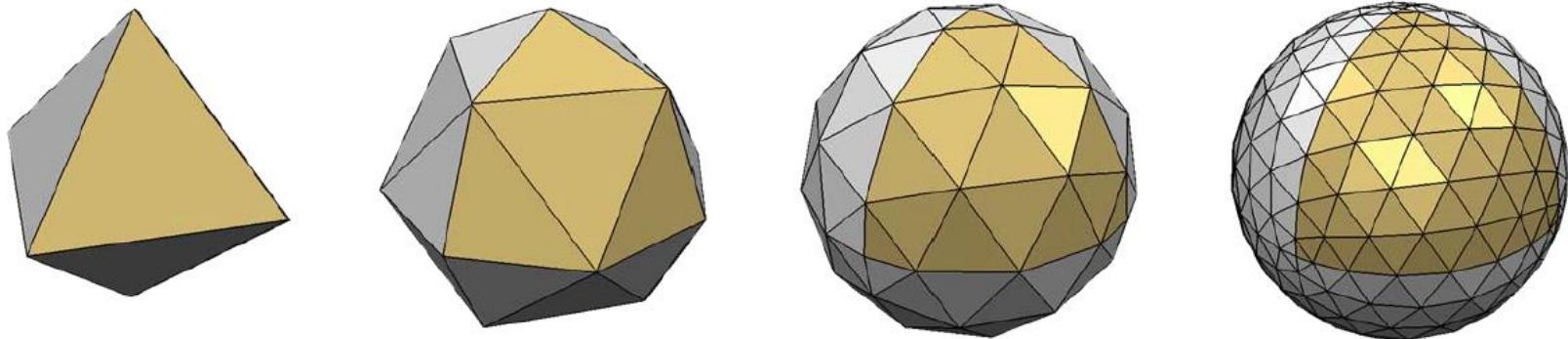
- Many patches for complex models
- Smoothness across patch boundaries
- Trimming for non-rectangular patches



# Subdivision Surfaces (细分曲面)

---

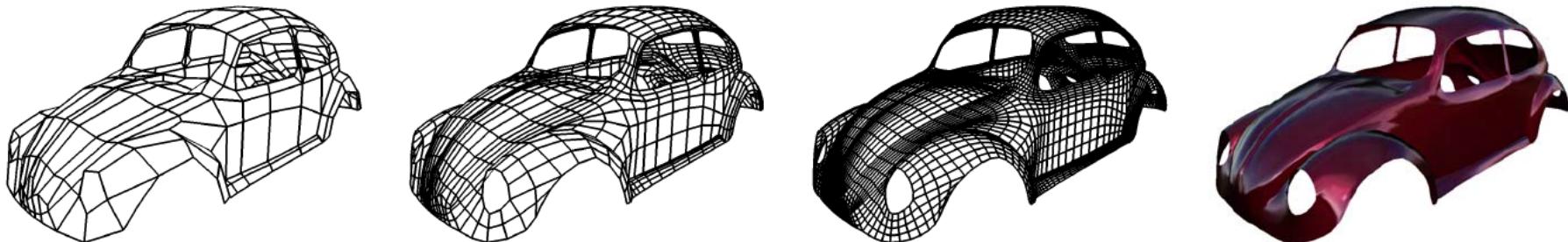
- Generalization of spline curves / surfaces
  - Arbitrary control meshes
  - Successive refinement (subdivision)
  - Converges to smooth limit surface
  - Connection between splines and meshes



# Subdivision Surfaces (细分曲面)

---

- Generalization of spline curves / surfaces
  - Arbitrary control meshes
  - Successive refinement (subdivision)
  - Converges to smooth limit surface
  - Connection between splines and meshes

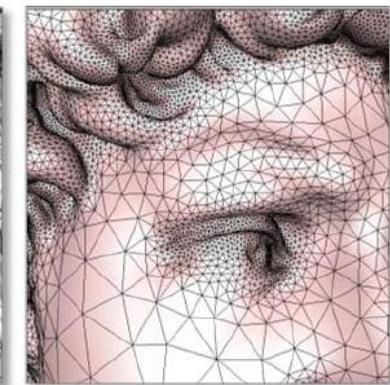
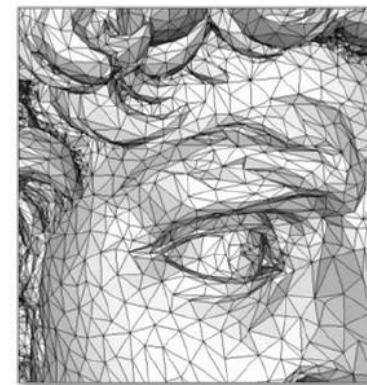
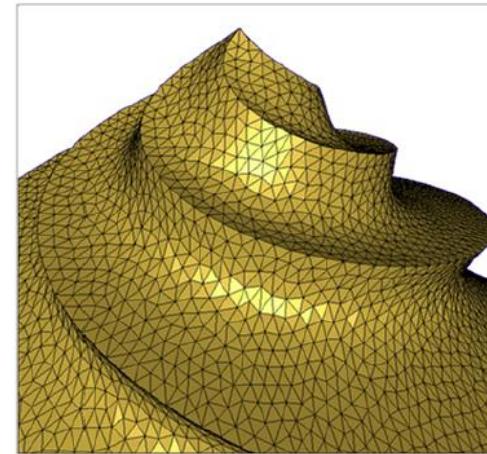


# Discrete Surfaces (离散曲面): Point Sets, Meshes

- Flexible
- Suitable for highly detailed scanned data
- No analytic surface
- No inherent “editability”



Mesh editing



# Mesh Processing & Editing



# Mesh Denoising

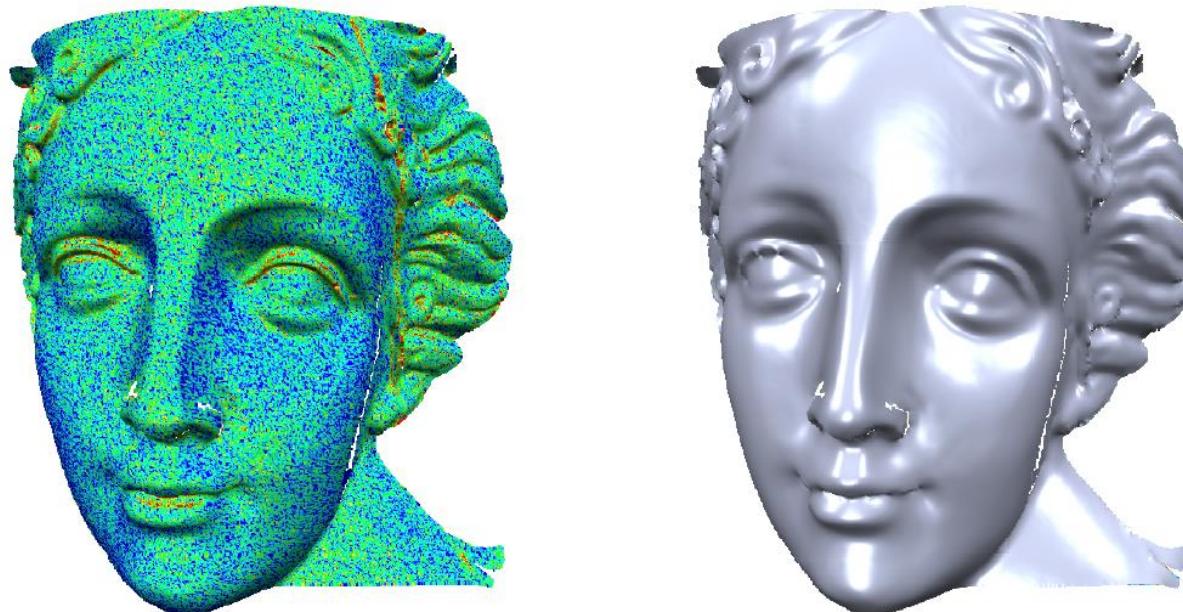
---

- Mesh Denoising (aka Smoothing, Filtering, Fairing)

**Input:** Noisy mesh (scanned or other)

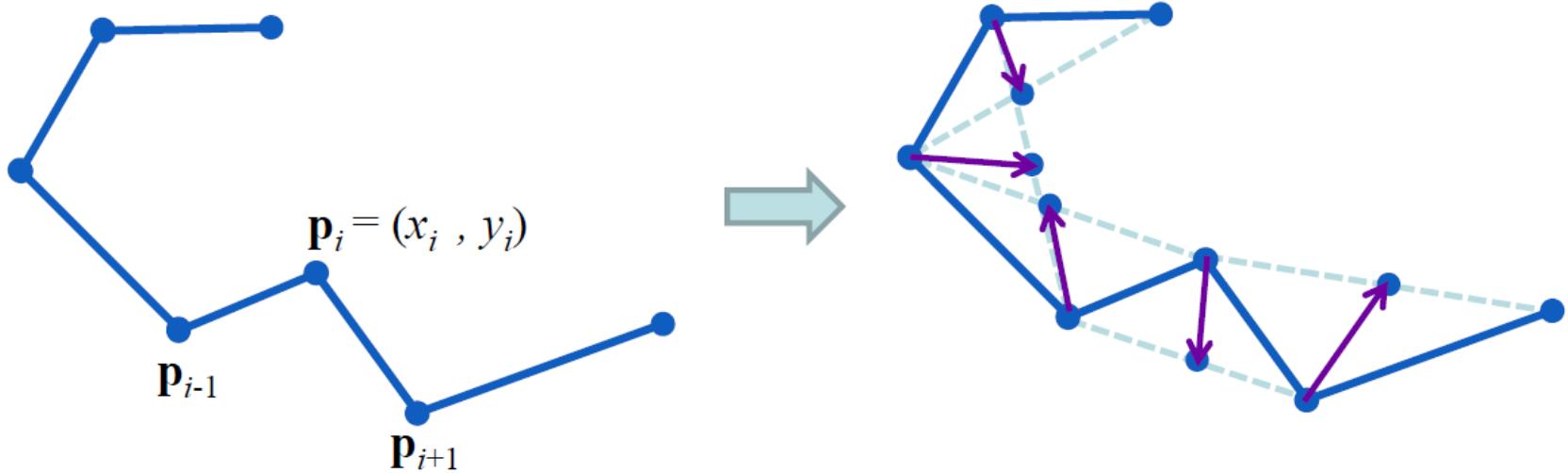
**Output:** Smooth mesh

**How:** Filter out high frequency noise



# Laplacian Smoothing

- An easier problem: How to smooth a curve?



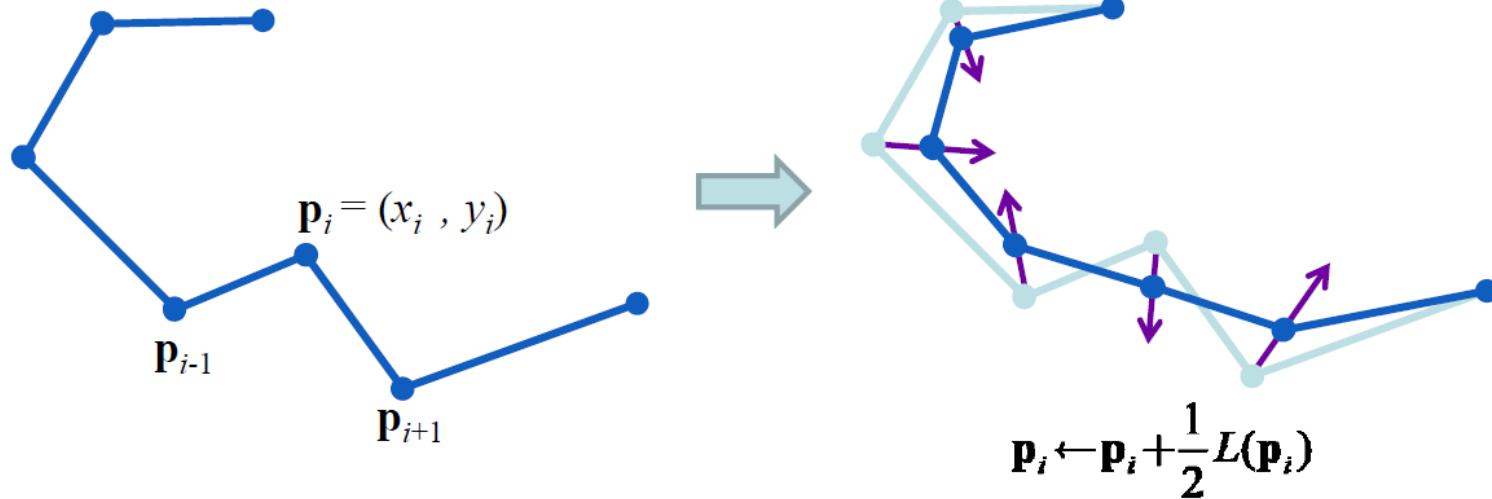
$$(p_{i-1} + p_{i+1})/2 - p_i$$

$$L(p_i) = \frac{1}{2}(p_{i+1} - p_i) + \frac{1}{2}(p_{i-1} - p_i)$$



# Laplacian Smoothing

An easier problem: How to smooth a curve?



Finite difference  
discretization of second  
derivative  
= Laplace operator in  
one dimension

$$L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

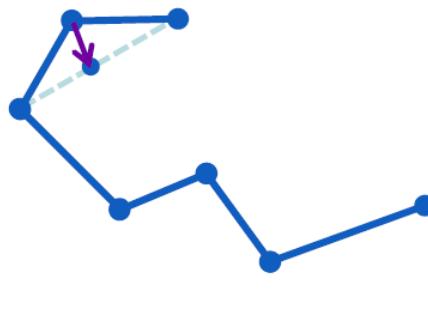


# Laplacian Smoothing on Meshes

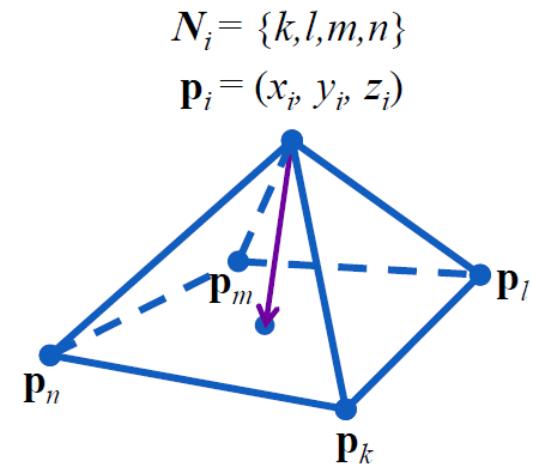
Same as for curves:

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \lambda \Delta \mathbf{p}_i^{(t)}$$

What is  $\Delta \mathbf{p}_i$  ?



$$\frac{1}{2}(\mathbf{p}_{i+1} + \mathbf{p}_{i-1}) - \mathbf{p}_i$$



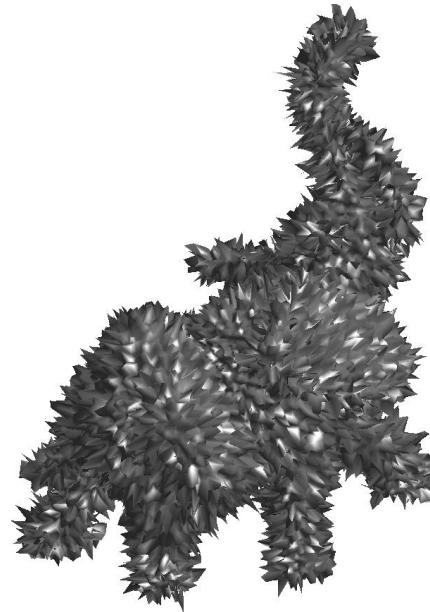
$$\frac{1}{|N_i|} \left( \sum_{j \in N_i} \mathbf{p}_j \right) - \mathbf{p}_i$$



# Mesh Denoising

---

- We generate artificially a noisy mesh by random normal displacement along the normal. We only perform normal displacements because tangential displacements do not impact the geometry of the mesh.



# Mesh Denoising with Filtering

---

The quality of a noisy mesh is improved by applying local averagings, that removes noise but also tends to smooth features.

The operator  $\tilde{W} : \mathbb{R}^n \rightarrow \mathbb{R}^n$  can be used to smooth a function, but it can also be applied to smooth the position  $W \in \mathbb{R}^{3 \times n}$ . Since they are stored as row of a matrix, one should applies  $\tilde{W}^*$  (transposed matrix) on the right side.

$$X^{(0)} = X \quad \text{and} \quad X^{(\ell+1)} = X^{(\ell)} W^*$$



# Mesh Denoising with Filtering

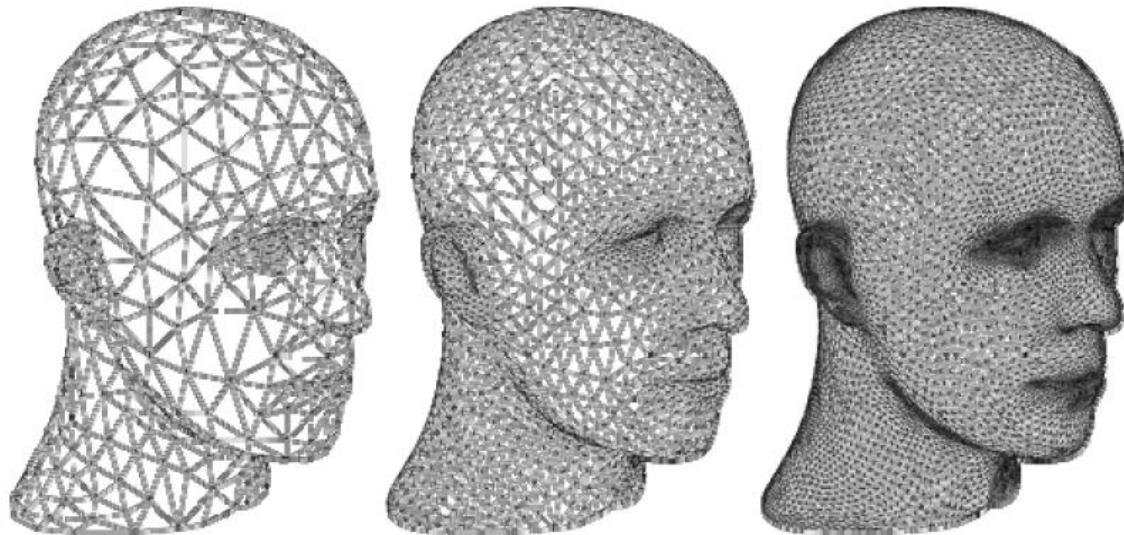
---



# Mesh Subdivision

---

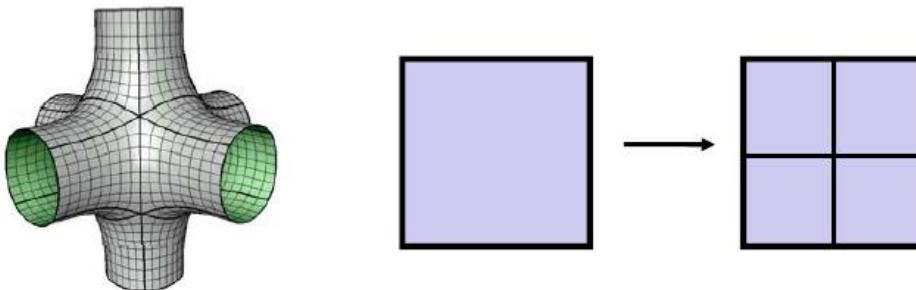
- No regular structure as for curves
  - Arbitrary number of edge-neighbors
  - Different subdivision rules for each valence



# Subdivision Rules

---

- How the connectivity changes



- How the geometry changes
  - Old points
  - New points



# Subdivision Zoo

---

- Classification of subdivision schemes

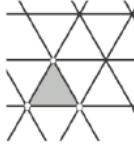
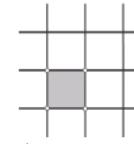
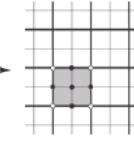
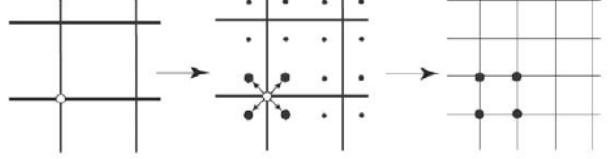
<b>Primal</b>	Faces are split into sub-faces
<b>Dual</b>	Vertices are split into multiple vertices

<b>Approximating</b>	Control points are not interpolated
<b>Interpolating</b>	Control points are interpolated



# Subdivision Zoo

- Classification of subdivision schemes

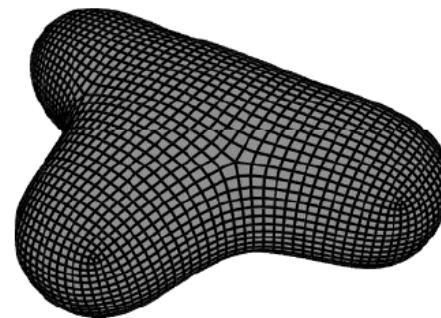
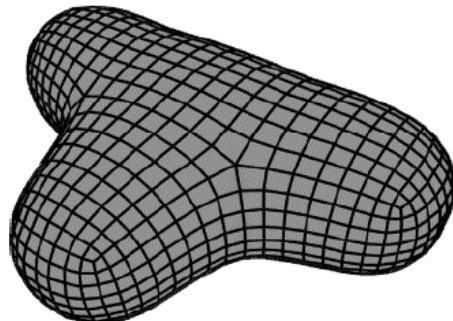
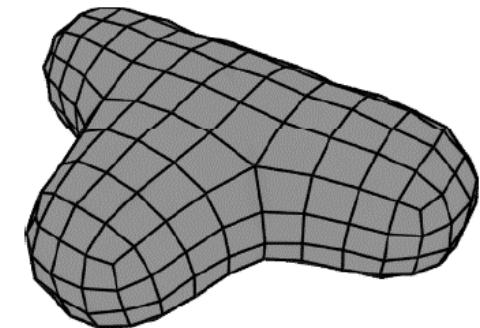
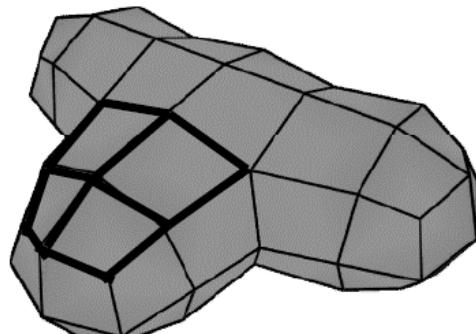
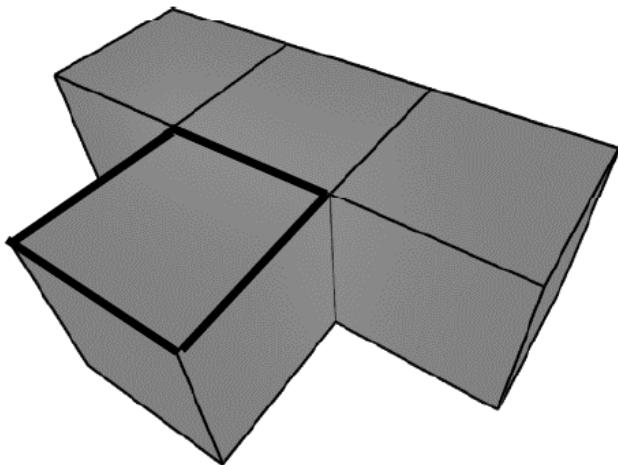
Primal (face split)		
	 → 	 → 
	<i>Triangular meshes</i>	<i>Quad Meshes</i>
Approximating	Loop( $C^2$ )	Catmull-Clark( $C^2$ )
Interpolating	Mod. Butterfly ( $C^1$ )	Kobbelt ( $C^1$ )
Dual (vertex split)		
		
<b>Dual (vertex split)</b>		
Doo-Sabin, Midedge( $C^1$ )		
Biquartic ( $C^2$ )		

- Many more...



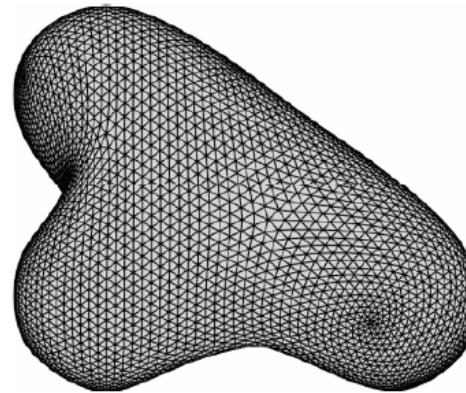
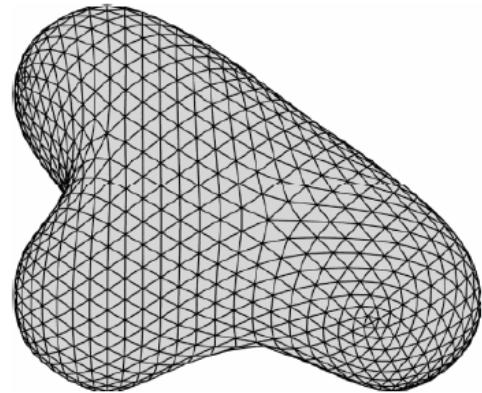
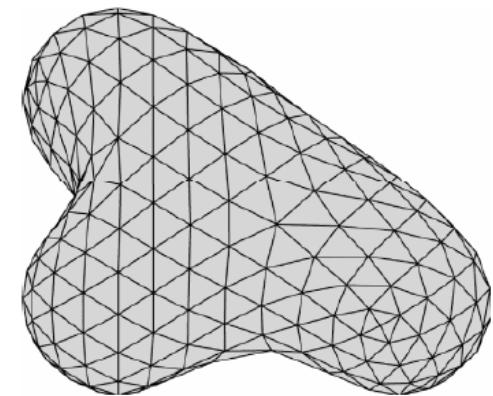
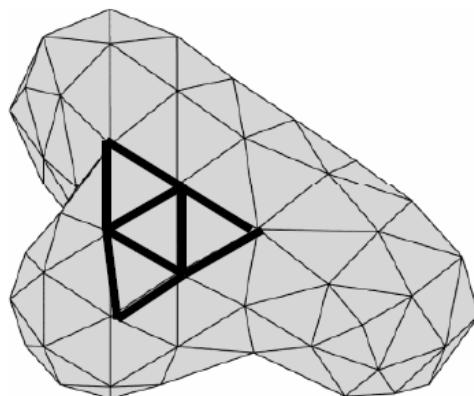
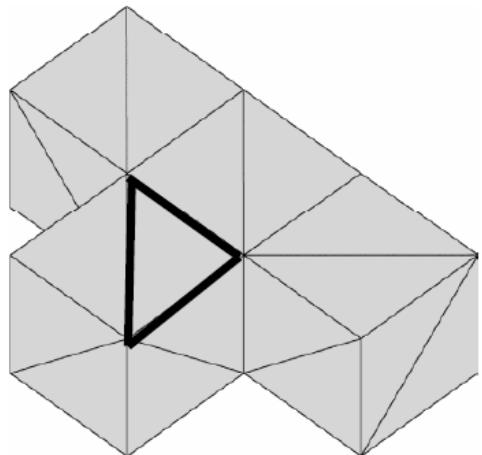
# Catmull-Clark Subdivision

---



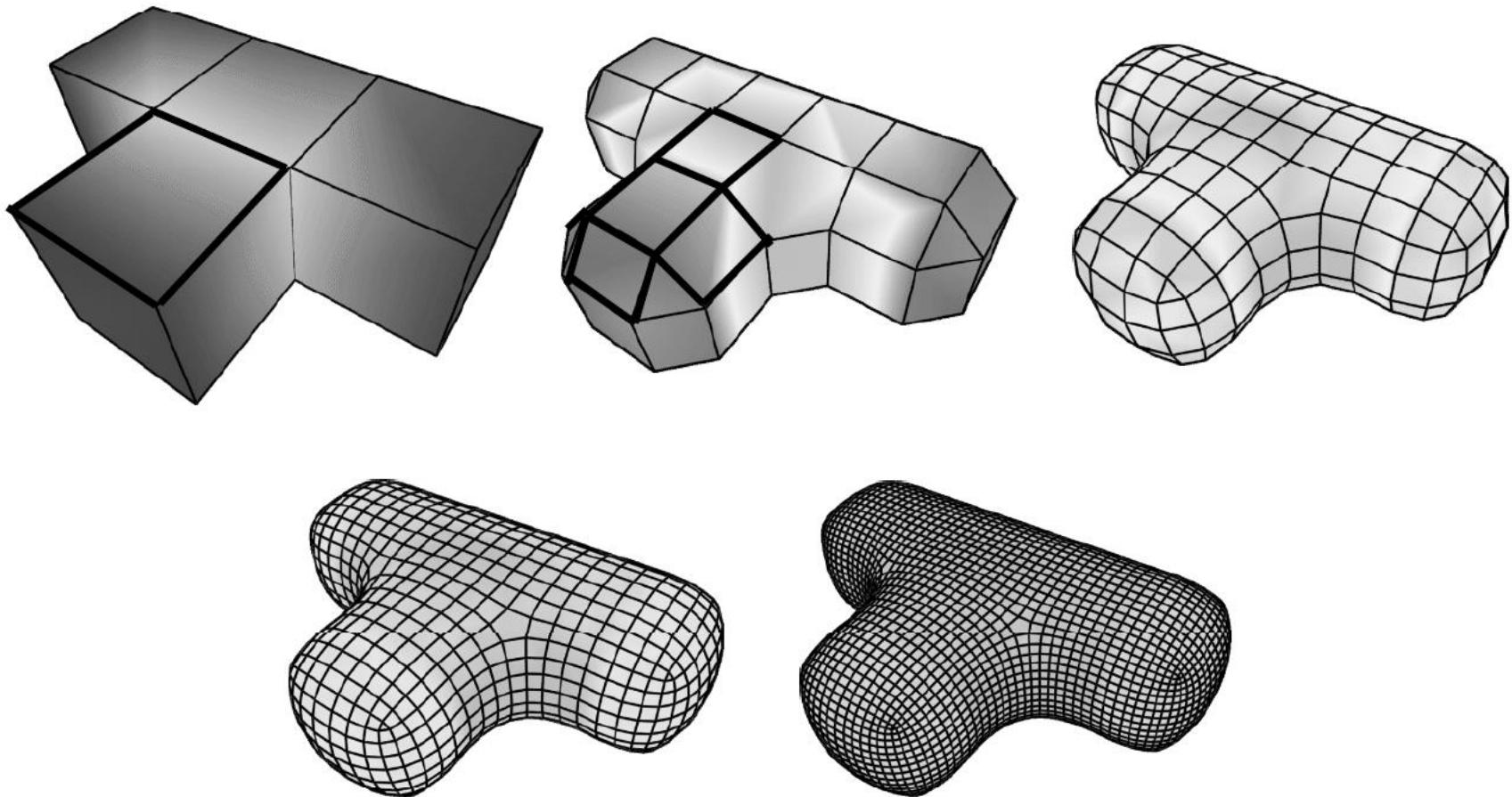
# Loop Subdivision

---



# Doo-Sabin Subdivision

---



# Mesh Simplification

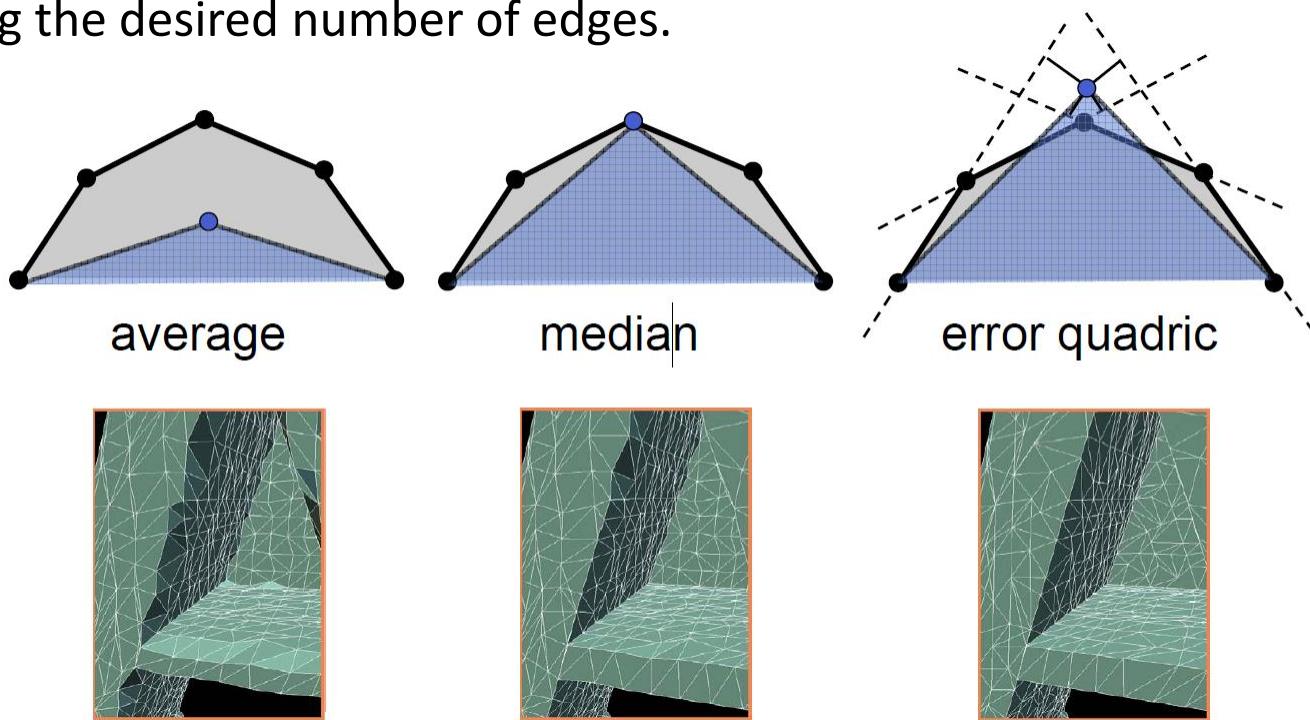
---

- Surface mesh simplification is the process of reducing the number of faces used in a surface mesh while keeping the overall shape, volume and boundaries preserved as much as possible. It is the opposite of subdivision.



# Mesh Simplification

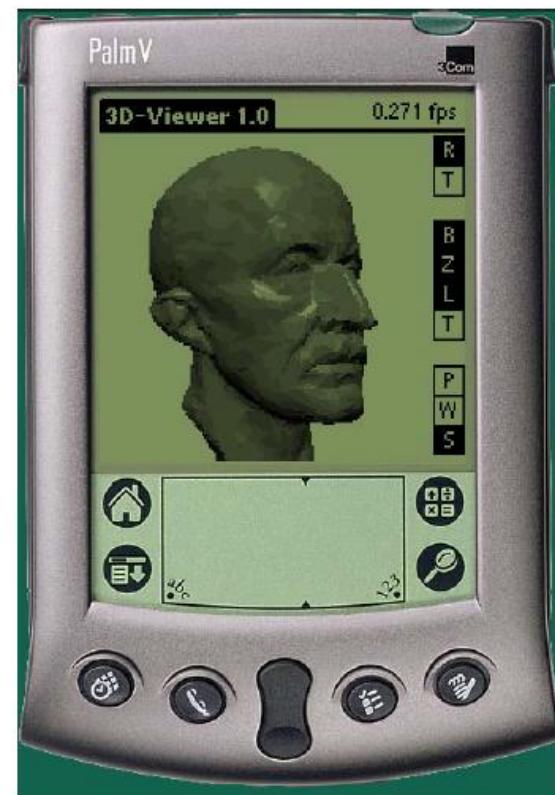
- Edges are collapsed according to a priority given by a user-supplied cost function, and the coordinates of the replacing vertex are determined by another user-supplied placement function.
- The algorithm terminates when a user-supplied stop predicate is met, such as reaching the desired number of edges.



# Mesh Simplification

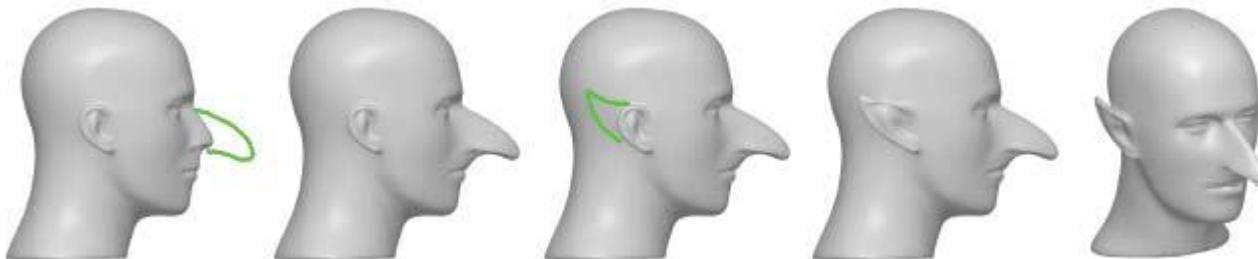
---

- Adaptation to hardware capabilities



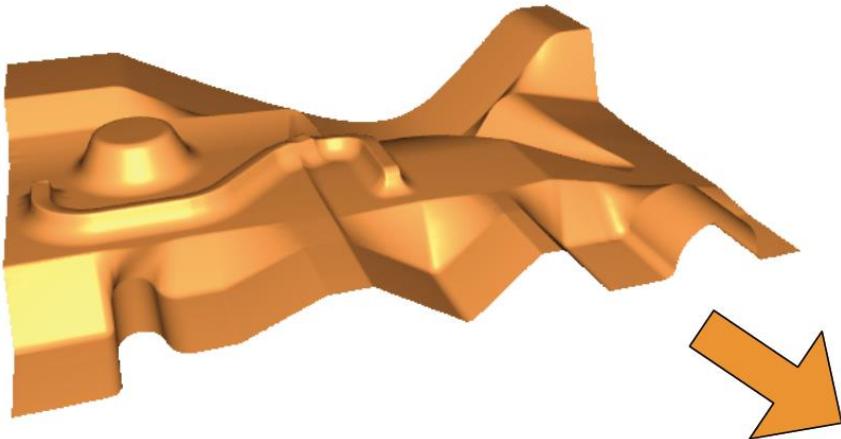
# Shapes and Deformations

- Why deformations?
  - Sculpting, customization
  - Character posing, animation
- Criteria?
  - Intuitive behavior and interface
  - Interactivity

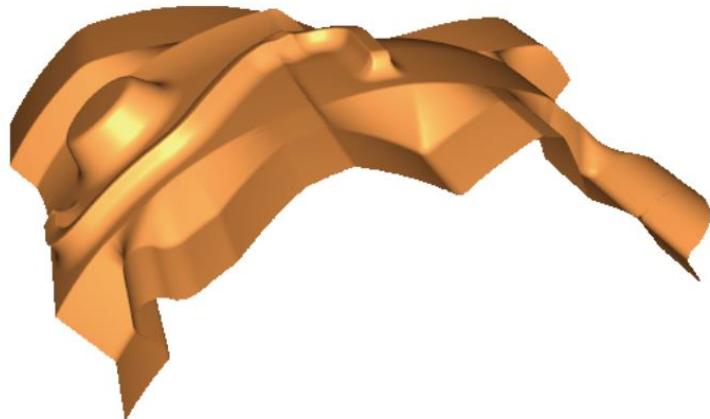


# Linear Surface-Based Deformation

- Mesh Deformation

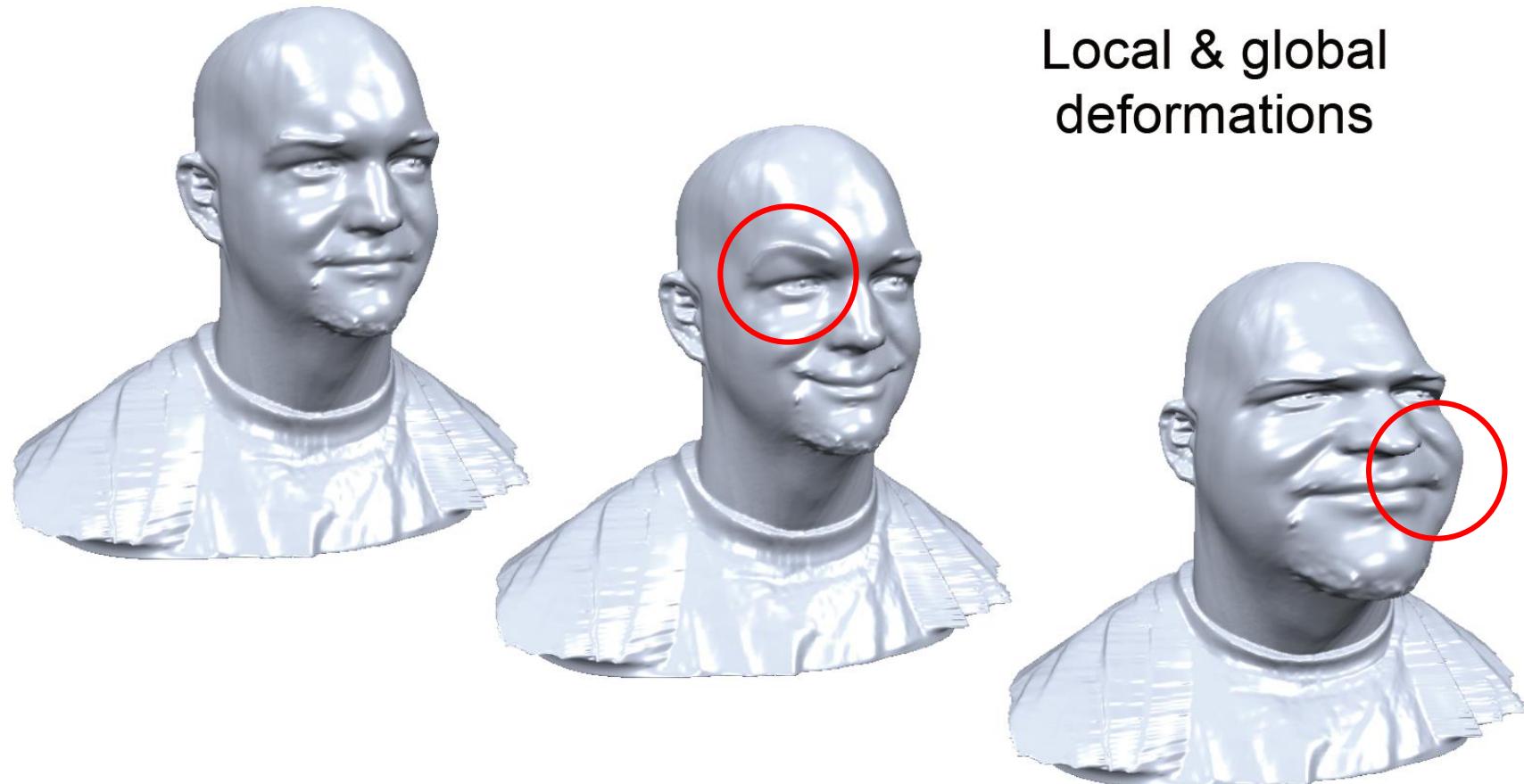


Global deformation  
with intuitive  
detail preservation



# Mesh Deformation

---

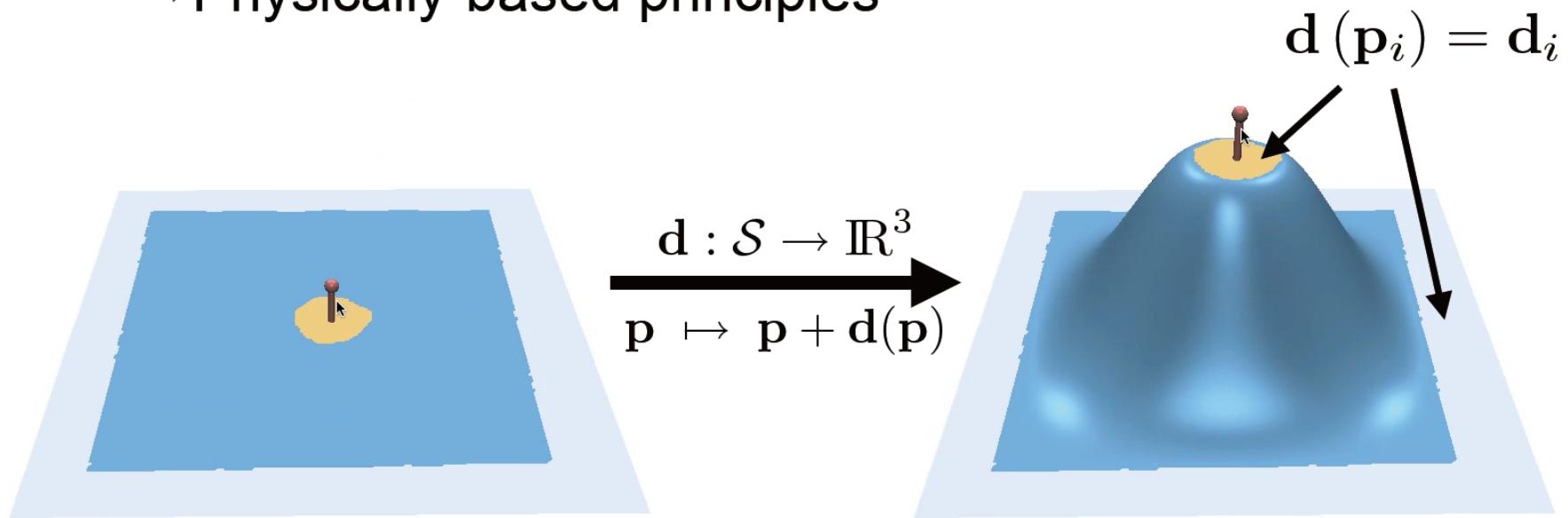


Local & global  
deformations



# Physically-Based Deformation

- Mesh deformation by displacement function  $\mathbf{d}$ 
    - Interpolate prescribed constraints
    - Smooth, intuitive deformation
- Physically-based principles



# Physically-Based Deformation

---

- Nonlinear stretching & bending energies

$$\int_{\Omega} k_s \left\| \mathbf{I} - \mathbf{I}' \right\|^2 + k_b \left\| \mathbf{II} - \mathbf{II}' \right\|^2 \, dudv$$

stretching    bending

- Linearize terms → Quadratic energy

$$\int_{\Omega} k_s \left( \|\mathbf{d}_u\|^2 + \|\mathbf{d}_v\|^2 \right) + k_b \left( \|\mathbf{d}_{uu}\|^2 + 2 \|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \right) \, dudv$$

stretching    bending



# Physically-Based Deformation

---

- Minimize linearized bending energy

$$E(\mathbf{d}) = \int_S \|\mathbf{d}_{uu}\|^2 + 2\|\mathbf{d}_{uv}\|^2 + \|\mathbf{d}_{vv}\|^2 \, dudv \rightarrow \min$$

$f(x) \rightarrow \min$

- Variational calculus  $\rightarrow$  Euler-Lagrange PDE

$$\Delta^2 \mathbf{d} := \mathbf{d}_{uuuu} + 2\mathbf{d}_{uuvv} + \mathbf{d}_{vvvv} = 0$$

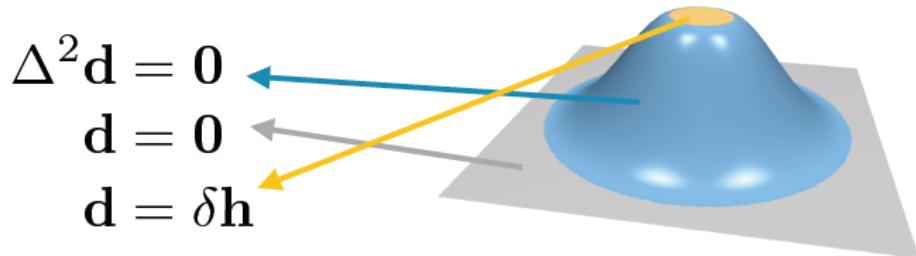
$f'(x) = 0$

→ “Best” deformation that satisfies constraints



# PDE Discretization

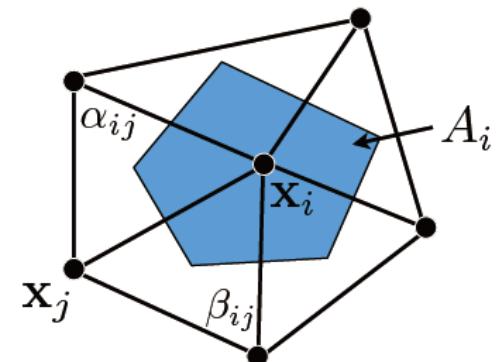
- Euler-Lagrange PDE



- Laplace discretization

$$\Delta \mathbf{d}_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}_i} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{d}_j - \mathbf{d}_i)$$

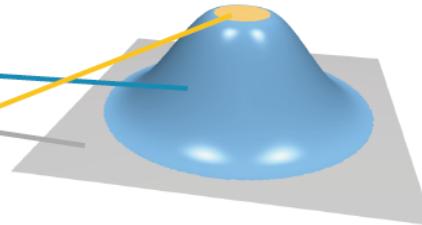
$$\Delta^2 \mathbf{d}_i = \Delta(\Delta \mathbf{d}_i)$$



# Linear System

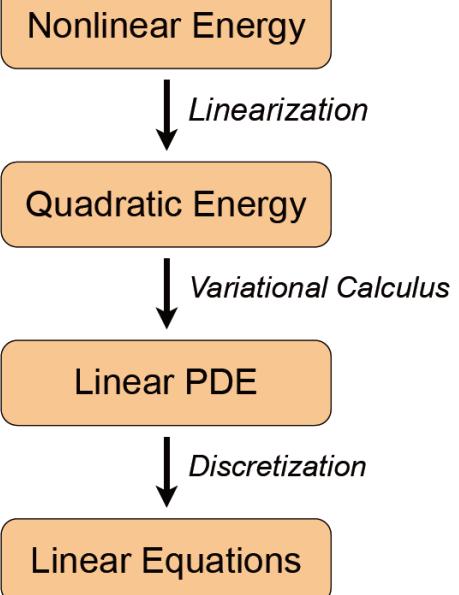
- Sparse linear system (19 nz)row)

$$\begin{pmatrix} \Delta^2 & & \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \vdots \\ d_i \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \delta h_i \end{pmatrix}$$

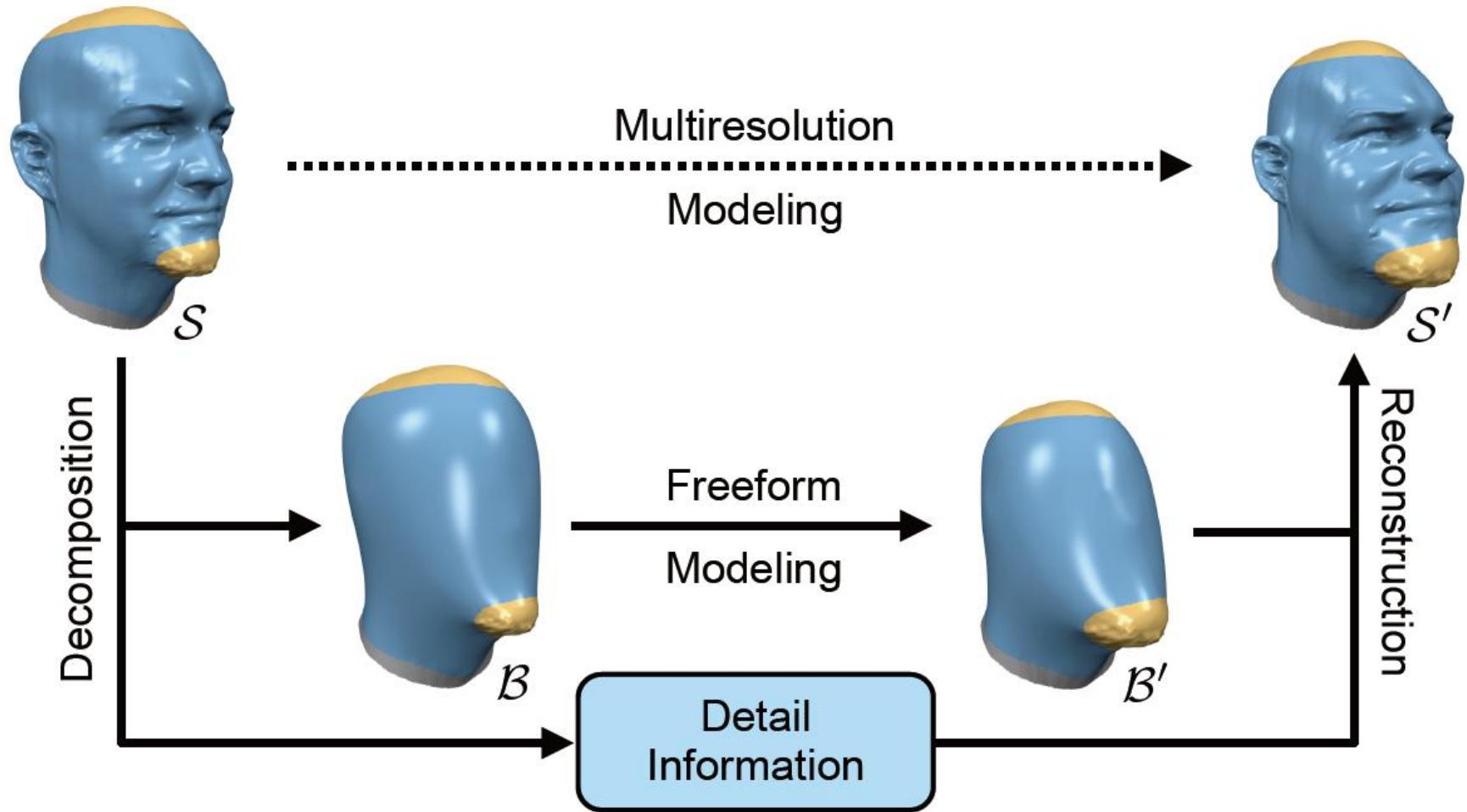


- Turn into symmetric positive definite system
- Solve this system each frame
  - Use efficient linear solvers !!!
  - Sparse Cholesky factorization

Derivation Steps



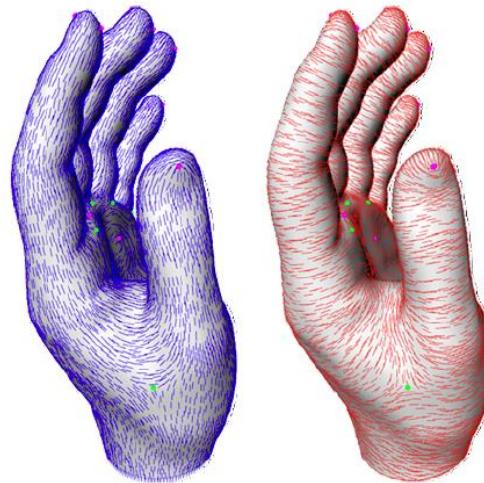
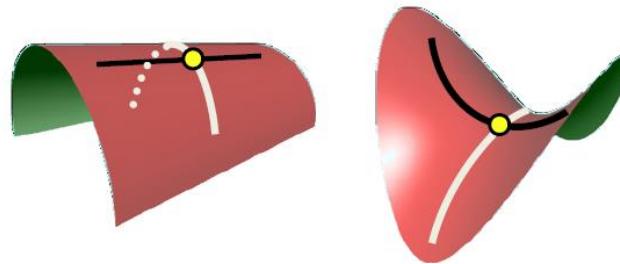
# Multi-resolution Deformation



# Differential Geometry

---

- Tool to analyze shape
- Key notions:
  - Tangents and normals
  - Curvatures
  - Laplace-Beltrami operator



# Differential Coordinates

---

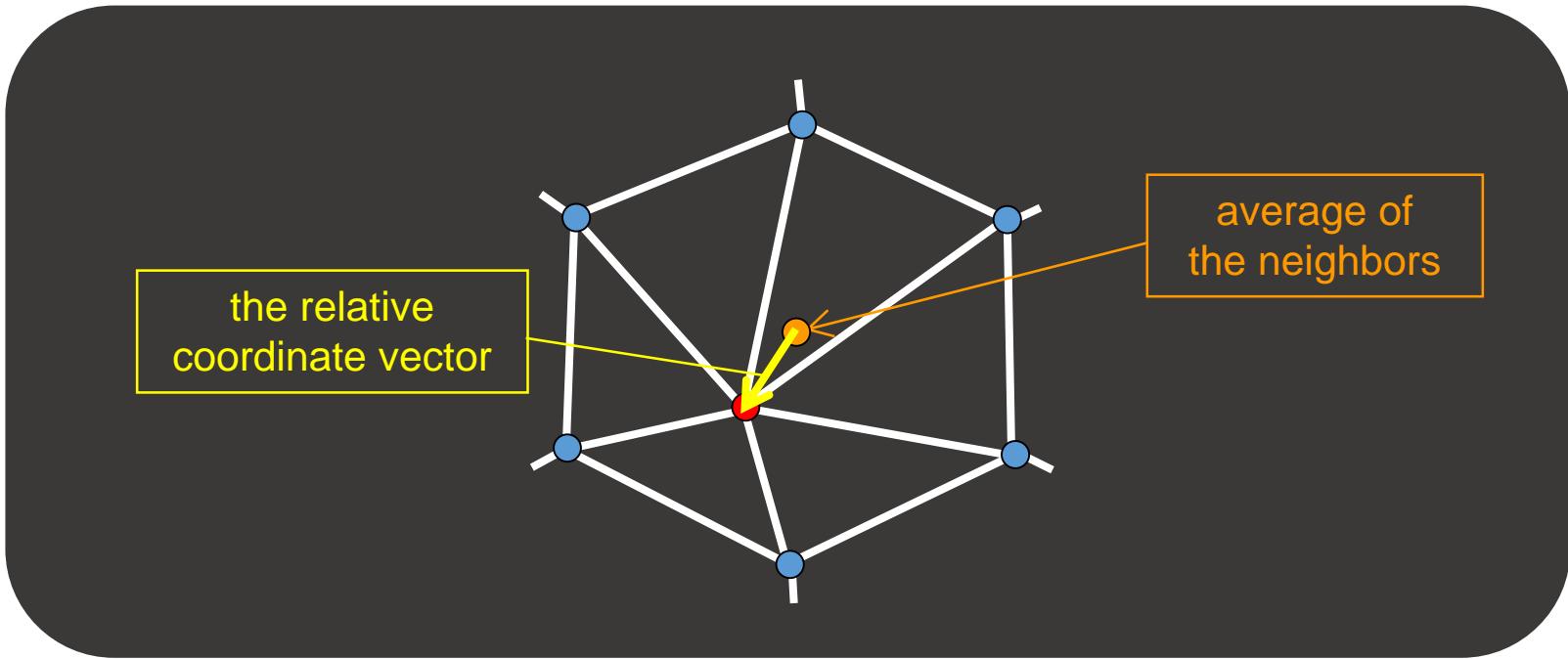
- Manipulate differential coordinates instead of spatial coordinates
  - Gradients, Laplacians, local frames
  - Intuition: Close connection to surface normal
- Find mesh with desired differential coordinates
  - Cannot be solved exactly
  - Formulate as energy minimization



# Differential coordinates

- Differential coordinates are defined for triangular mesh vertices

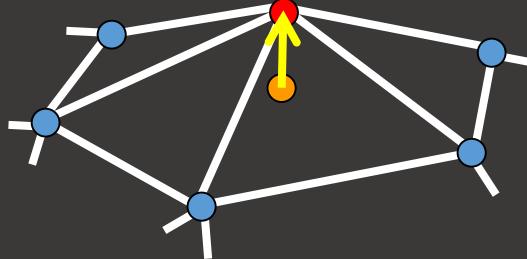
$$\delta_i = L(v_i) = v_i - \frac{1}{d_i} \sum_{j \in N(i)} v_j$$



# Differential coordinates

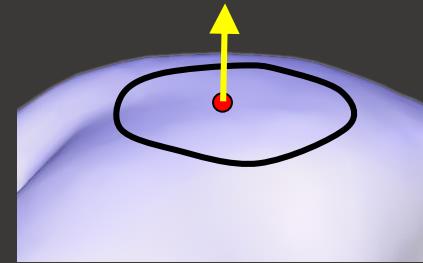
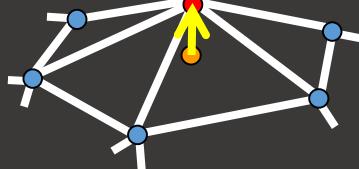
- Differential coordinates are defined for triangular mesh vertices

$$\delta_i = L(v_i) = v_i - \frac{1}{d_i} \sum_{j \in N(i)} v_j$$



# Why differential coordinates?

- They represent the local detail / local shape description
  - The direction approximates the normal
  - The size approximates the mean curvature



$$\delta_i = \frac{1}{d_i} \sum_{v \in N(i)} (v_i - v)$$

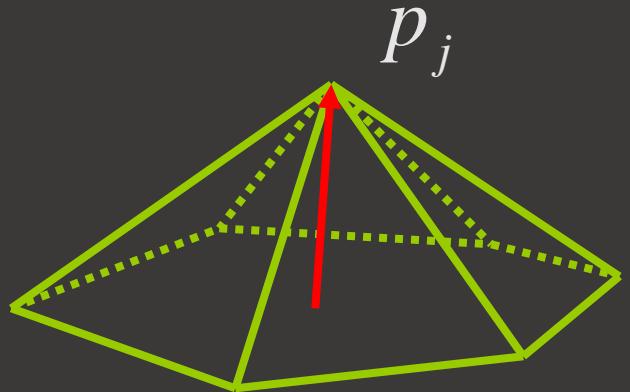
$$\frac{1}{len(\gamma)} \int_{v \in \gamma} (v_i - v) ds$$

$$\lim_{len(\gamma) \rightarrow 0} \frac{1}{len(\gamma)} \int_{v \in \gamma} (v_i - v) ds = H(v_i) n_i$$

# Laplacian reconstruction

- Denote by  $G = (V, E, P)$  a triangular mesh with geometry  $P$ , embedded in  $R^3$ .
- For each vertex  $p_j \in P$  we define the Laplacian vector:

$$L_j(P) = p_j - \frac{1}{d_j} \sum_{i:(i,j) \in E} p_i$$



- The Laplacians represents the details locally.

# Laplacian reconstruction

---

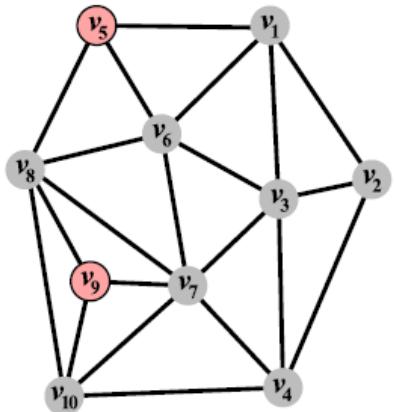
- The operator  $L$  is **linear** and thus can be represented by the following matrix:

$$M_{ij} = \begin{cases} 1 & i = j \\ -\frac{1}{d_i} & j \in \{j : (j, i) \in E\} \\ 0 & otherwise \end{cases}$$



# Laplacian reconstruction

- A small example of a triangular mesh and its associated Laplacian matrix



The mesh

$$\begin{bmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 & -1 \\ -1 & -1 & 5 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & & 3 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 6 & -1 \\ -1 & -1 & -1 & 6 & -1 \\ -1 & -1 & -1 & -1 & 3 \\ -1 & & & -1 & 4 \end{bmatrix}$$

The symmetric Laplacian  $L_s$

$$\begin{bmatrix} 4 & -1 & -1 & & -1 & & \\ -1 & 3 & -1 & -1 & & & \\ -1 & -1 & 5 & -1 & -1 & -1 & \\ -1 & -1 & -1 & 4 & -1 & -1 & \\ -1 & -1 & -1 & -1 & 4 & -1 & \\ -1 & -1 & -1 & -1 & -1 & 4 & \\ -1 & -1 & -1 & -1 & -1 & -1 & \\ -1 & -1 & -1 & -1 & -1 & -1 & \\ -1 & -1 & -1 & -1 & -1 & -1 & \\ -1 & & & & & & \end{bmatrix}$$

Invertible Laplacian

$$\begin{bmatrix} 4 & -1 & -1 & -1 & -1 & & \\ -1 & 3 & -1 & -1 & -1 & & \\ -1 & -1 & 5 & -1 & -1 & -1 & \\ -1 & -1 & -1 & 4 & -1 & -1 & \\ -1 & -1 & -1 & -1 & 3 & -1 & \\ -1 & -1 & -1 & -1 & -1 & 4 & \\ -1 & -1 & -1 & -1 & -1 & -1 & \\ -1 & -1 & -1 & -1 & -1 & -1 & \\ -1 & -1 & -1 & -1 & -1 & -1 & \\ -1 & & & & & & \end{bmatrix}$$

2-anchor  $\tilde{L}$



# Laplacian reconstruction

---

- Thus for reconstructing the mesh from the Laplacian representation:  
add constraints to get full rank system and therefore unique solution, i.e. unique minimizer to the functional

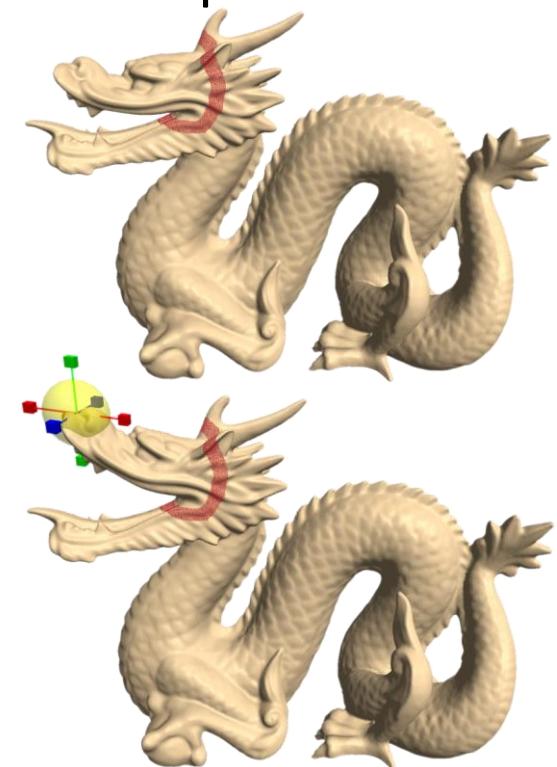
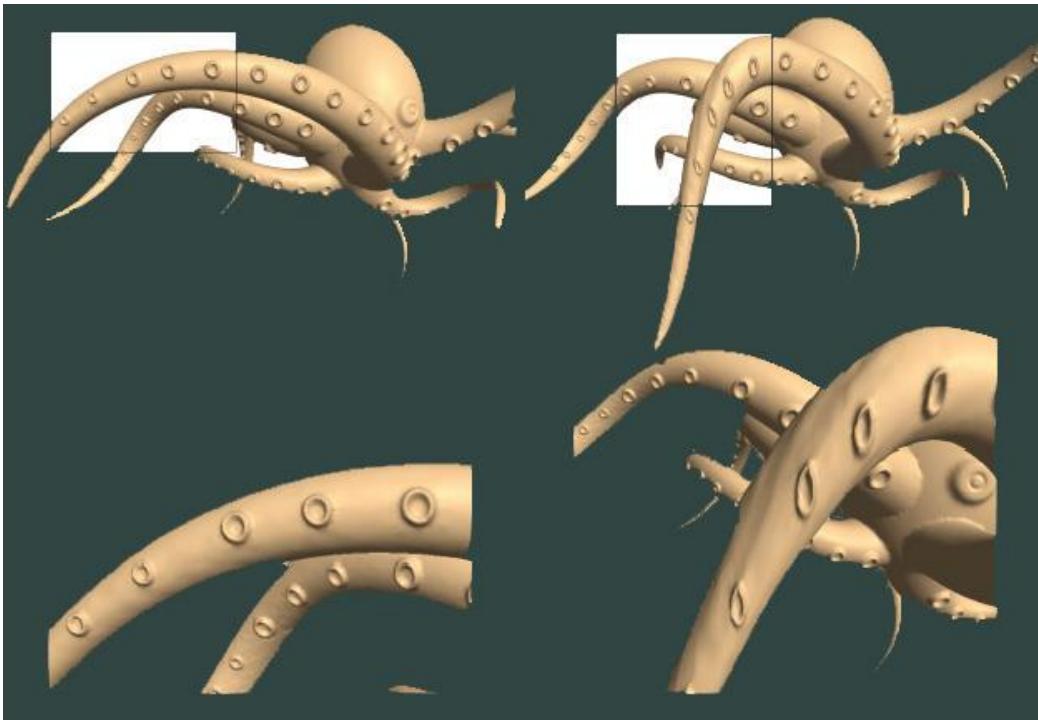
$$\|M \cdot P^{(x)} - \delta^{(x)}\|^2 + \sum_{i \in I} w_i (p_i^{(x)} - c_i^{(x)})^2$$

where  $I$  is the index set of constrained vertices ,  $w_i > 0$  are weights and  $c_i$  are the spatial constraints.



# Laplacian reconstruction

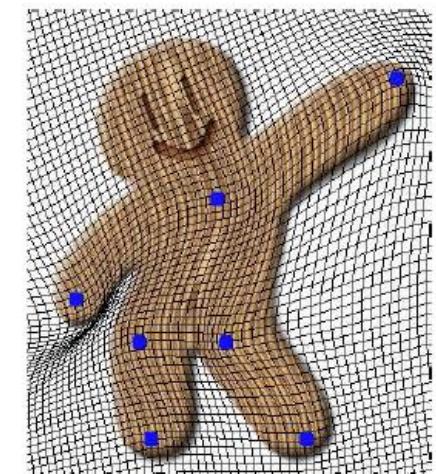
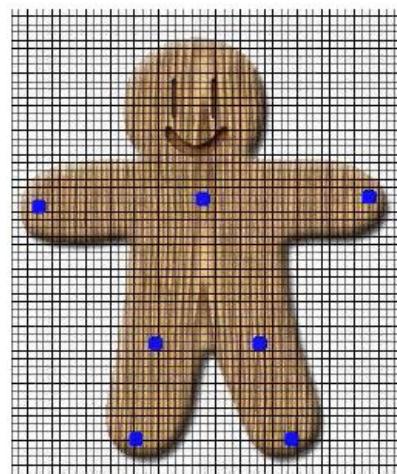
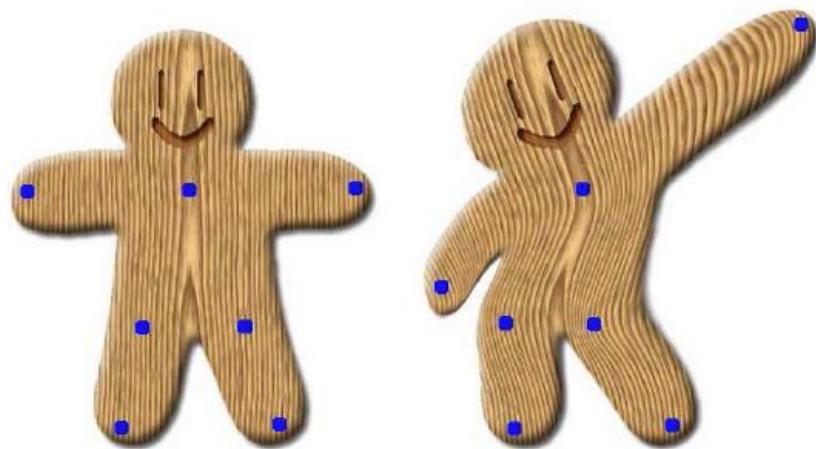
- Laplacian reconstruction gives smooth transformation, interactive time and ease of user interface -using few spatial constraints
- but doesn't preserve details orientation and shape



# As-Rigid-As-Possible Deformation

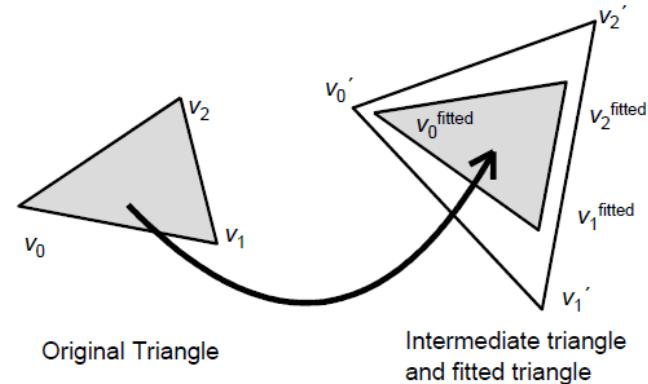
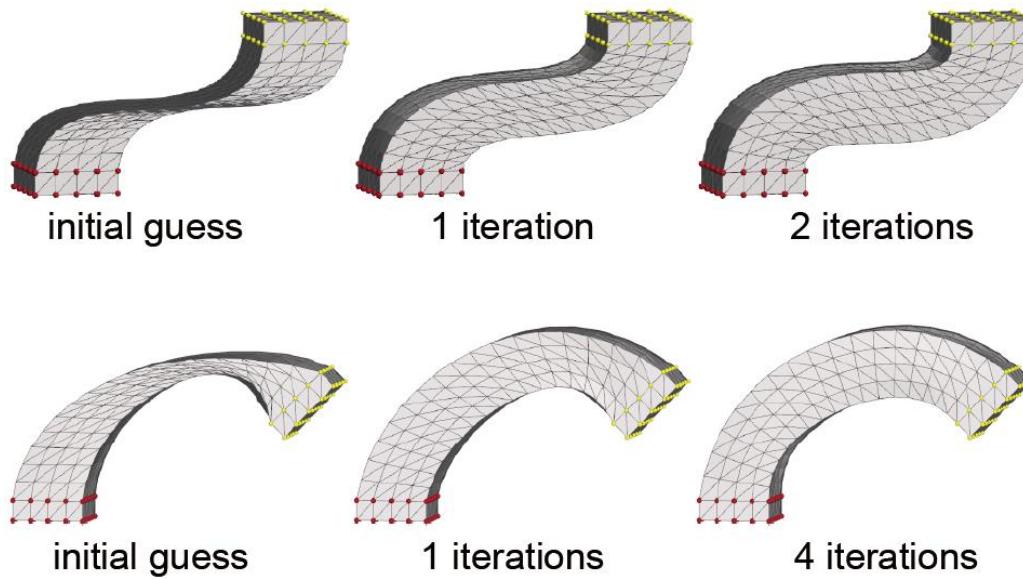
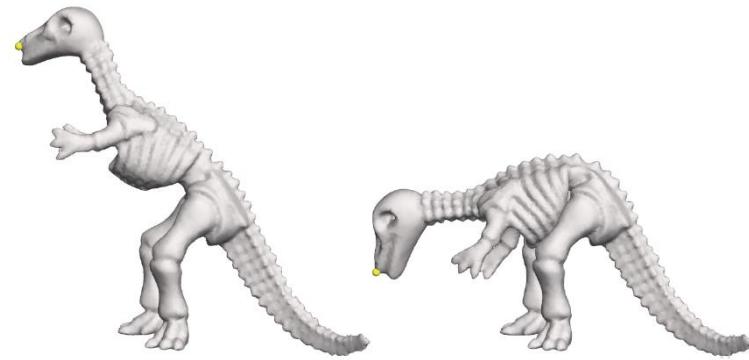
---

- Points or segments as control objects
- First developed in 2D and later extended to 3D by Zhu and Gortler (2007)



# As-Rigid-As-Possible Deformation

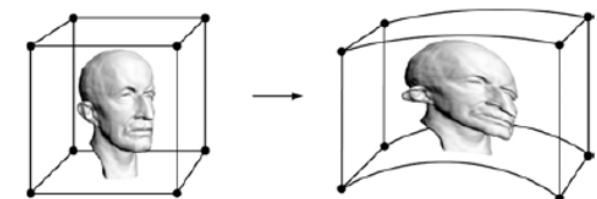
- Smooth large scale deformation
- Local as-rigid-as-possible behavior
  - Preserves small-scale details



# Space Deformation

- Displacement function defined on the ambient space

$$\mathbf{d} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$$



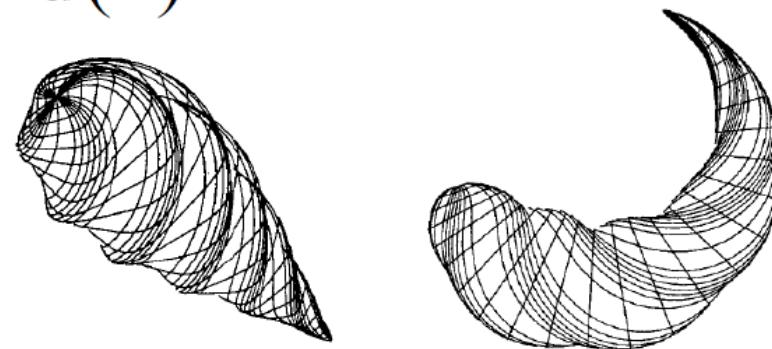
- Evaluate the function on the points of the shape embedded in the space

$$\mathbf{x}' = \mathbf{x} + \mathbf{d}(\mathbf{x})$$

Twist warp

Global and local deformation of solids

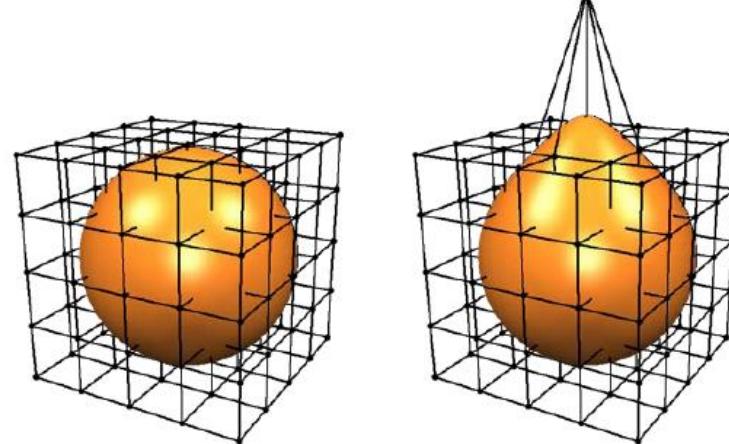
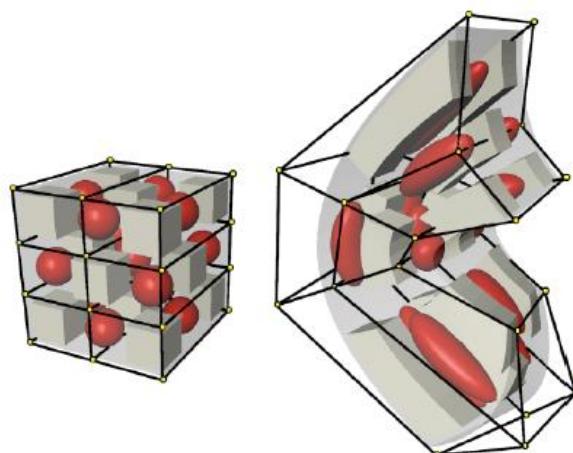
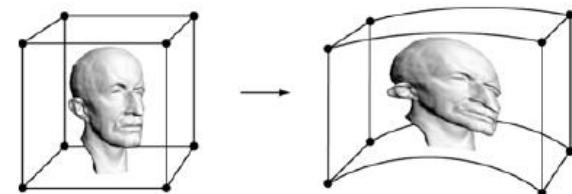
[A. Barr, SIGGRAPH 84]



# Space Deformation

- Control object = lattice
- Basis functions  $B_i(\mathbf{x})$  are trivariate tensor-product splines:

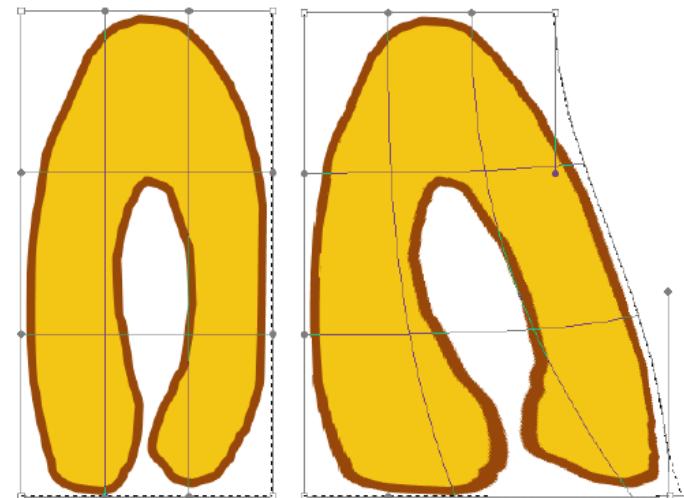
$$\mathbf{d}(x, y, z) = \sum_{i=0}^l \sum_{j=0}^m \sum_{k=0}^n \mathbf{d}_{ijk} N_i(x) N_j(y) N_k(z)$$



# Lattice as Control Object

---

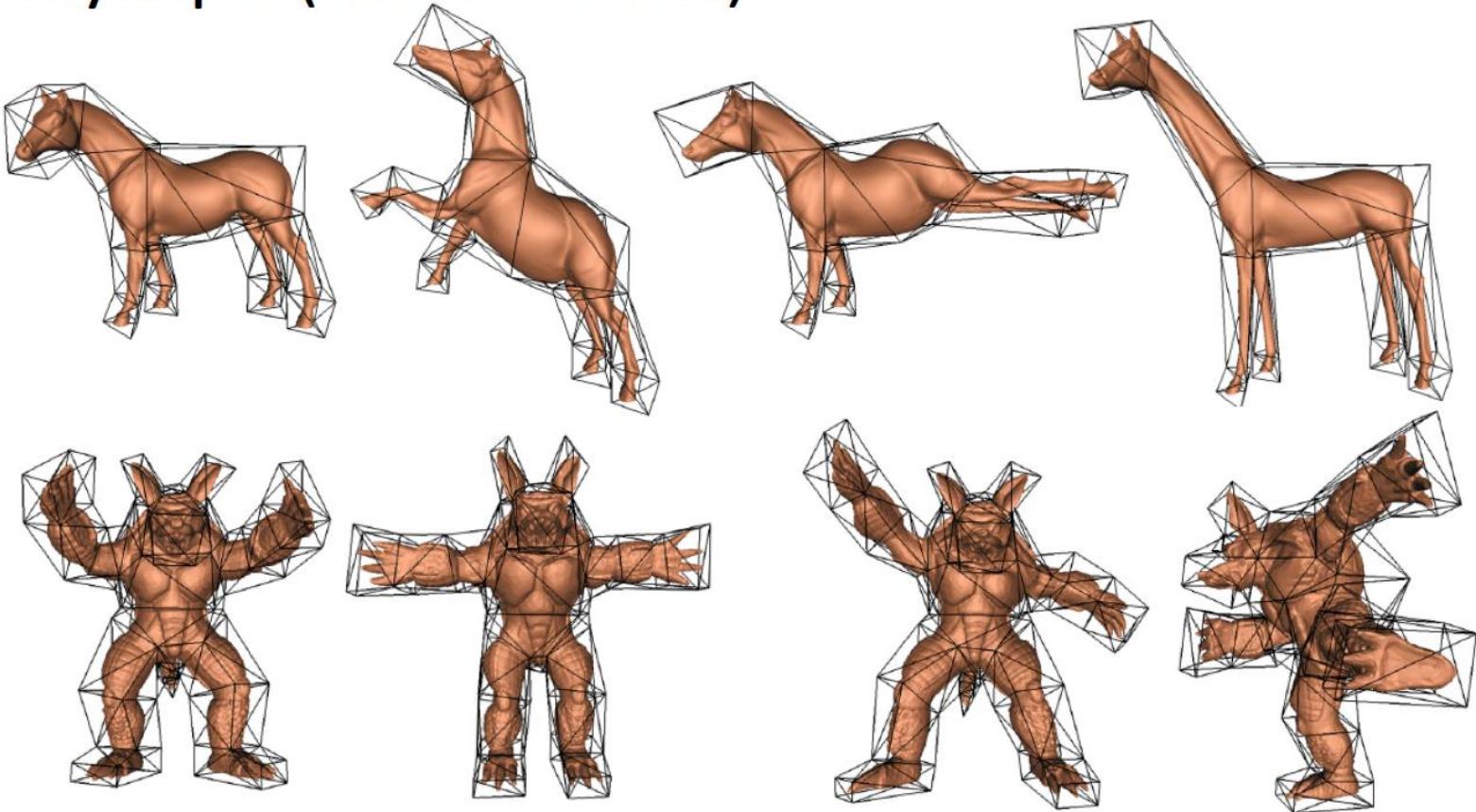
- Difficult to manipulate
- The control object is not related to the shape of the edited object
- Part of the shape in close Euclidean distance always deform similarly, even if geodesically far



# Cage-based Deformations

- Cage = crude version of the input shape
- Polytope (not a lattice)

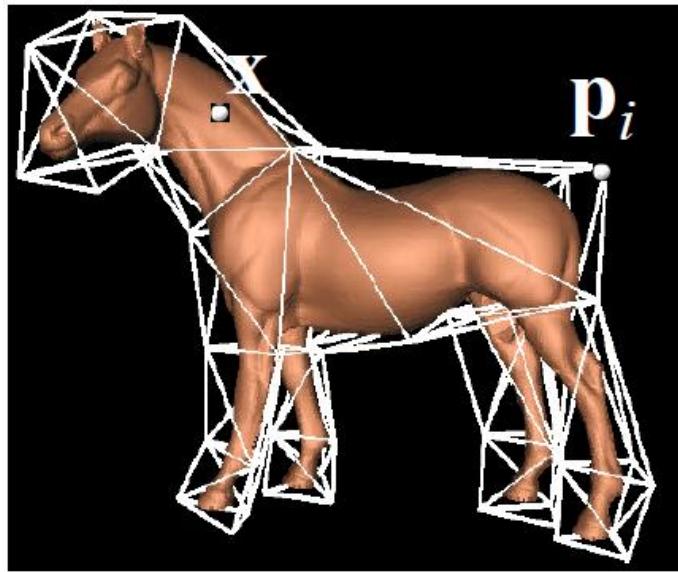
[Ju et al. 2005]



# Cage-based Deformations

---

- Cage = crude version of the input shape
- Polytope (not a lattice)
- Each point  $\mathbf{x}$  in space is represented w.r.t. to the cage elements using coordinate functions



$$\mathbf{x} = \sum_{i=1}^k w_i(\mathbf{x}) \mathbf{p}_i$$

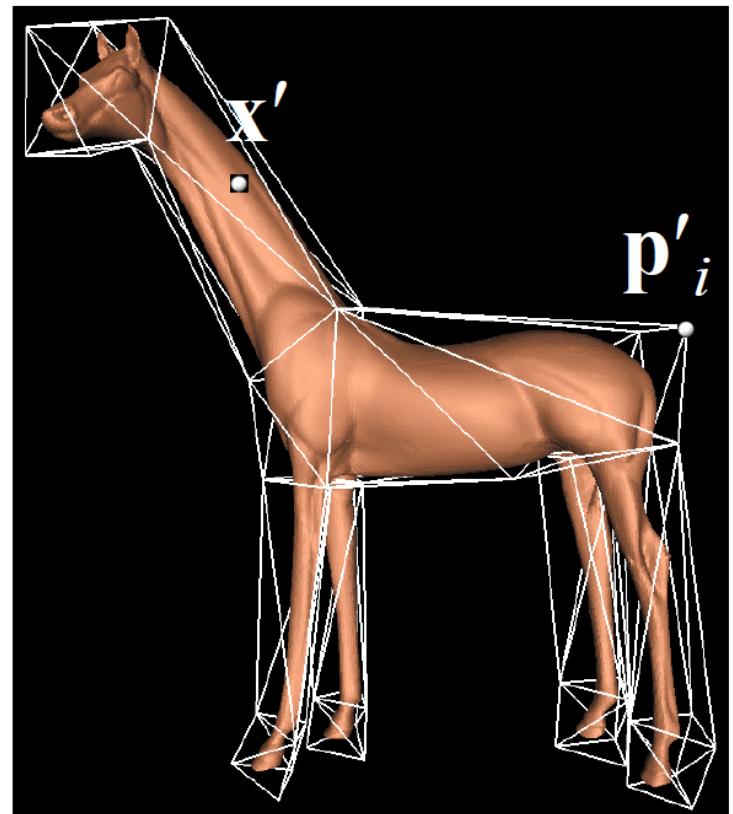
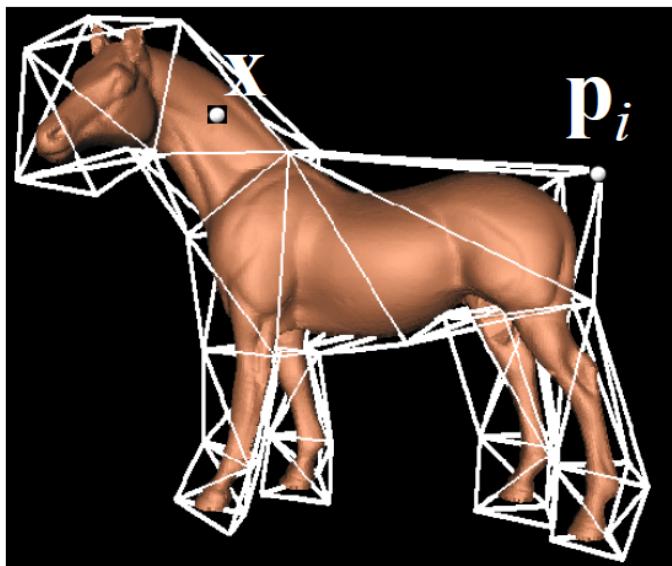
[Ju et al. 2005]



# Cage-based Deformations

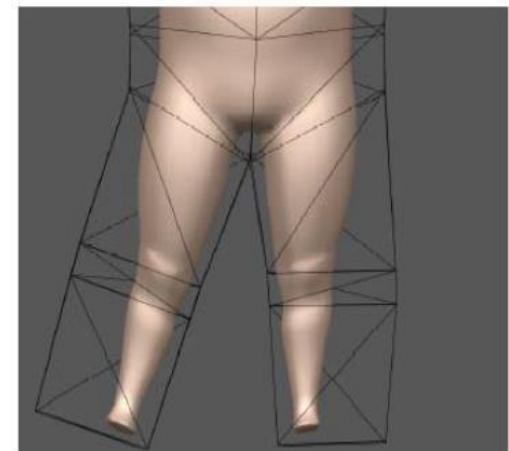
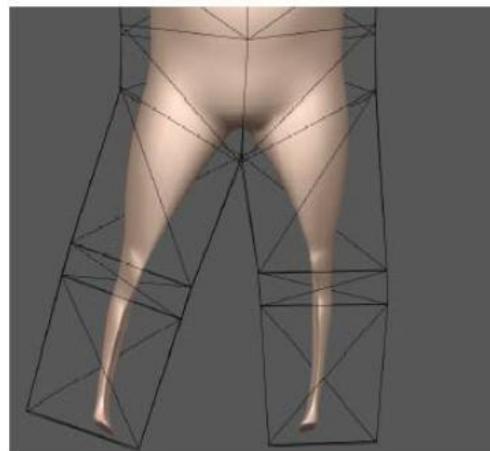
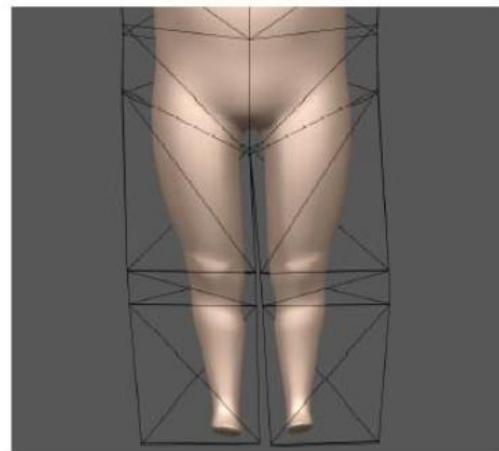
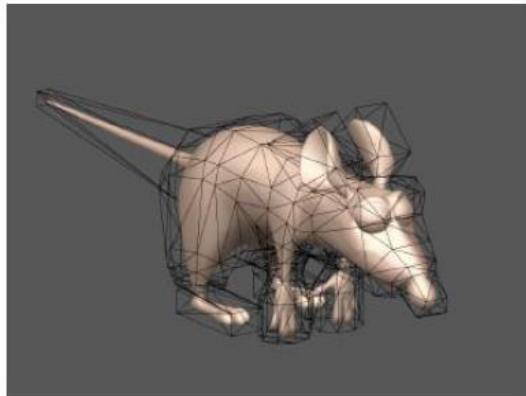
- Cage = crude version of the input shape
- Polytope (not a lattice)

$$\mathbf{x}' = \sum_{i=1}^k w_i(\mathbf{x}) \mathbf{p}'_i$$



# Coordinate Functions

- Harmonic coordinates (Joshi et al. 2007)



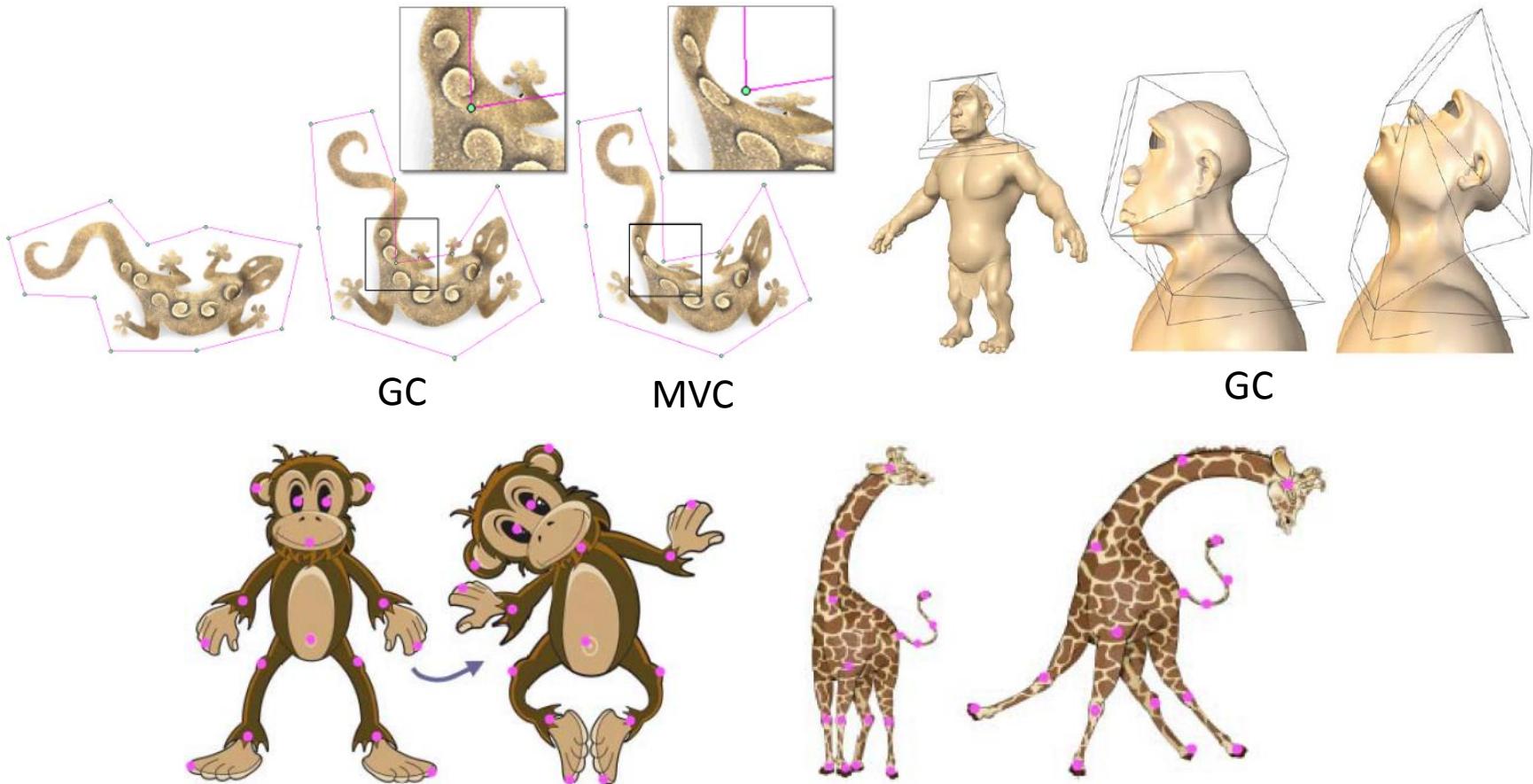
MVC

HC



# Green coordinates

- Closed-form solution
- Conformal in 2D, quasi-conformal in 3D



# Polygon Mesh Processing

---

- <http://www.pmp-book.org/>
- “Geometric Modeling Based on Polygonal Meshes”
- <https://hal.inria.fr/inria-00186820/document>

