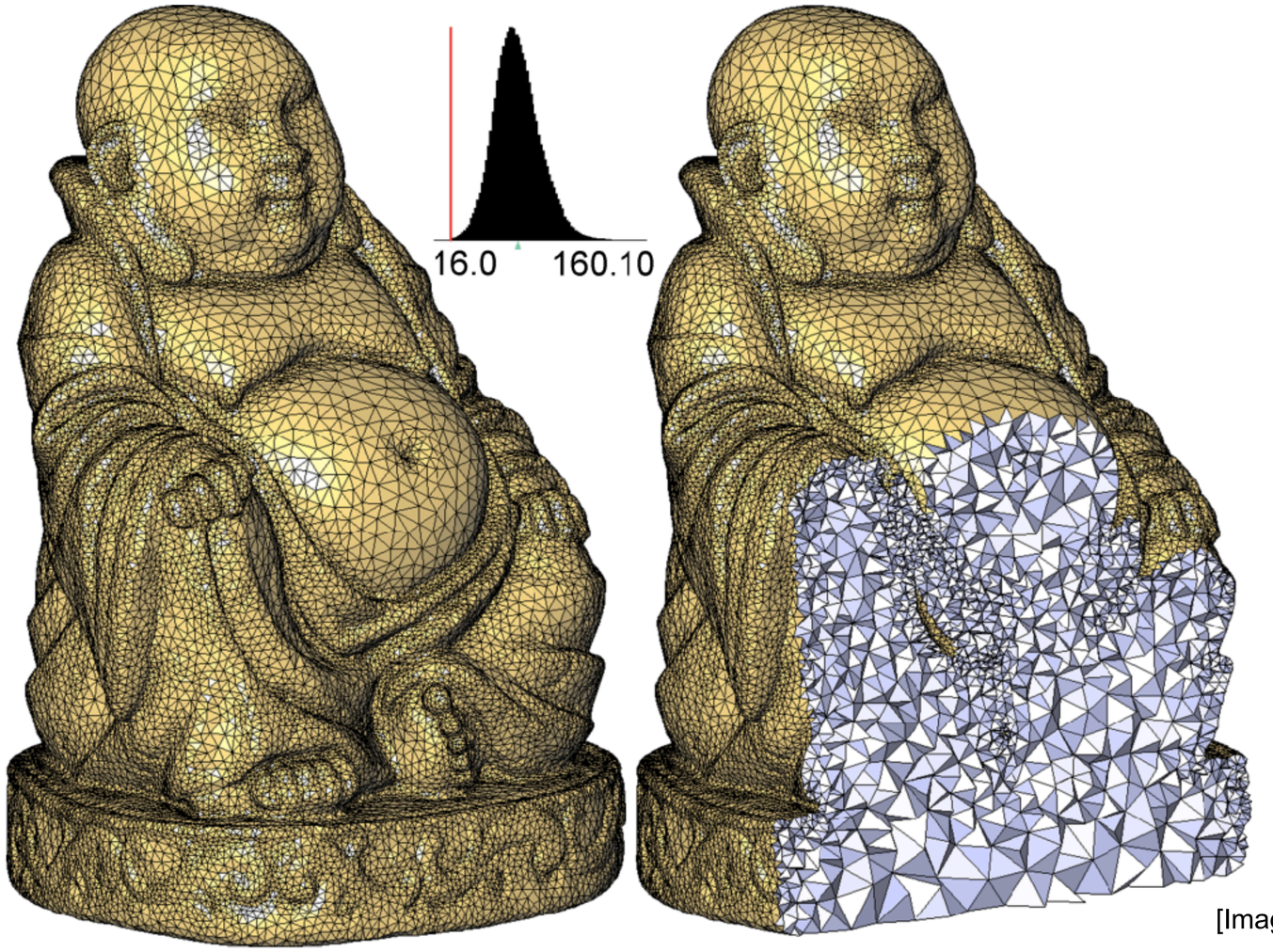
Winter 2022-23

Acquisition and Processing of 3D Geometry

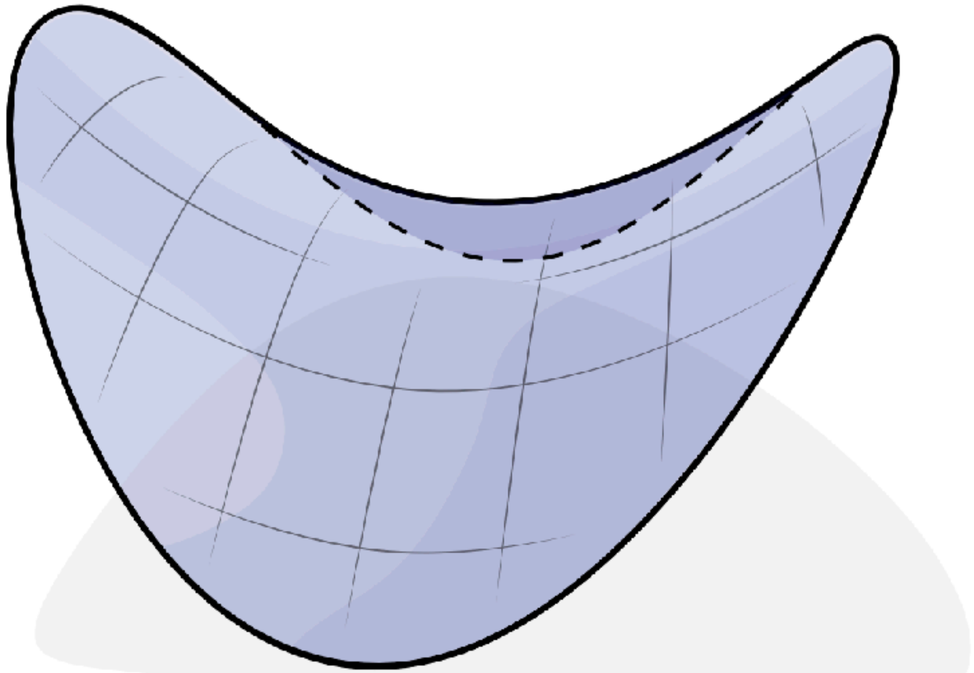
(COMP0119)

Niloy J. Mitra

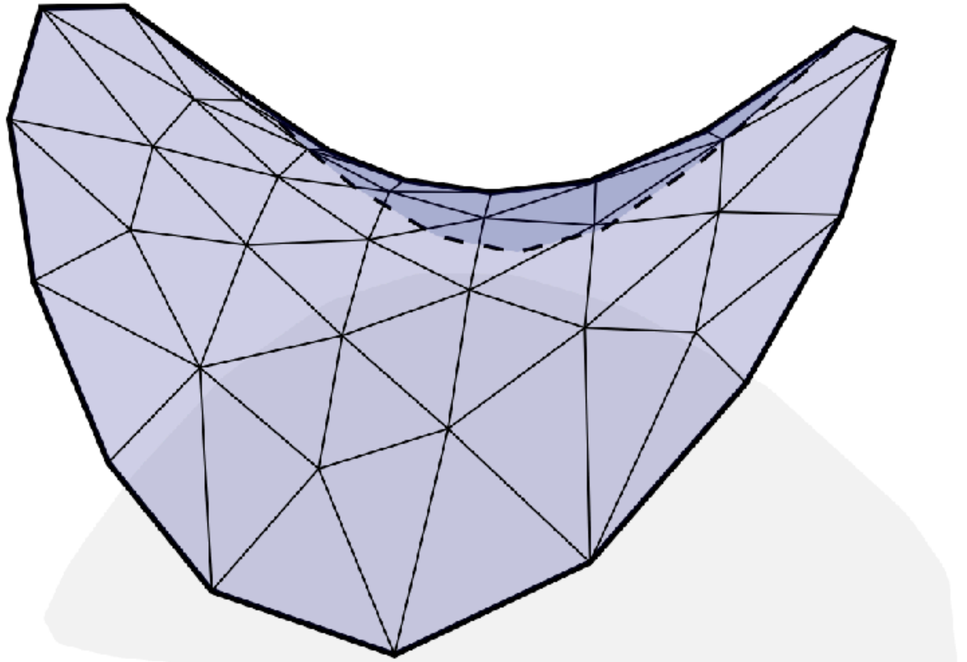


[Image source: Geometrica]

Continuous vs Discrete Representation



Continuous



Discrete

[Image from Keenan Crane]

Interactive Shape Modeling

Tools for design, editing and animation of digital shapes

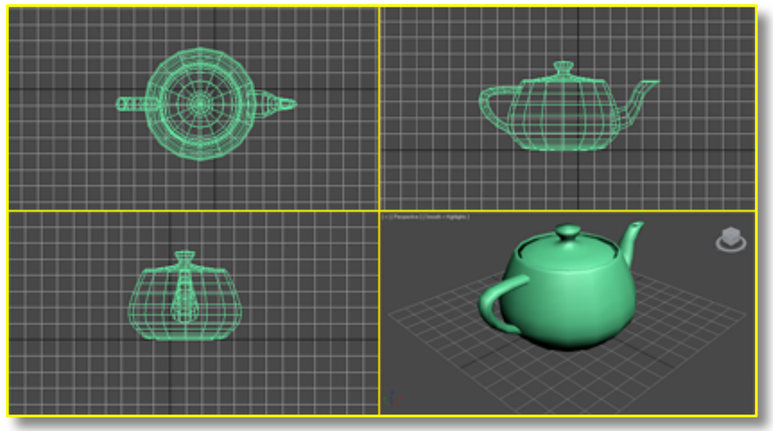
- **Interactive:** fast algorithms
- **Intuitive:** convenient interface + predictable outcome

Digital Shape Modeling

How do shapes find their way into computers?

- Geometric modeling is difficult

Humans have no direct “video out”

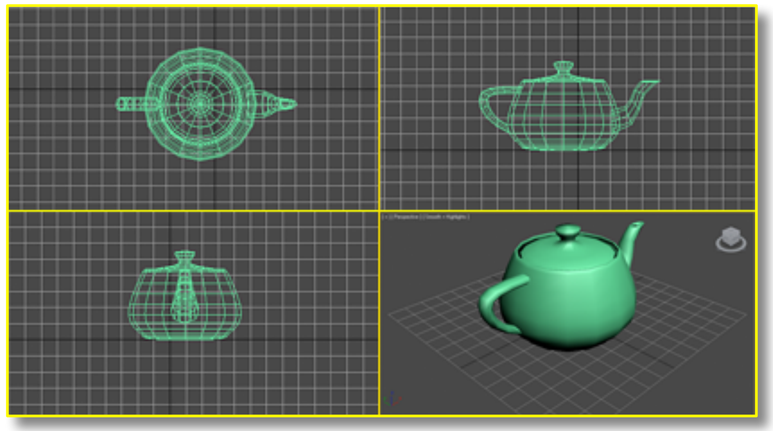


“Translation” from 2D to 3D is hard

Digital Shape Modeling

How do shapes find their way into computers?

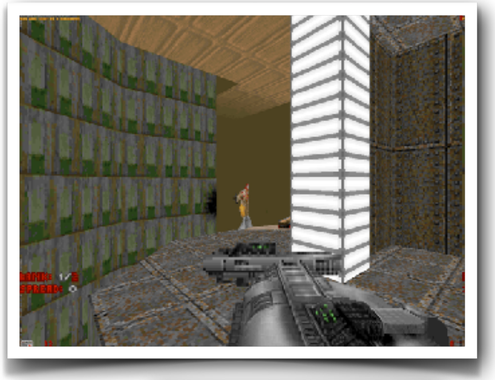
- Geometric modeling is difficult

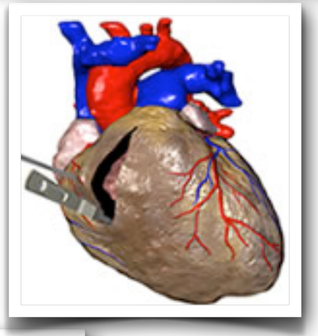
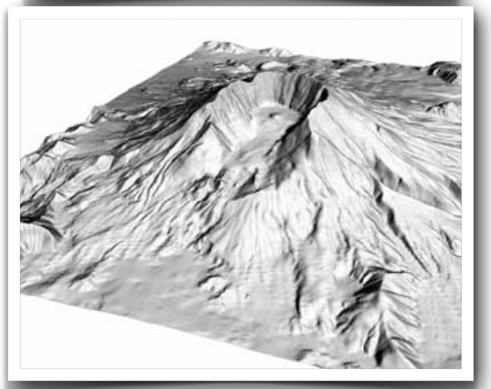
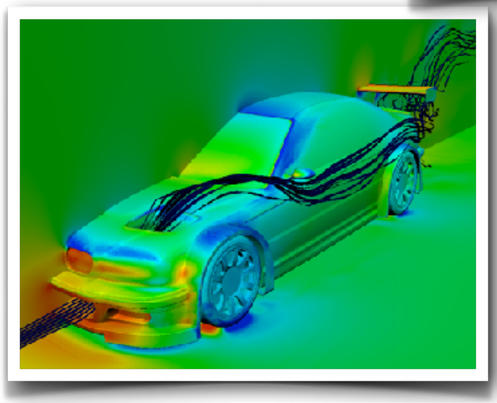


computation can compensate for lack of direct ability to convey visual information

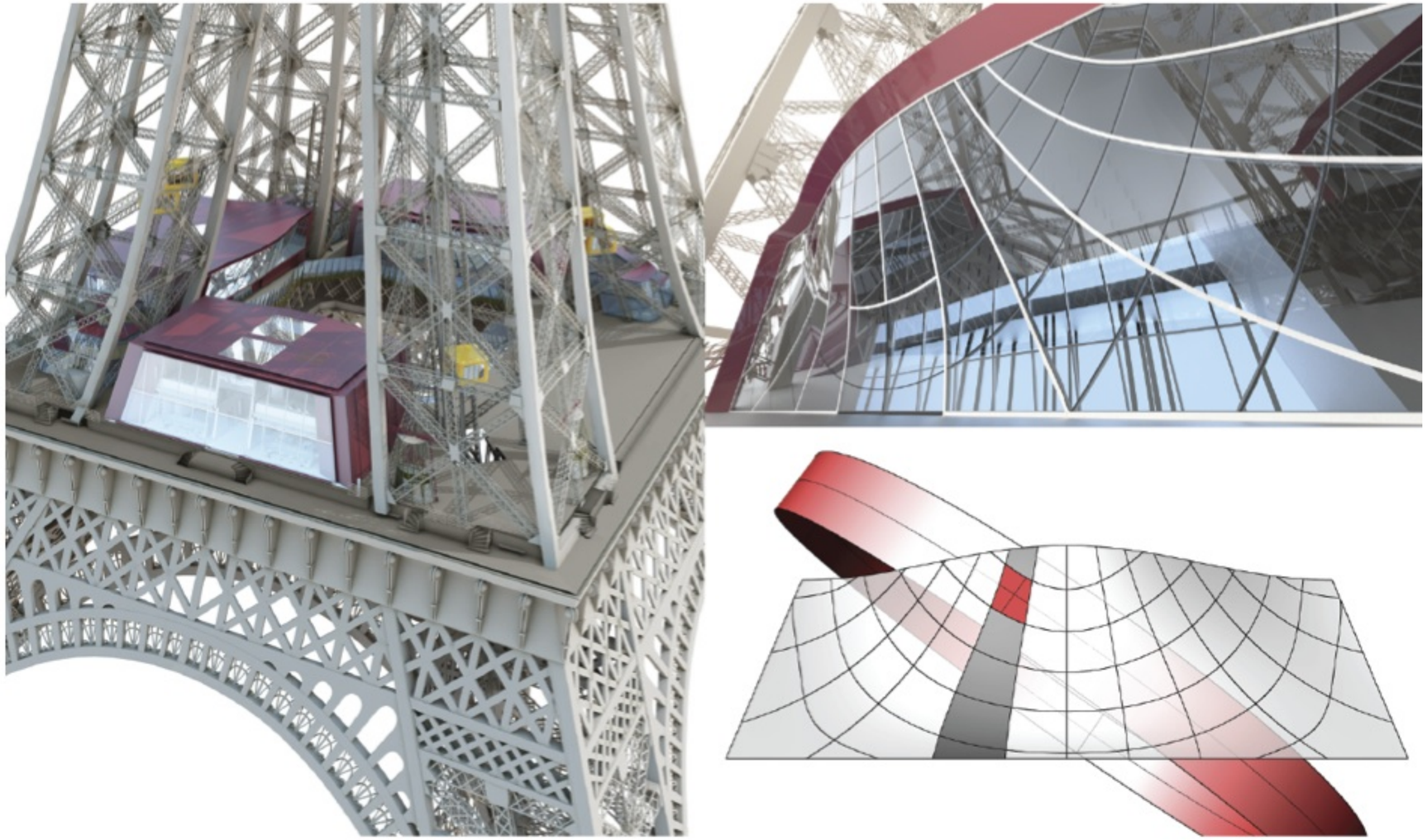
Application Areas

- Computer games
- Movie production
- Engineering
- Cultural Heritage
- Topography
- Architecture
- Medicine
- many more …



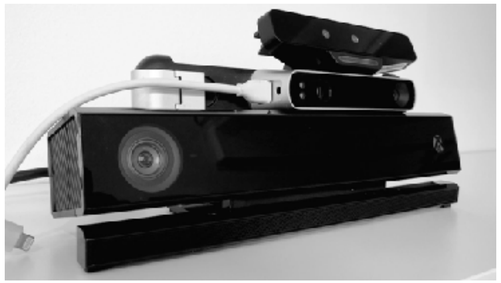





Form Finding (Collaboration w/ Evolute)



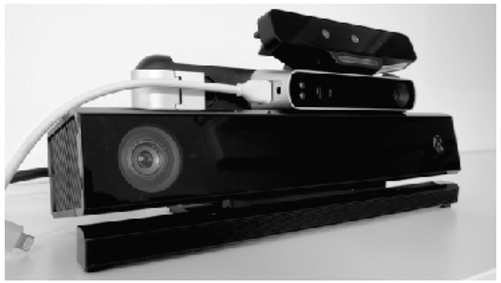
Digitizing Physical Models

Digitizing Physical Models



laser scanner

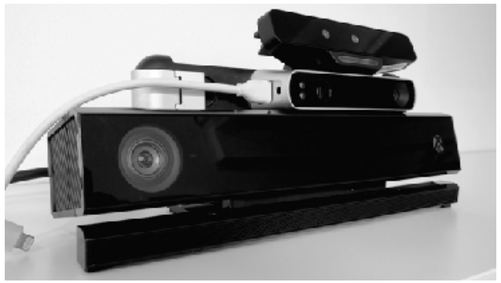
Digitizing Physical Models



laser scanner

3D geometry

Digitizing Physical Models



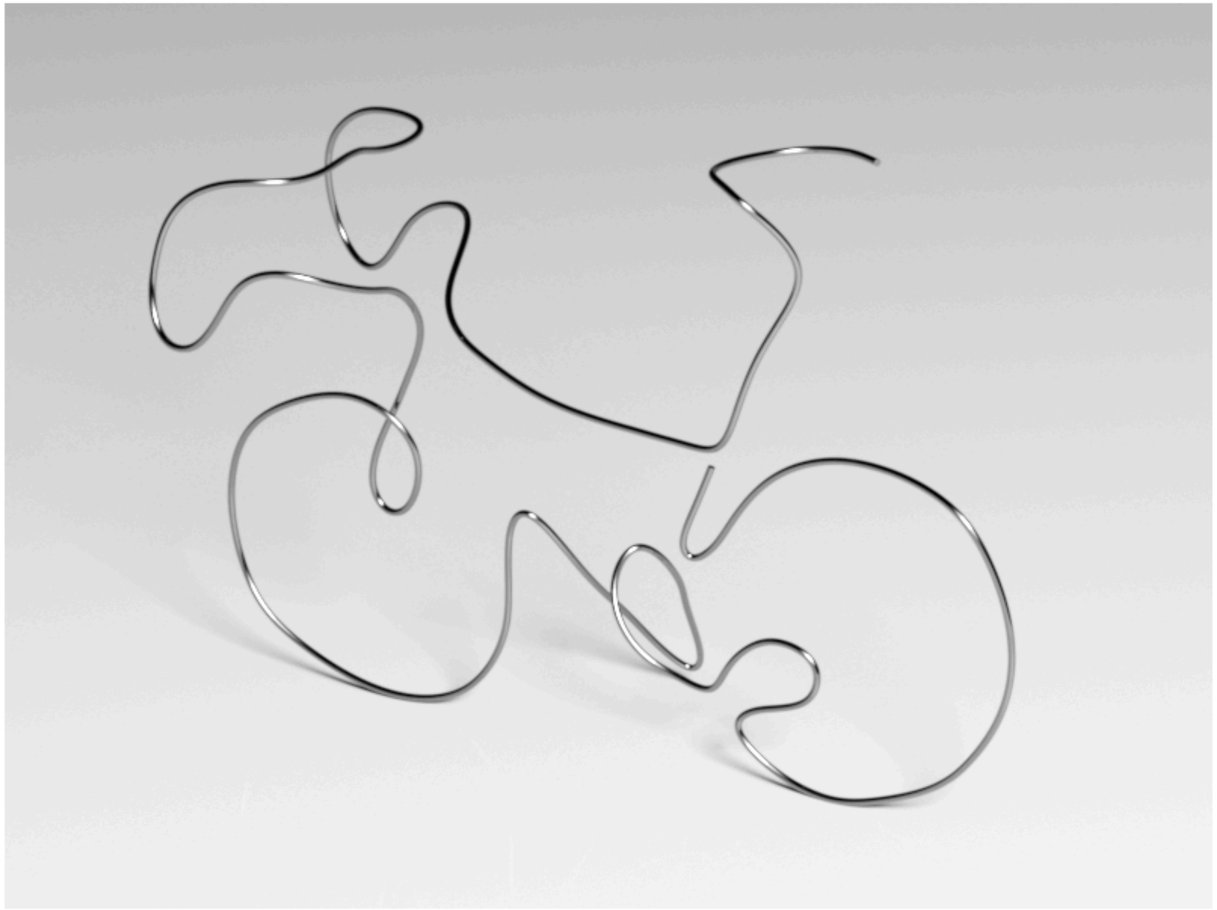
laser scanner

3D geometry

entertainment

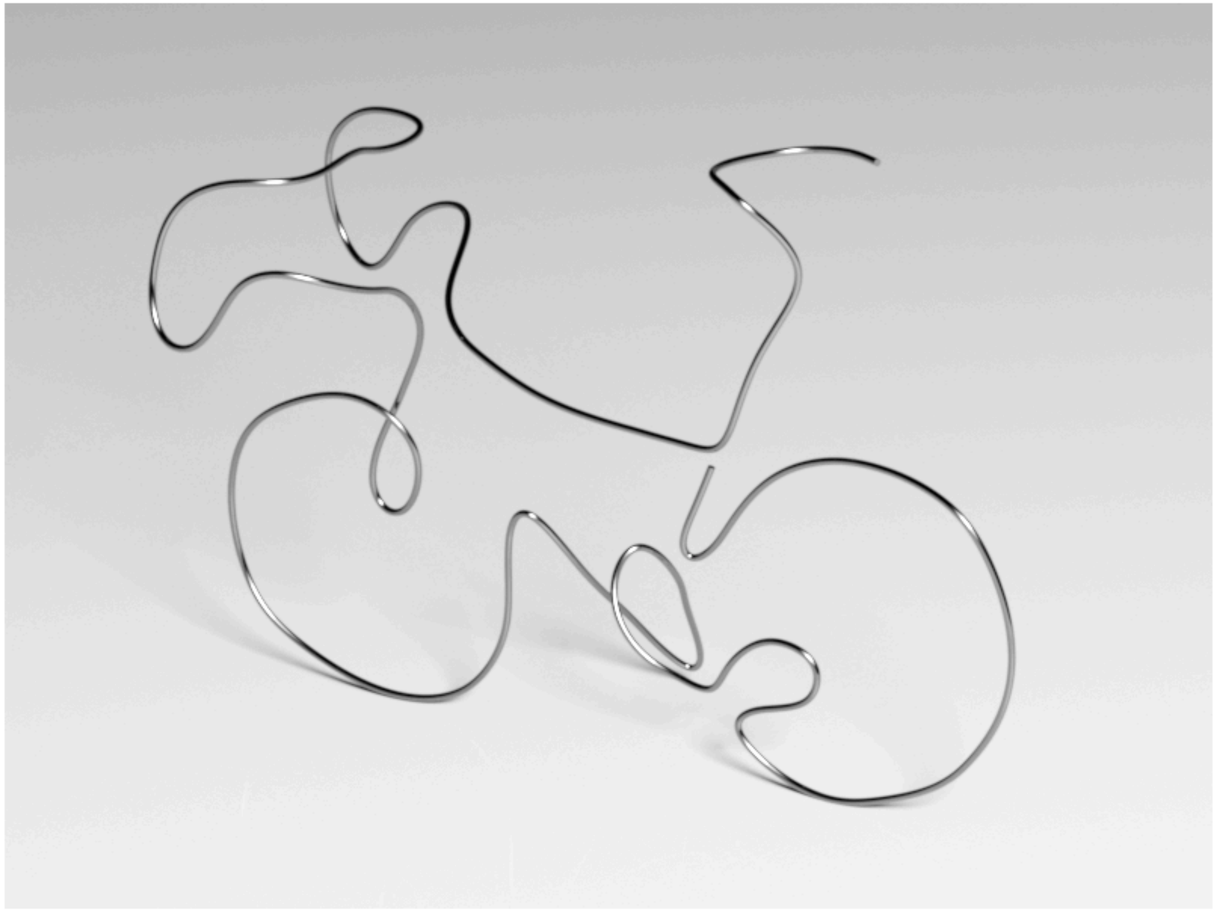
Scanning

BIKE



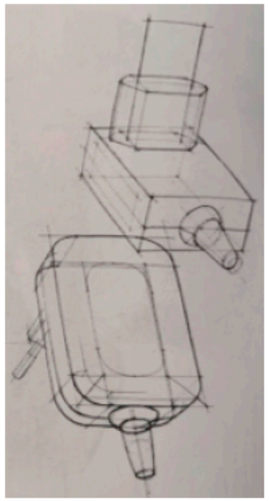
Scanning

BIKE



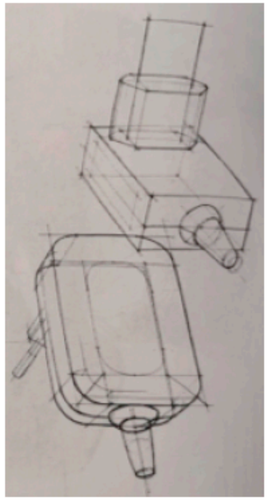
2D to 3D ‘Lifting’

(a) inspirational sketch

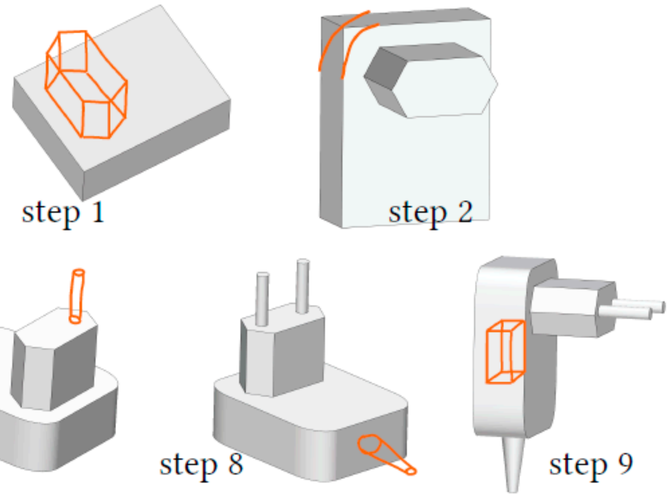


2D to 3D ‘Lifting’

(a) inspirational sketch

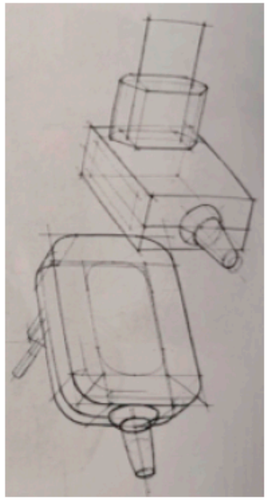


(b) sketching sequence using Sketch2CAD

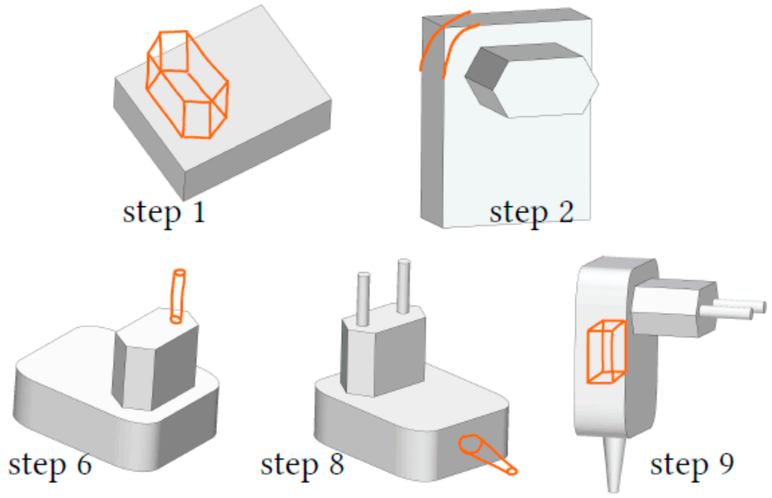


2D to 3D ‘Lifting’

(a) inspirational sketch



(b) sketching sequence using Sketch2CAD

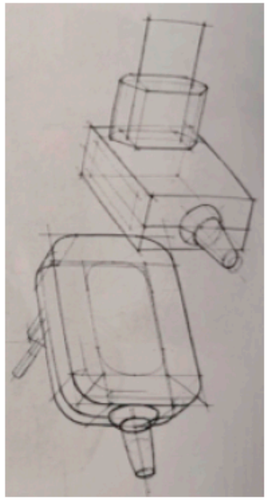


(c) inferred CAD instructions

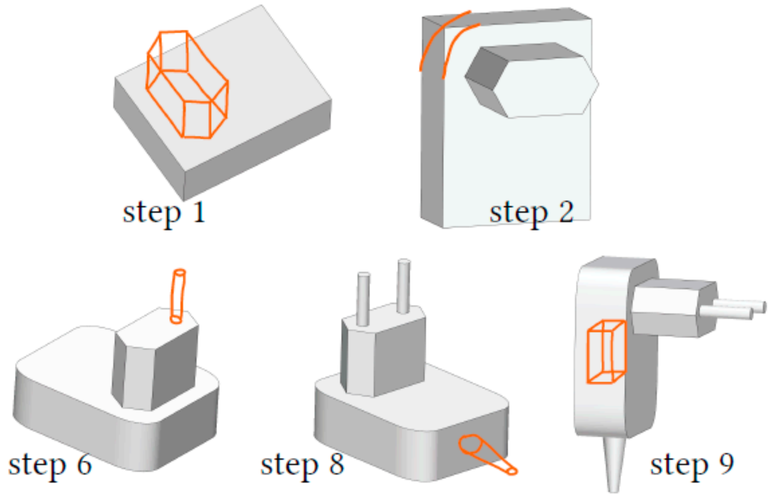
`AddPolyhedron`: < plane 3, length 0.25 >
`BevelCorner`: < plane 3, corner 3 >
`BevelCorner`: < plane 3, corner 2 >
`BevelCorner`: < plane 3, corner 0 >
`BevelCorner`: < plane 3, corner 1 >
`AddSweepShape`: < plane 4, length 0.20 >
`AddSweepShape`: < plane 4, length 0.20 >
`AddSweepShape`: < plane 1, length 0.23 >
`SubtractPolyhedron`: < plane 12, length 0.03 >
`SubtractPolyhedron`: < plane 13, length 0.03 >

2D to 3D ‘Lifting’

(a) inspirational sketch



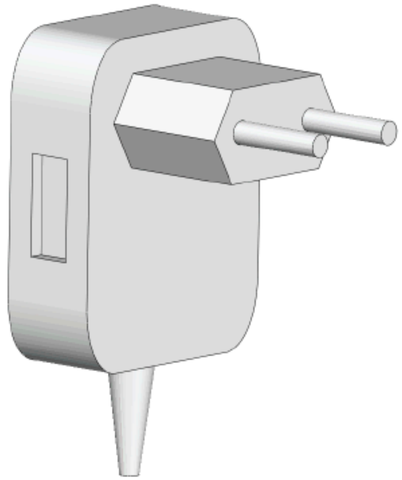
(b) sketching sequence using Sketch2CAD



(c) inferred CAD instructions

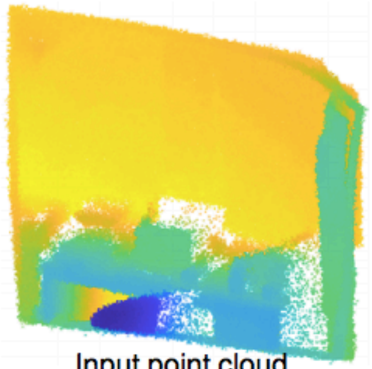
`AddPolyhedron`: < plane 3, length 0.25 >
`BevelCorner`: < plane 3, corner 3 >
`BevelCorner`: < plane 3, corner 2 >
`BevelCorner`: < plane 3, corner 0 >
`BevelCorner`: < plane 3, corner 1 >
`AddSweepShape`: < plane 4, length 0.20 >
`AddSweepShape`: < plane 4, length 0.20 >
`AddSweepShape`: < plane 1, length 0.23 >
`SubtractPolyhedron`: < plane 12, length 0.03 >
`SubtractPolyhedron`: < plane 13, length 0.03 >

(d) CAD model

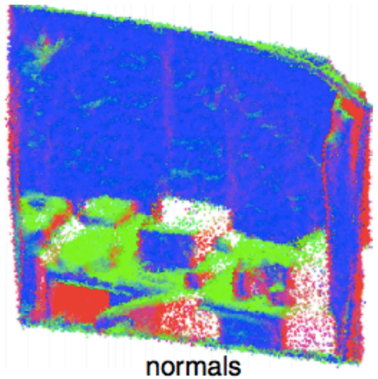


Estimating Local Properties

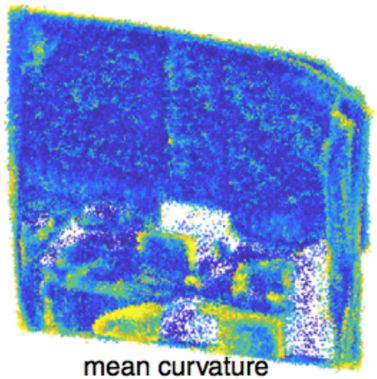
RGBD Image



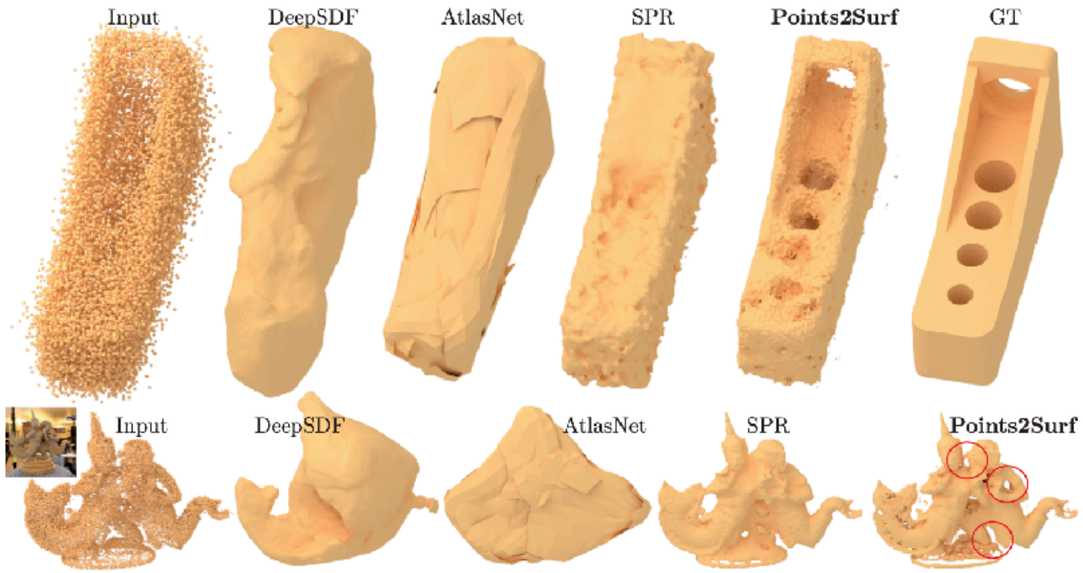
Input point cloud



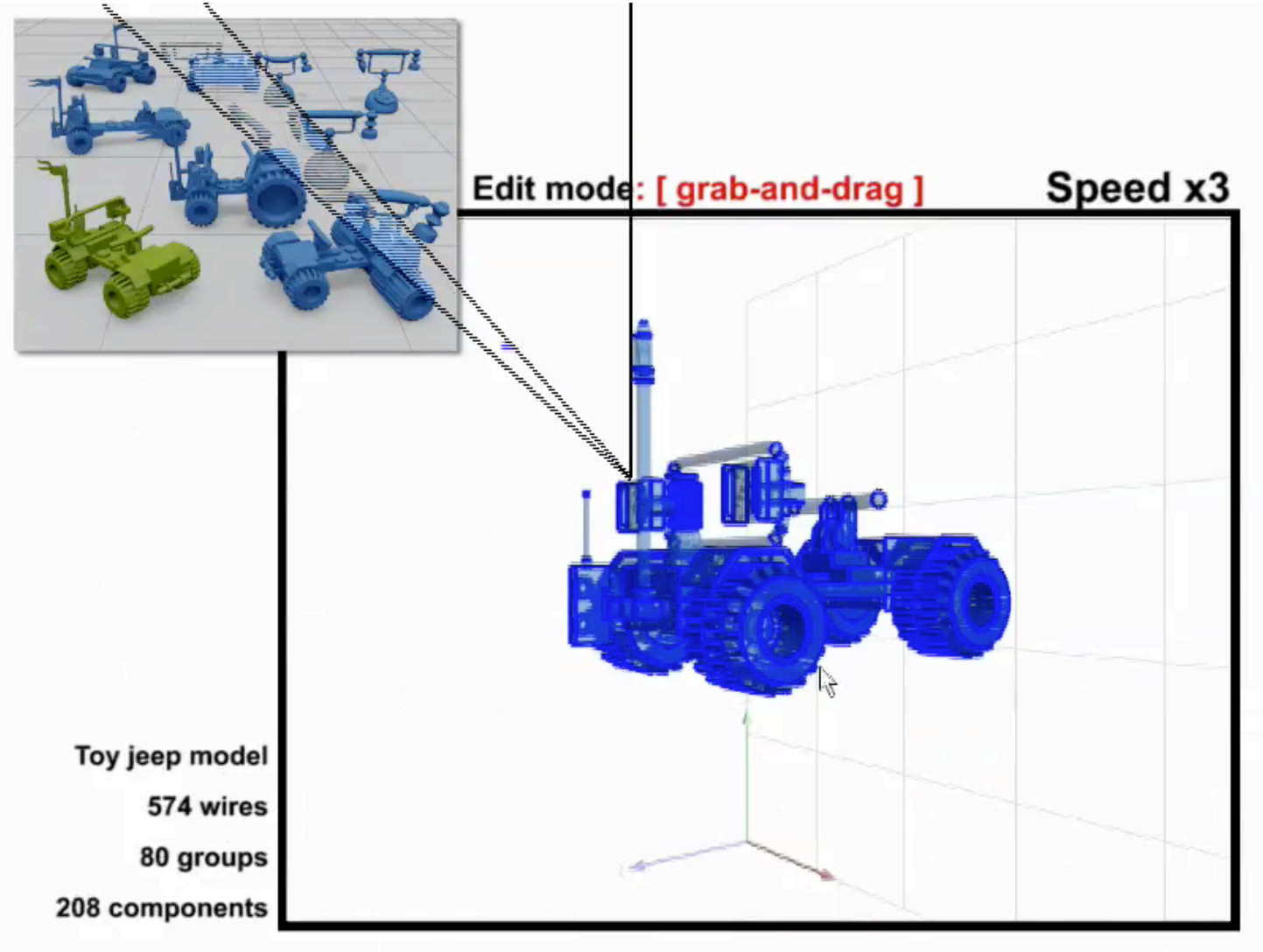
normals



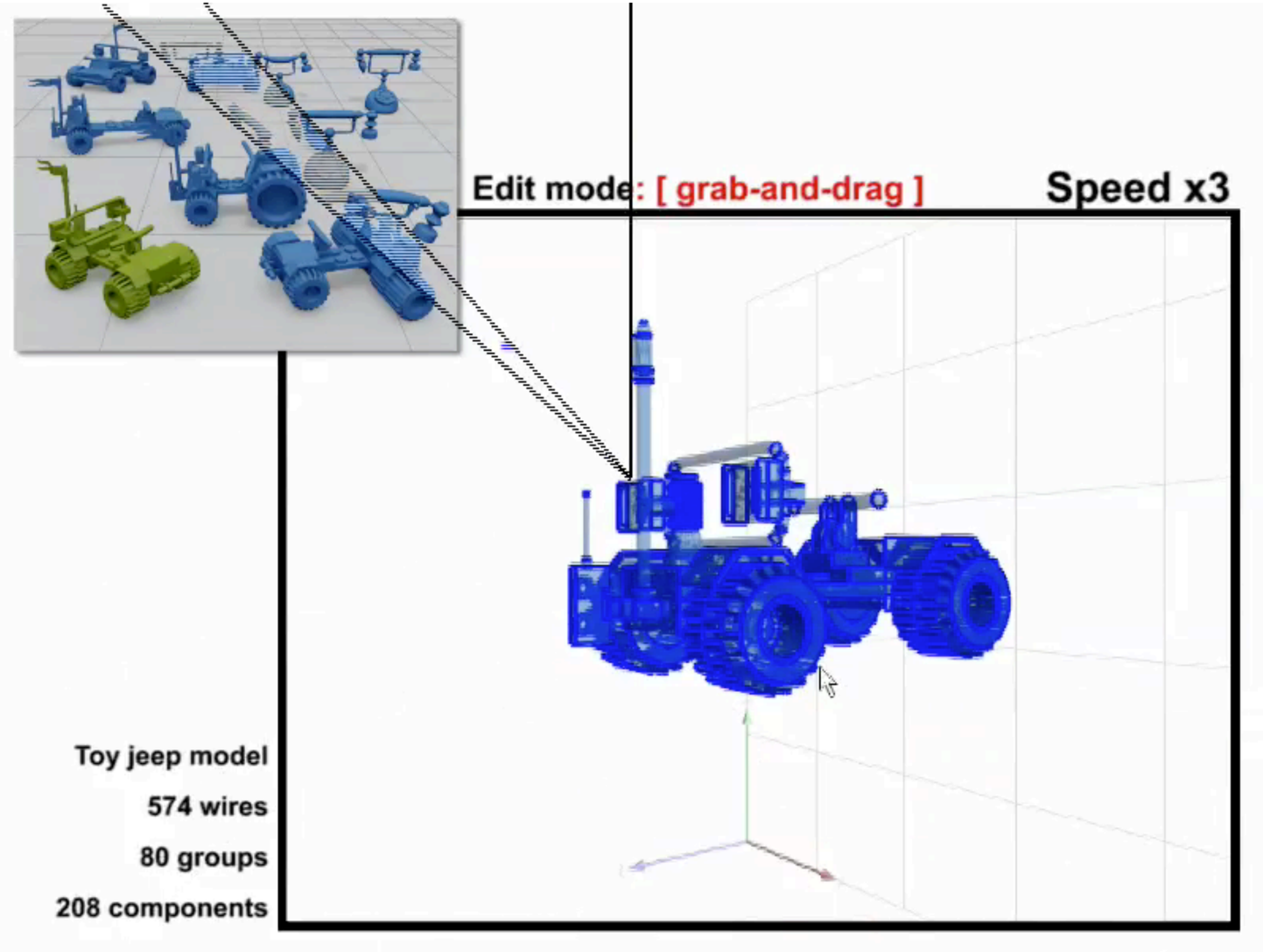
mean curvature



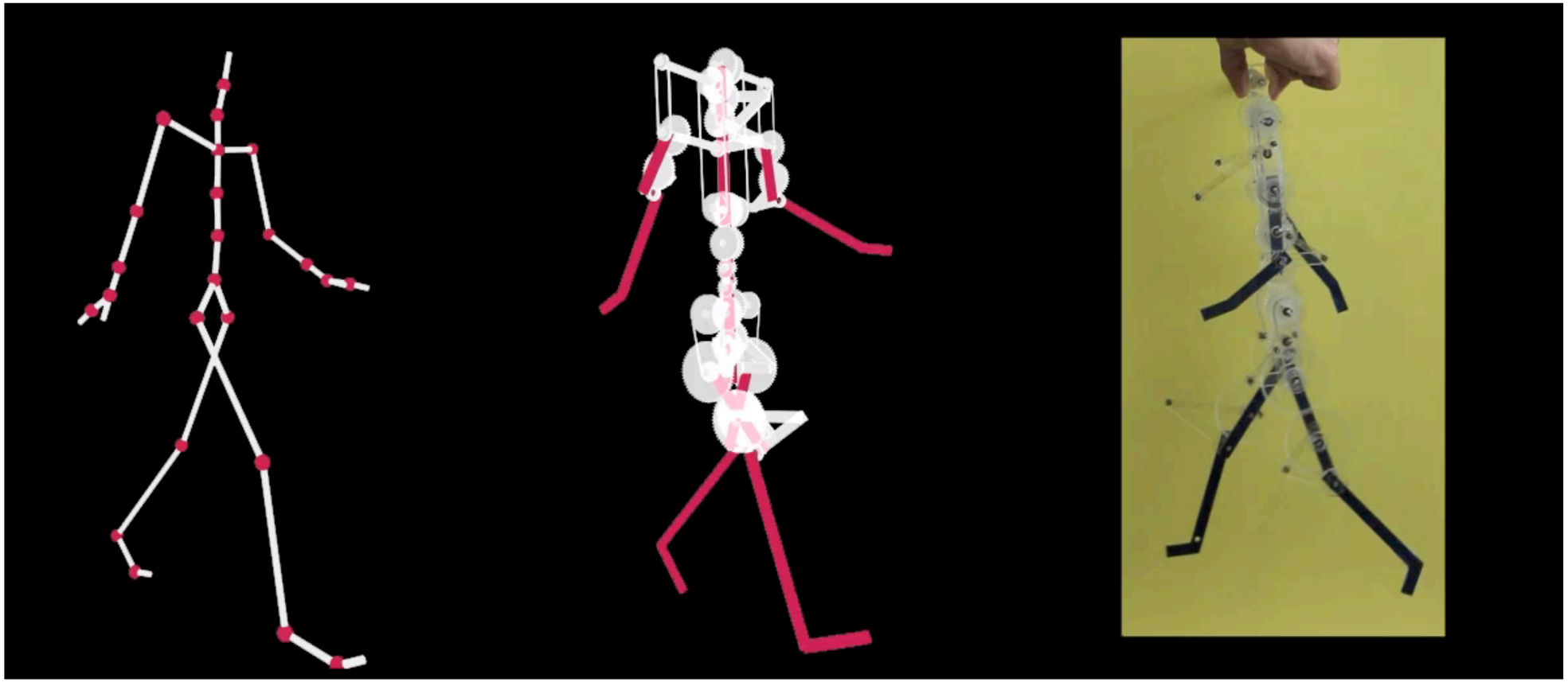
Mesh Manipulation



Mesh Manipulation

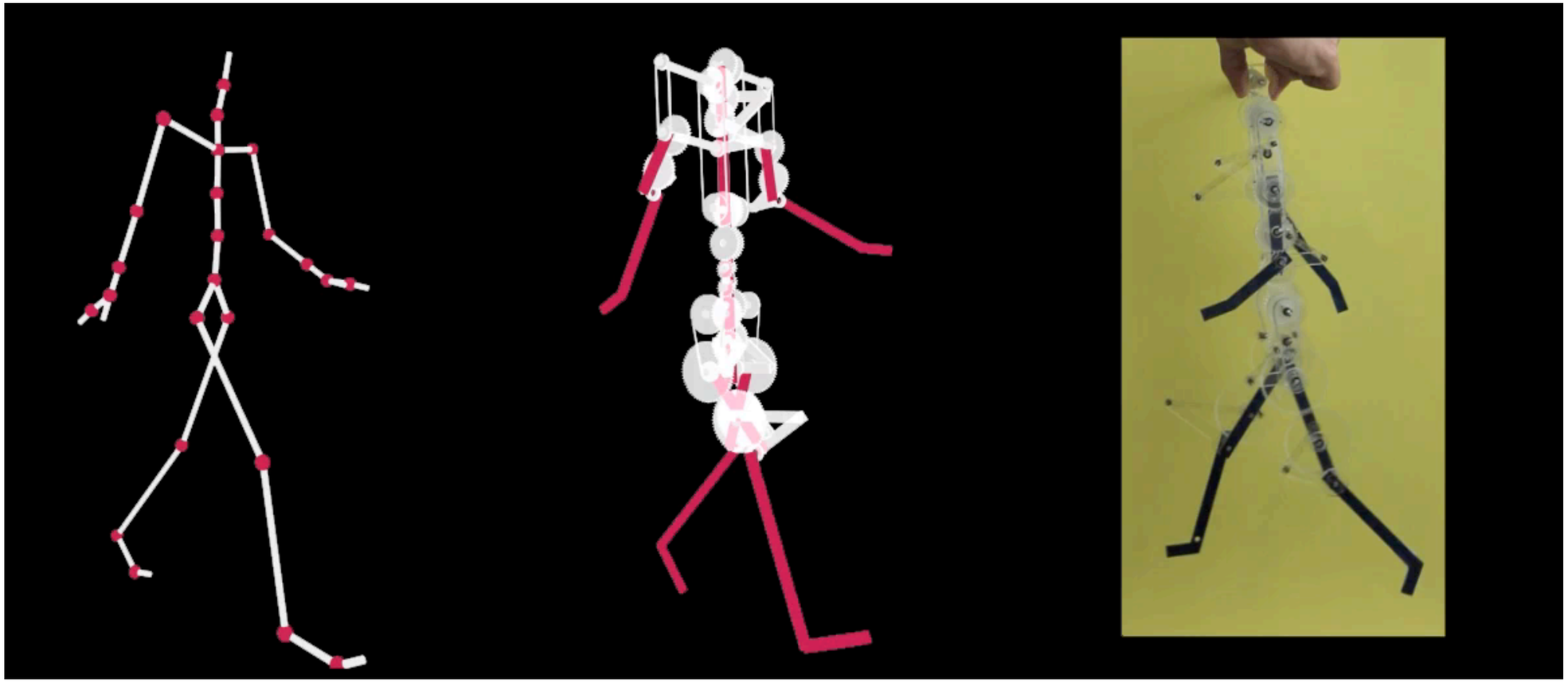


Motion to Geometry



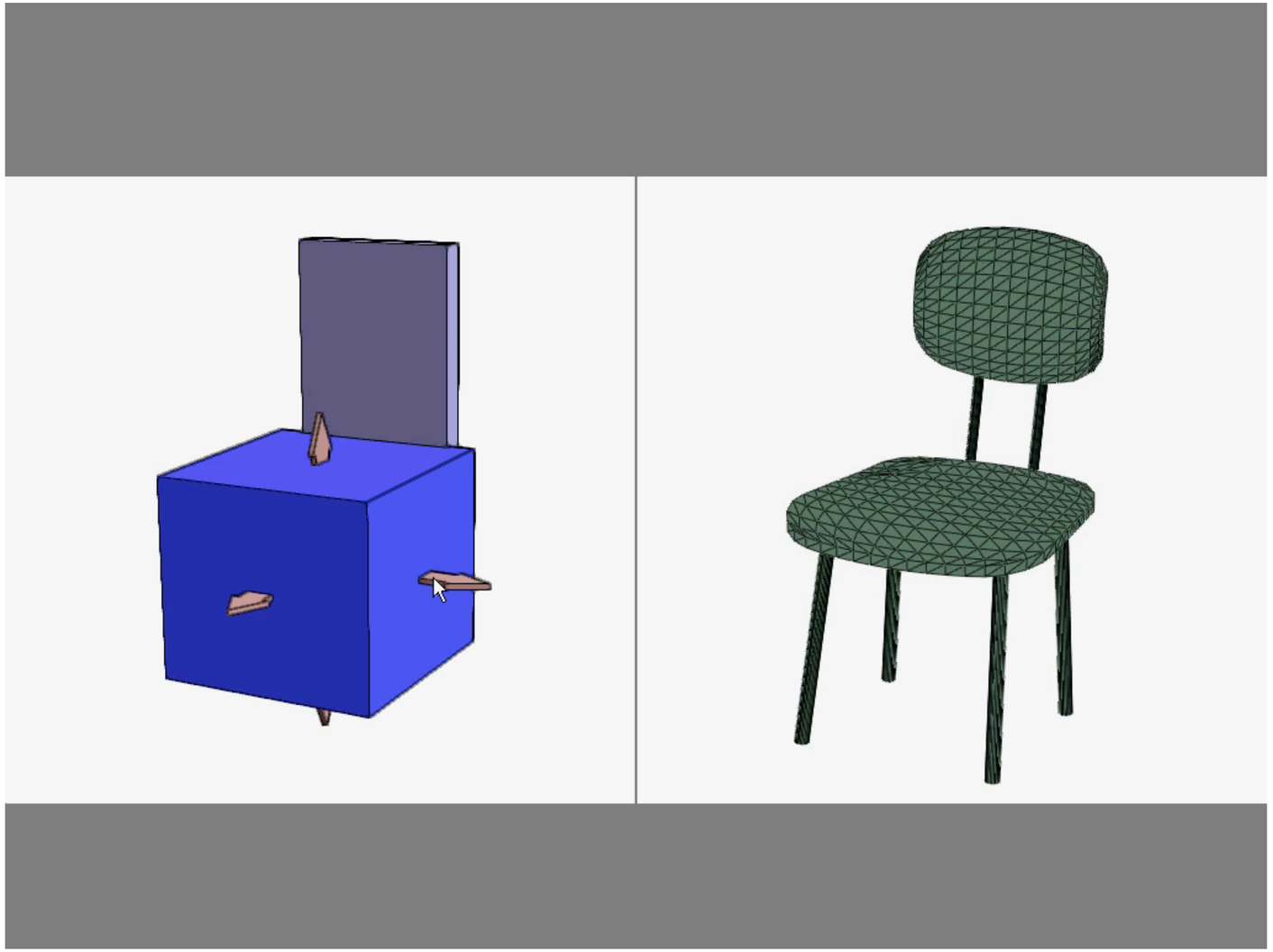
Input Motion | Generated Automaton | Physical Prototype

Motion to Geometry

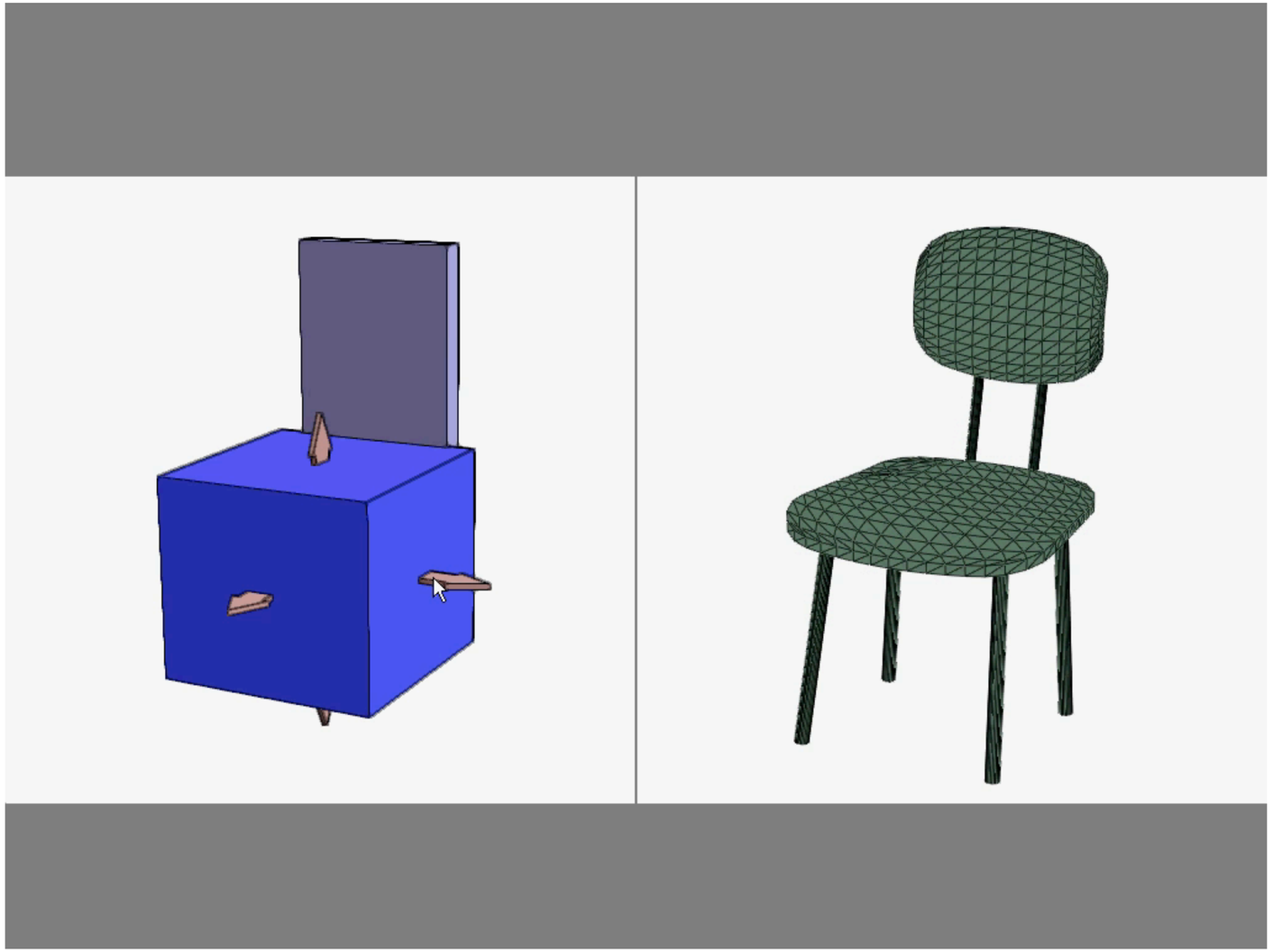


Input Motion | Generated Automaton | Physical Prototype

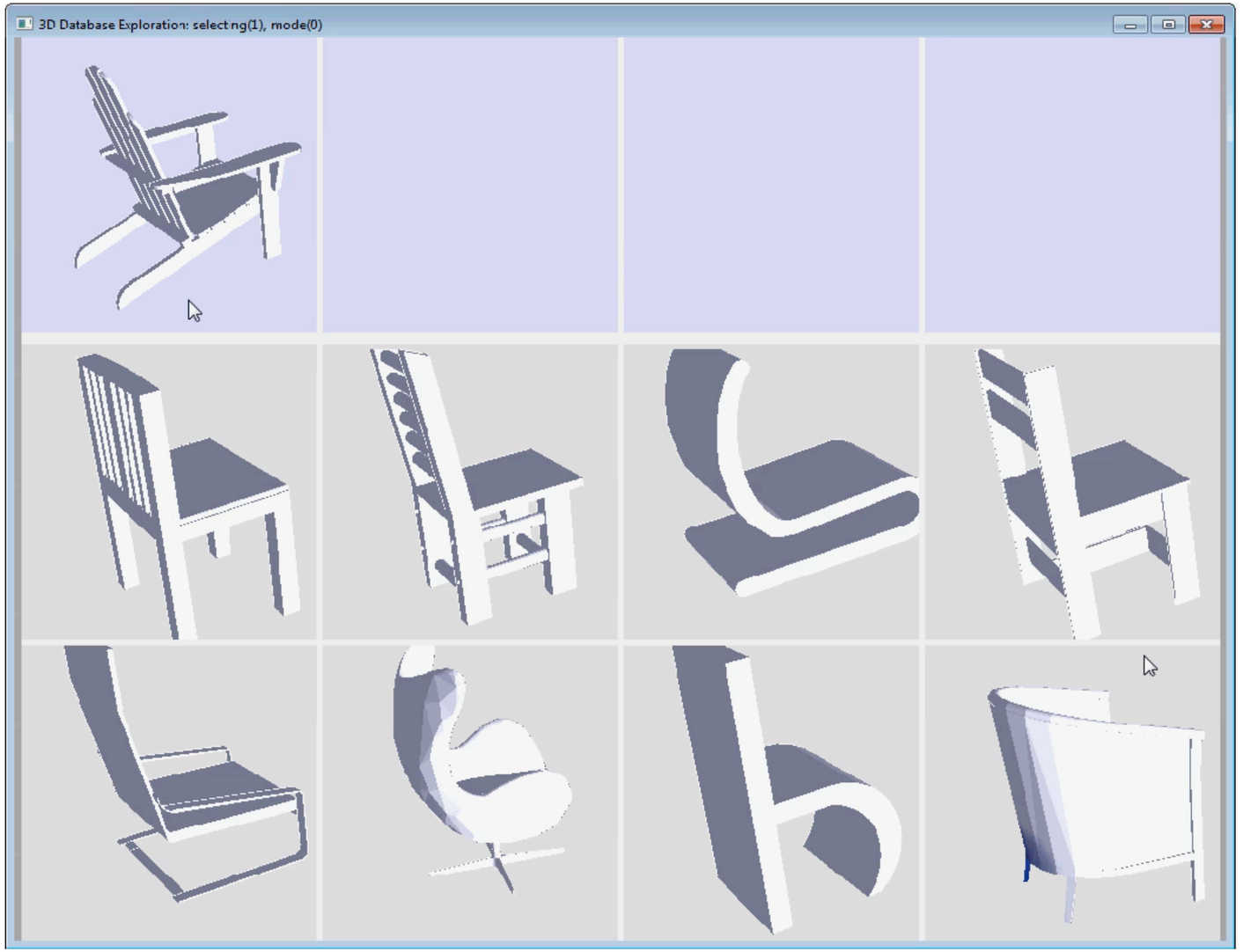
Exploring Data Collections



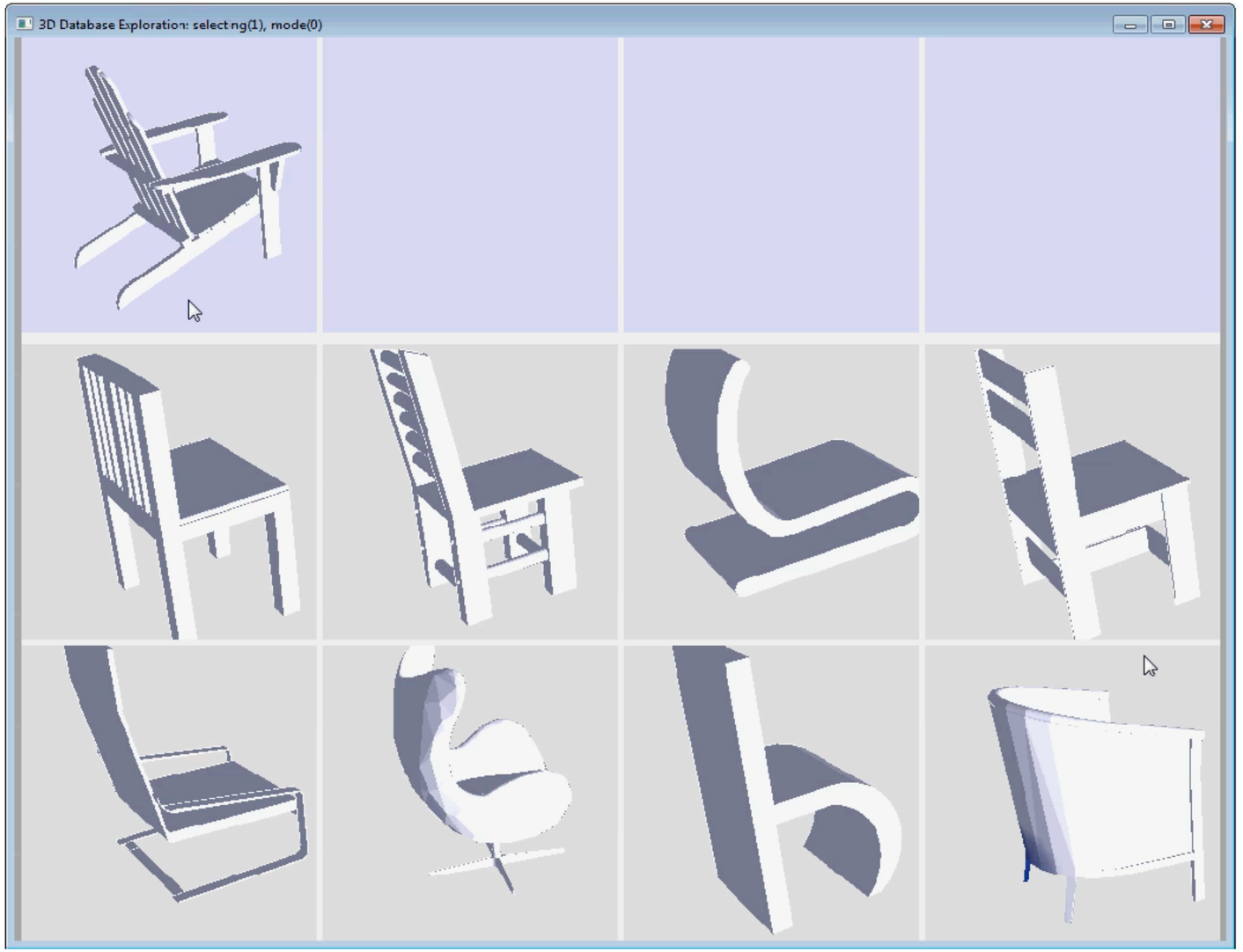
Exploring Data Collections



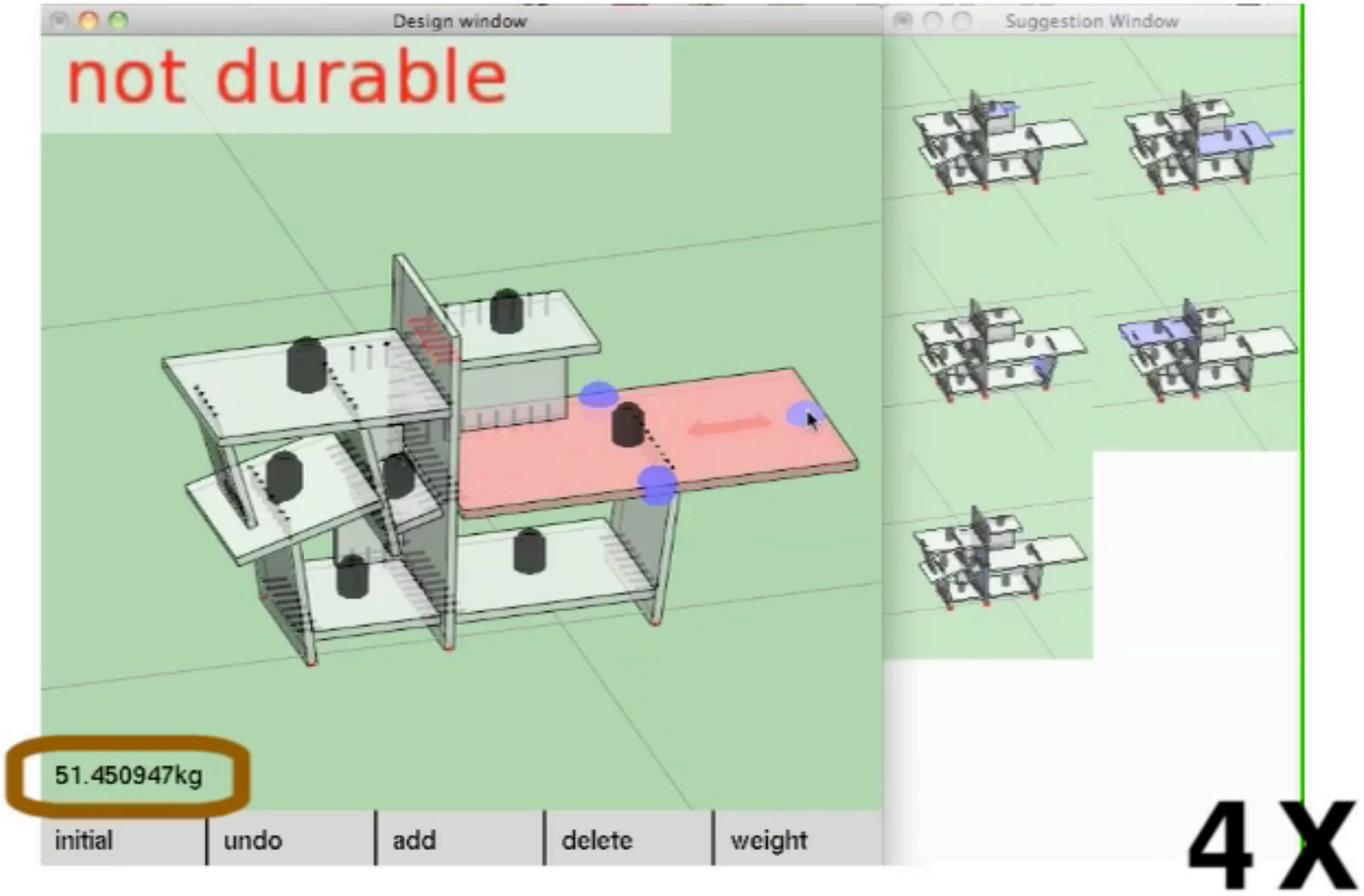
Exploring Model Collections



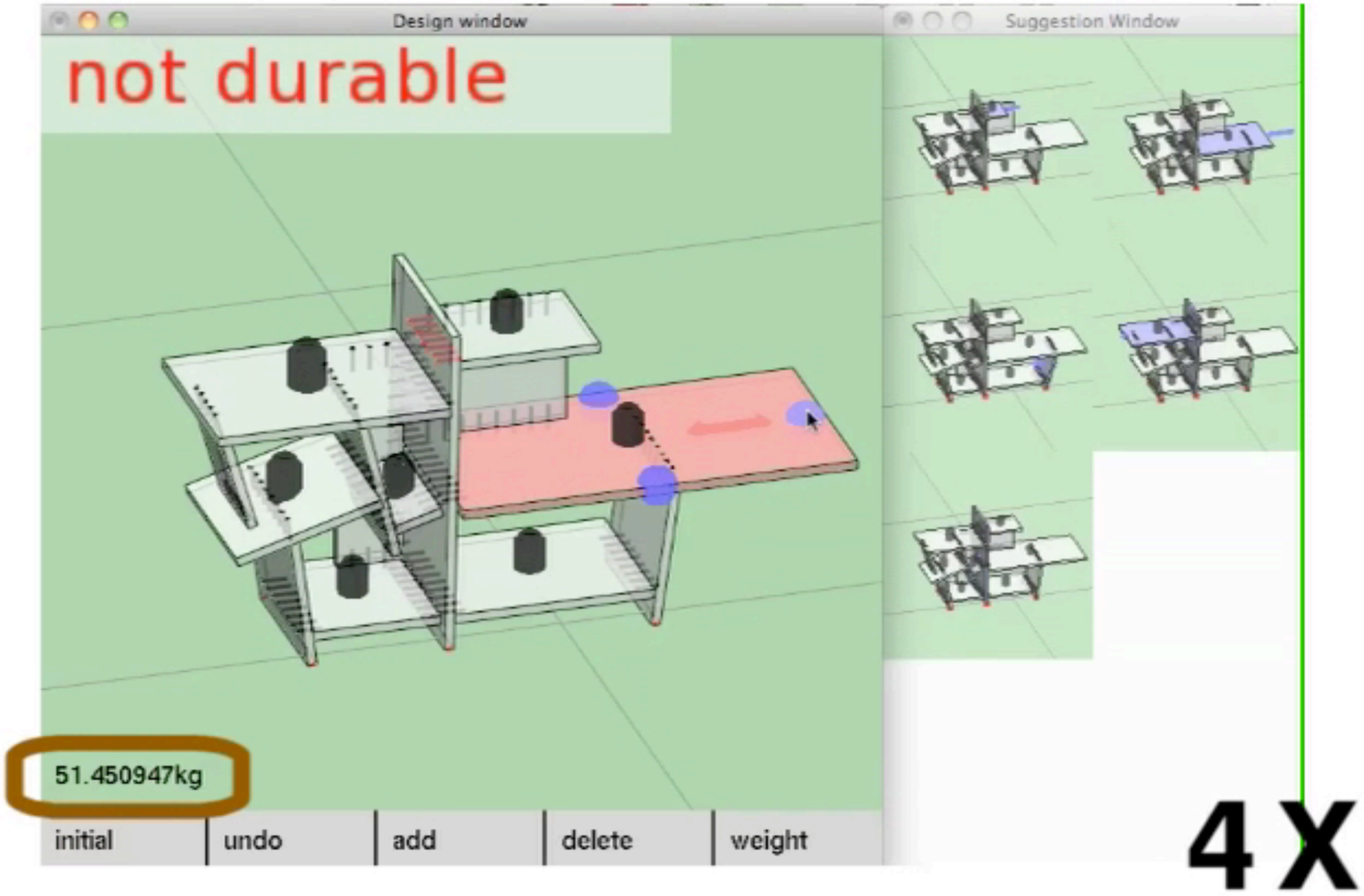
Exploring Model Collections



“Forms as Force Diagrams”

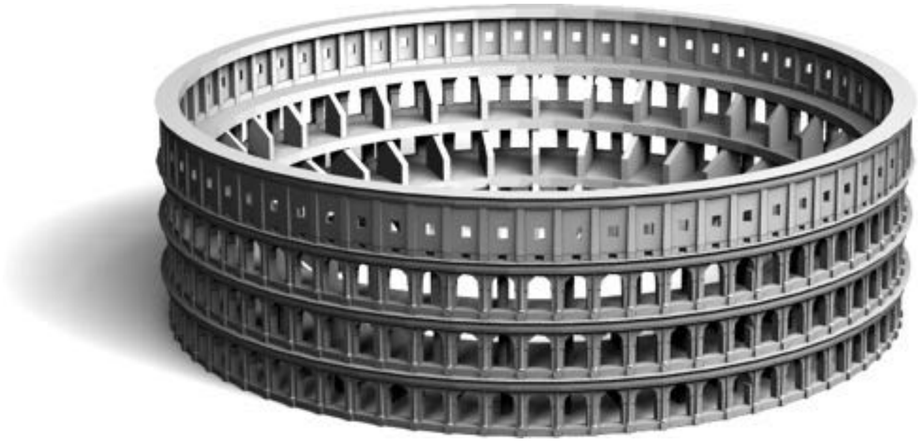


“Forms as Force Diagrams”



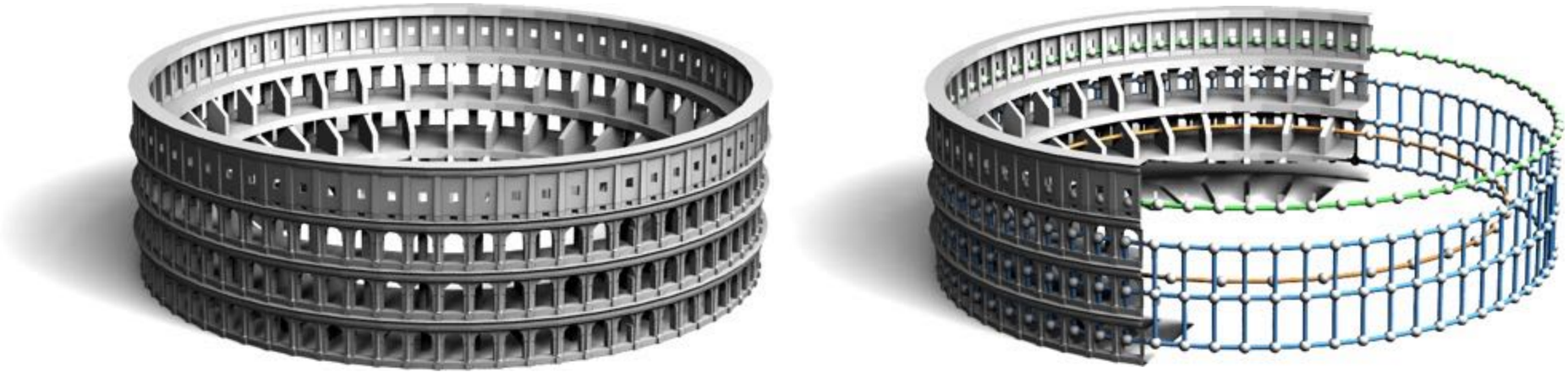
Detecting Regularities

low-level geometry → (structure+element) + variations



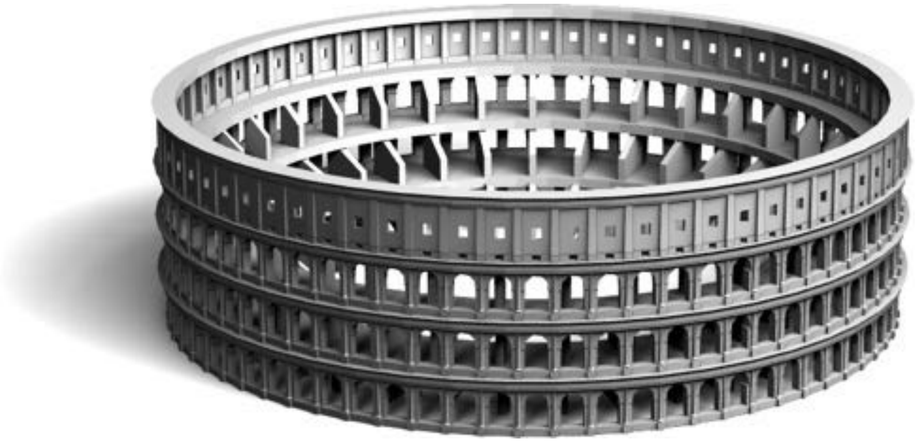
Detecting Regularities

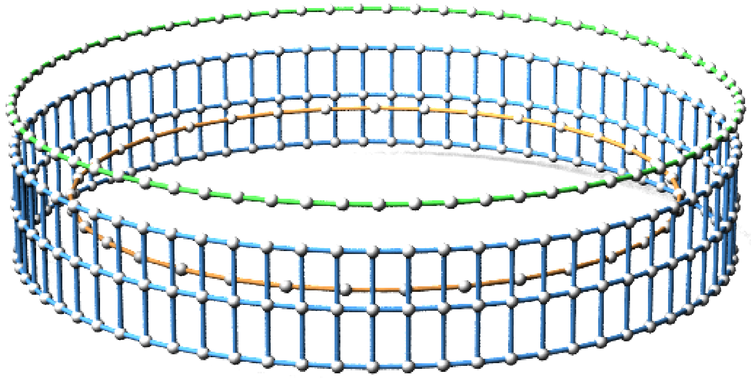
low-level geometry → (structure+element) + variations



Detecting Regularities

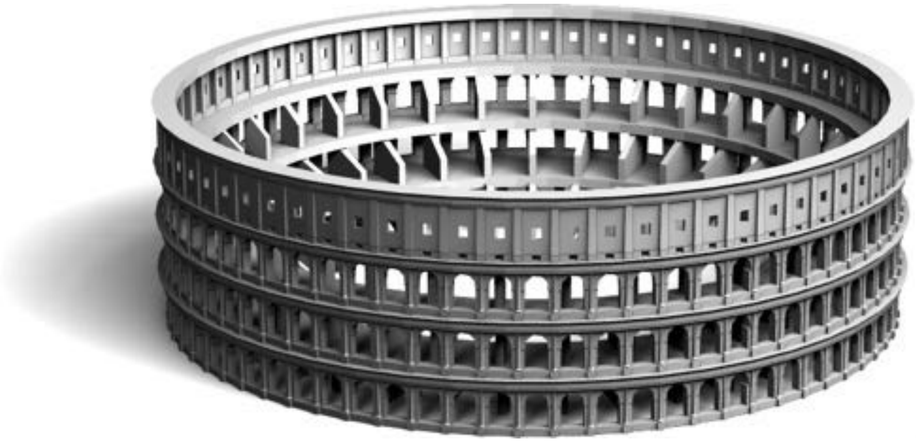
low-level geometry → (structure+element) + variations

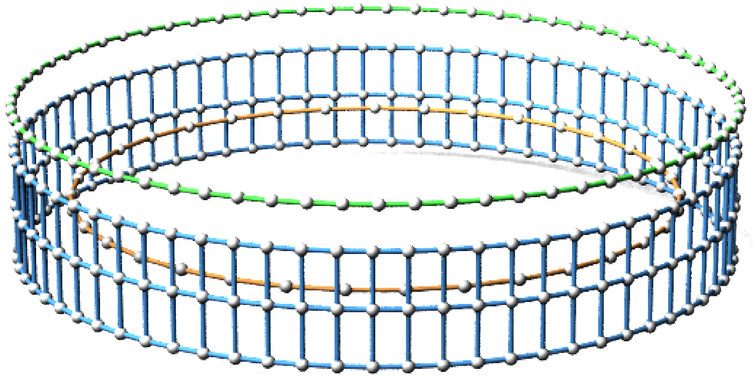




Detecting Regularities

low-level geometry → (structure+element) + variations



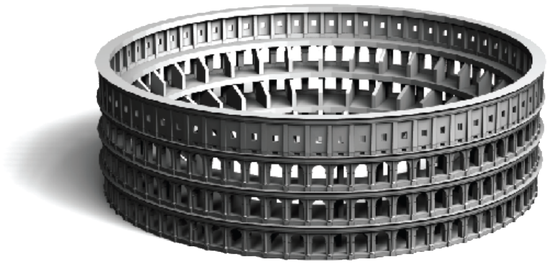


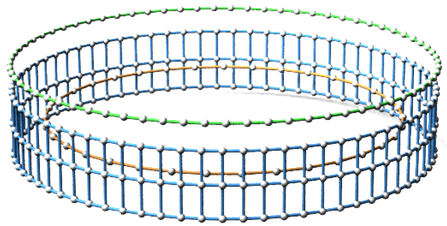
Rot: 72

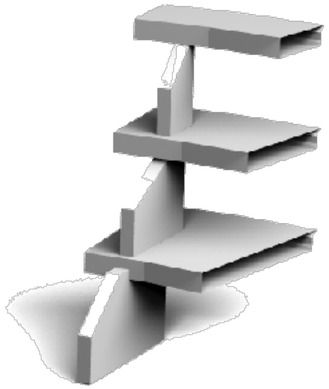
Rot x Trans: 72 x 3

Rot: 35

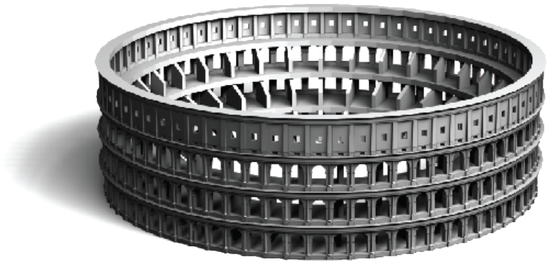
Find and Replace

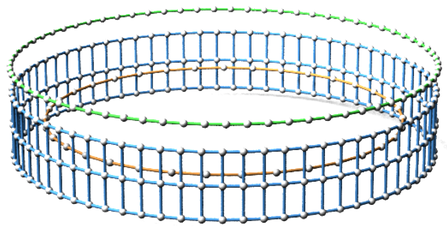


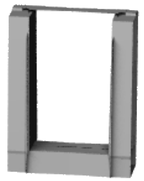


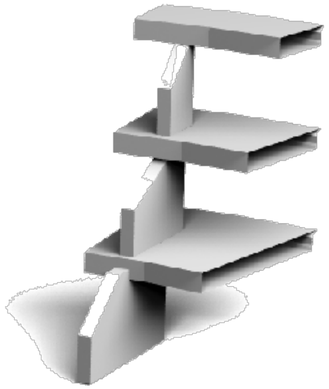


Find and Replace

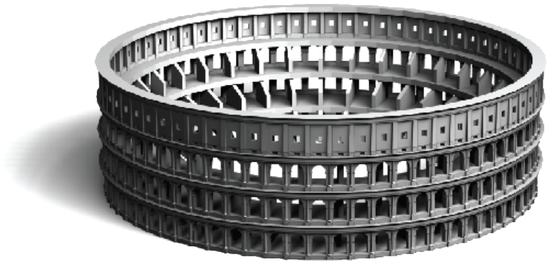


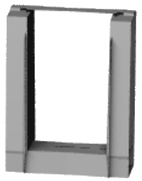


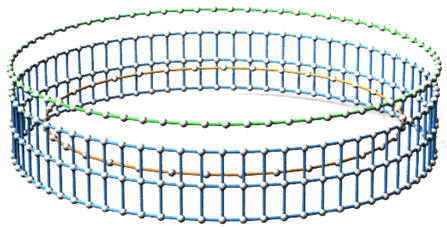


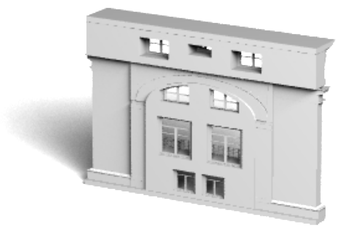


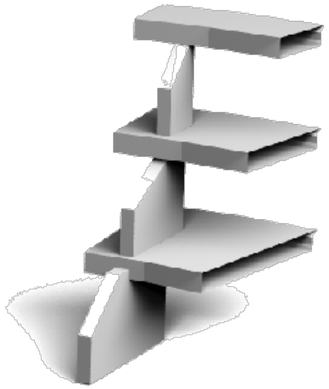
Find and Replace



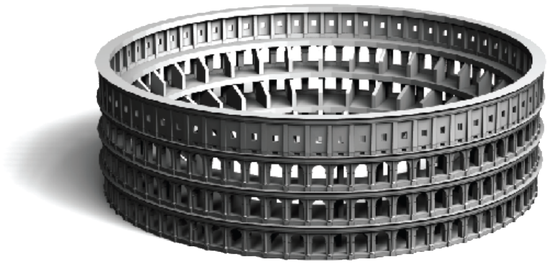


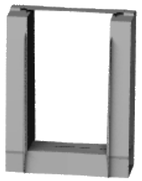


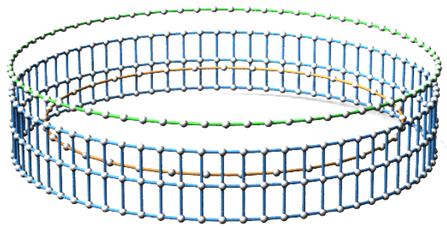


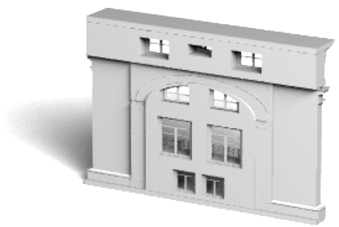


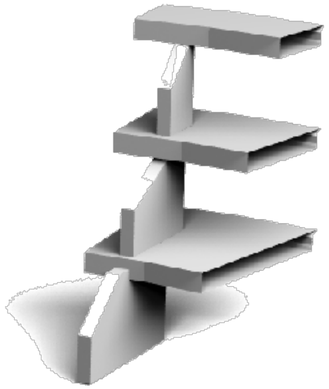
Find and Replace



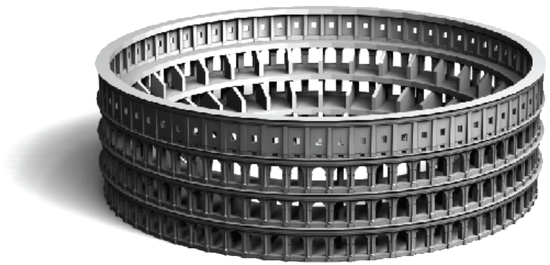


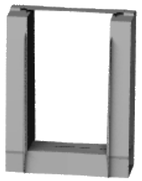


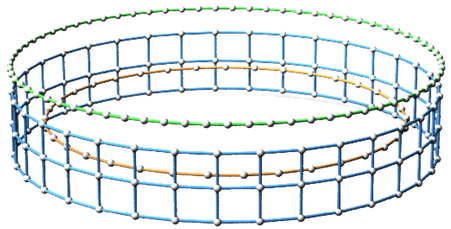


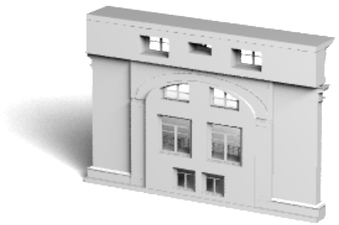


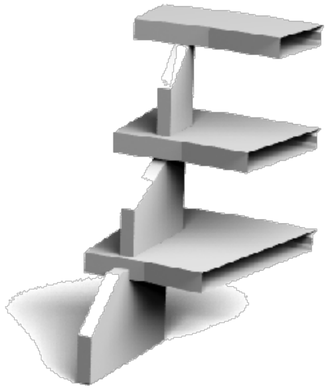
Find and Replace



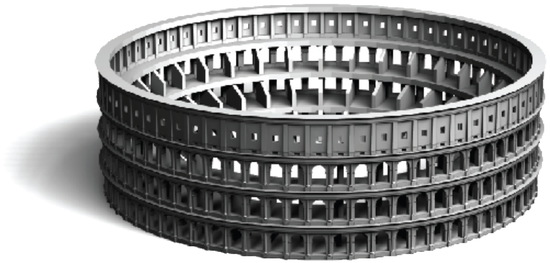


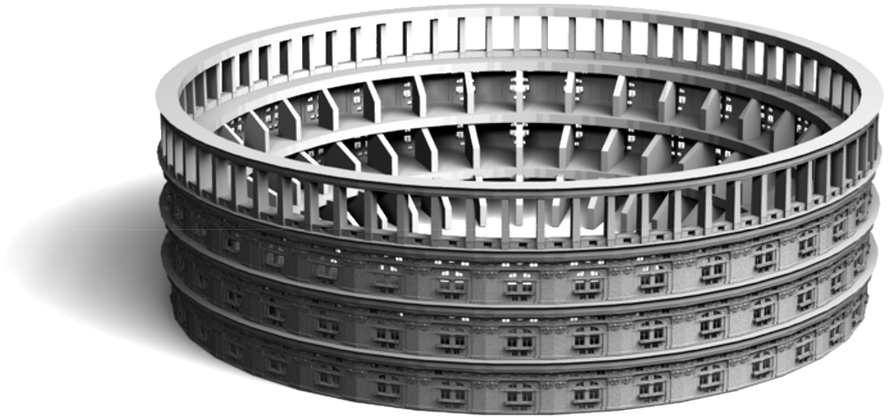


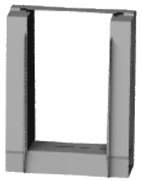


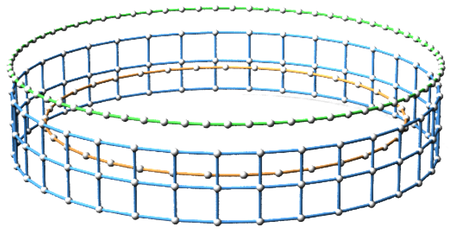


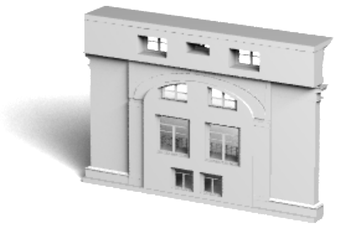
Find and Replace

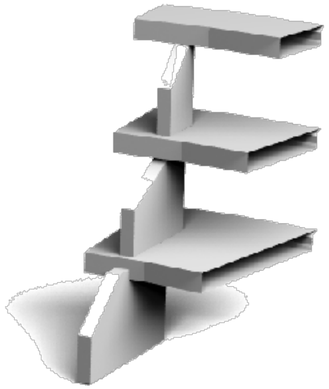




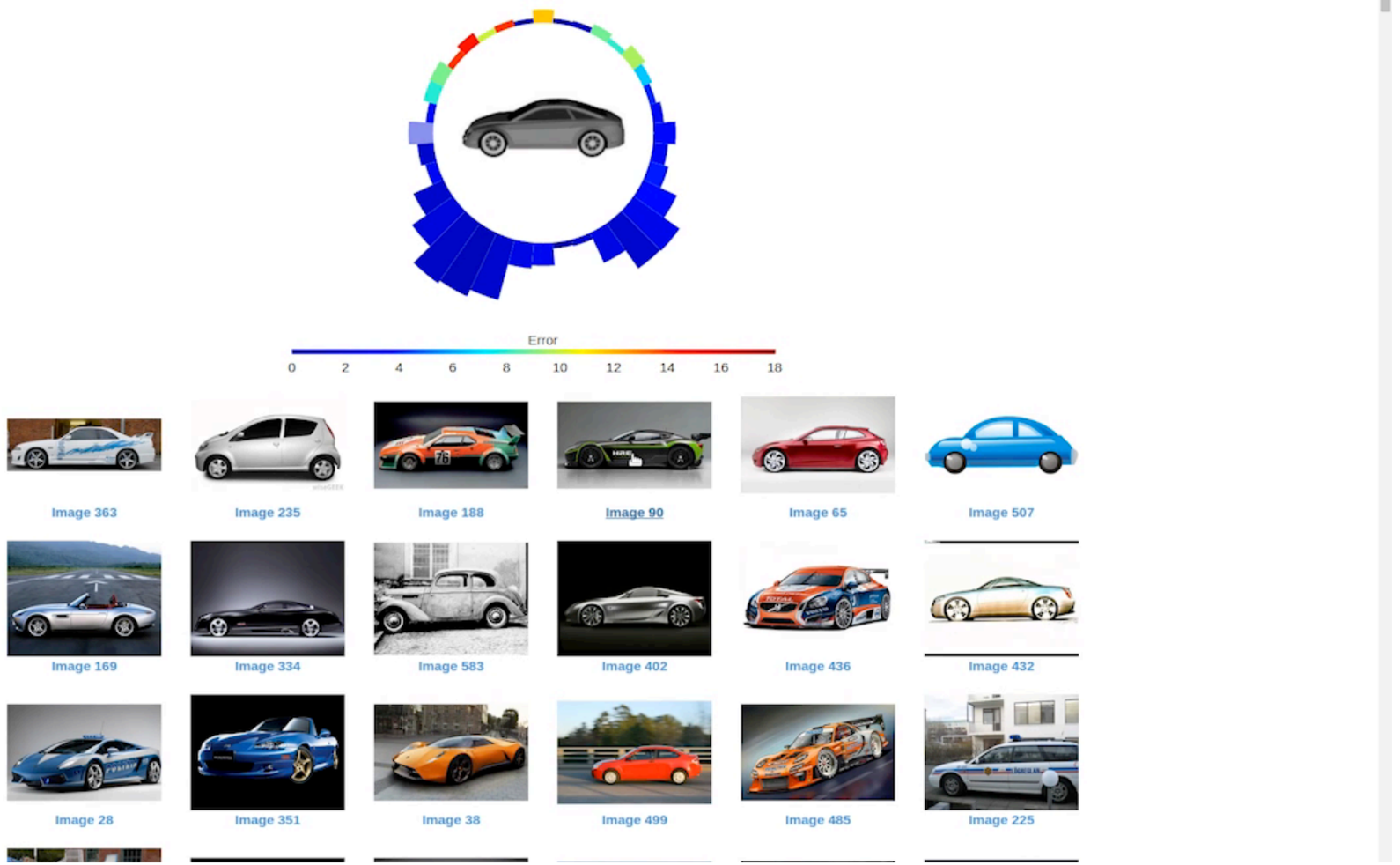




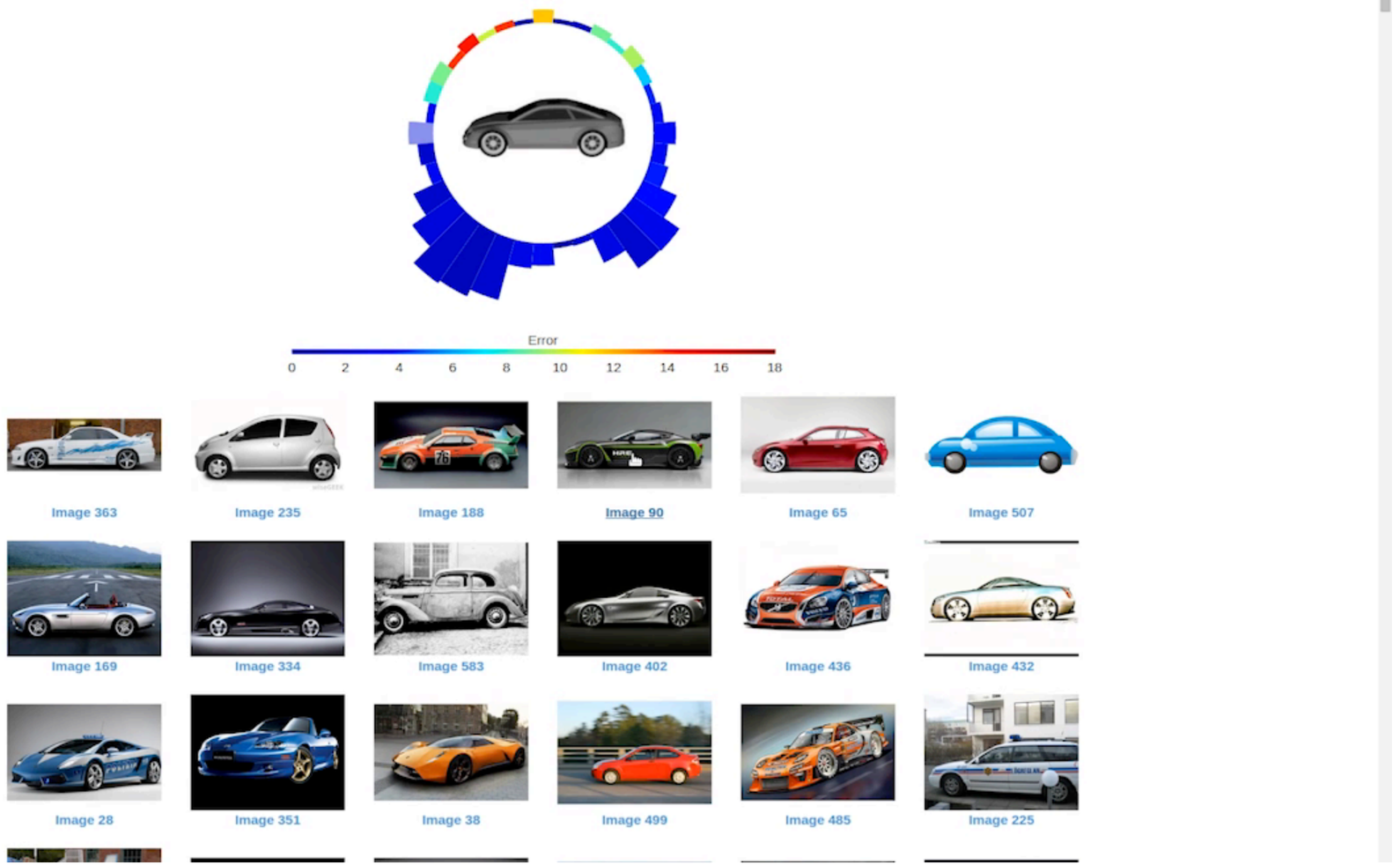




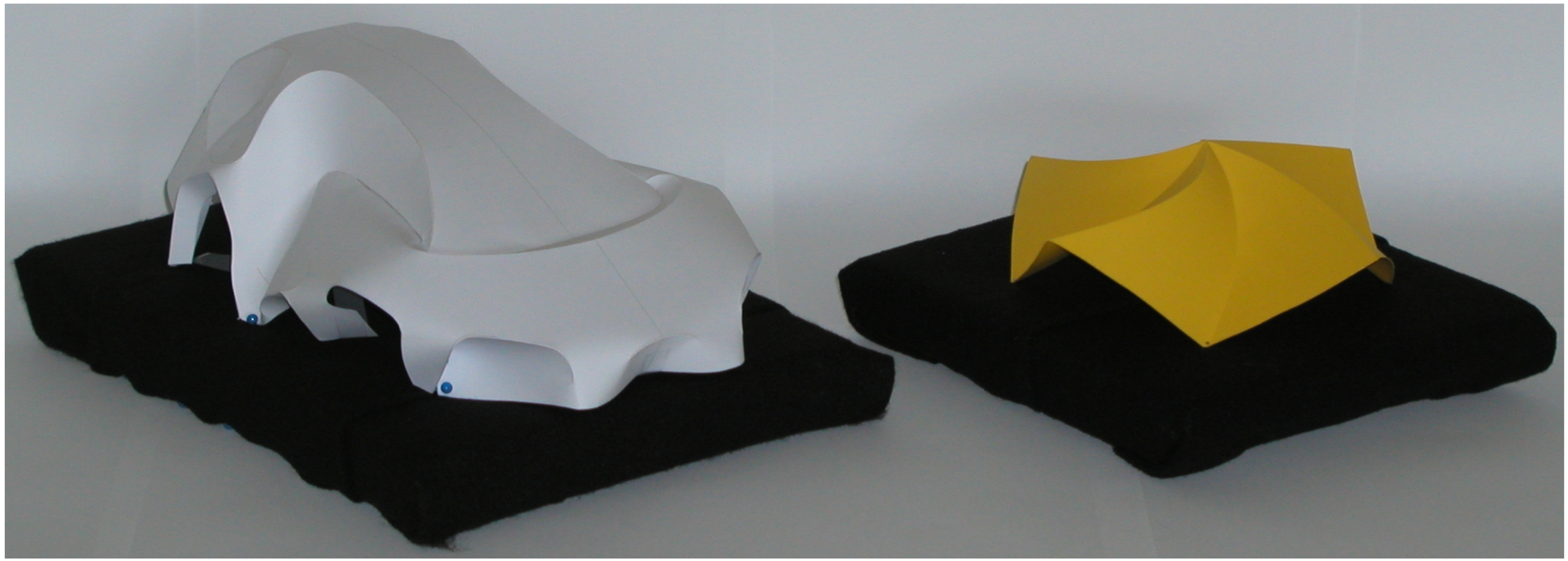
CrossLink: Images <> 3D Models



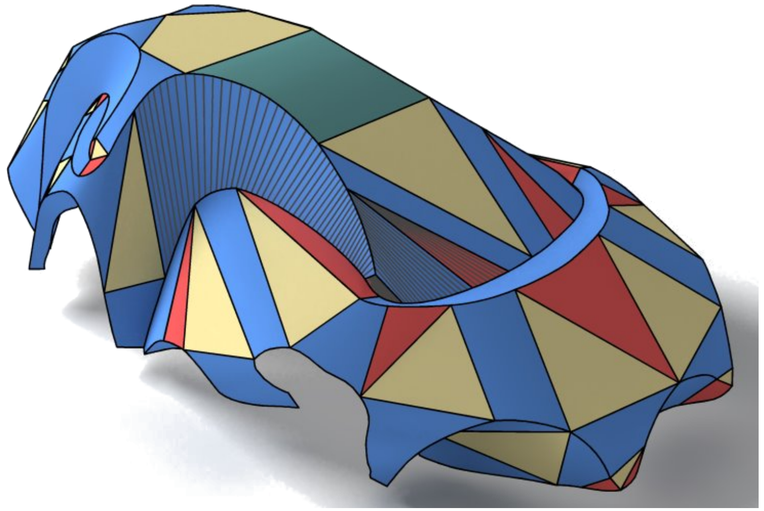
CrossLink: Images <> 3D Models



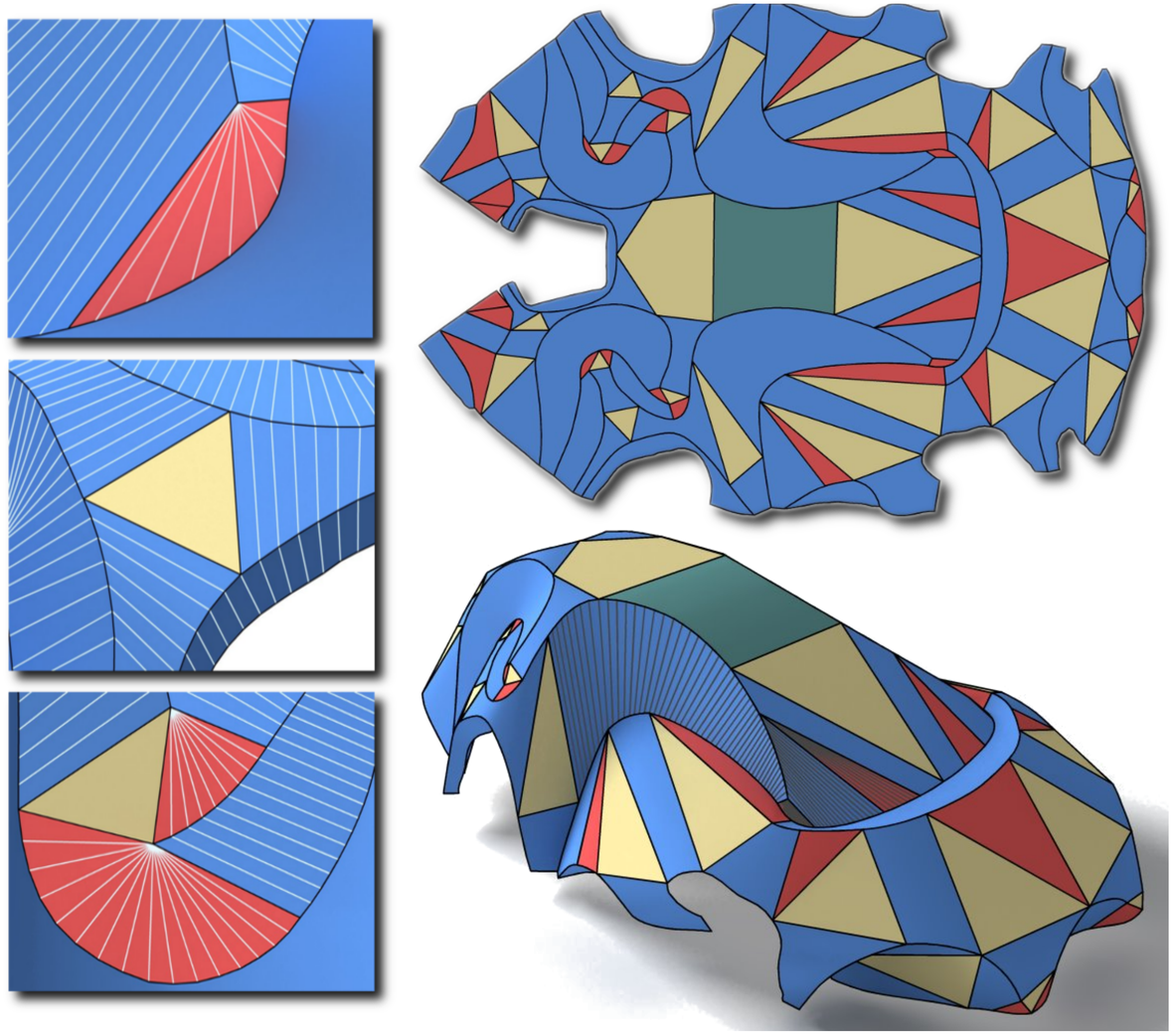
Geometry of Folds



Layout in 2D



Layout in 2D



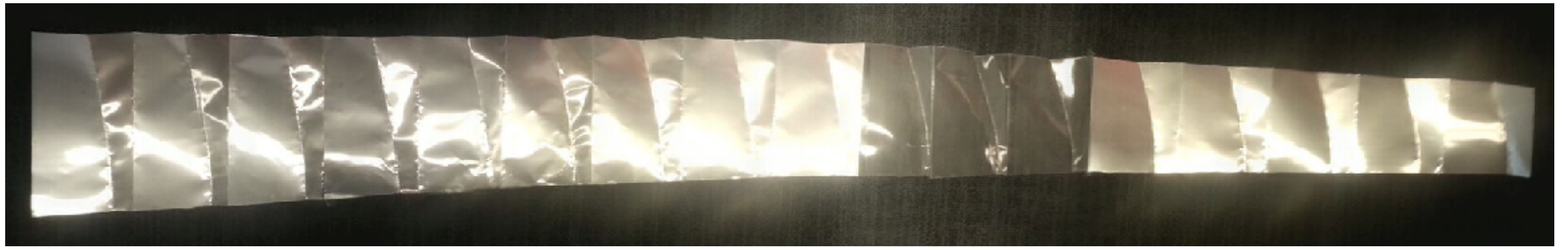
Folded Roof

Folded Roof

Folded Roof

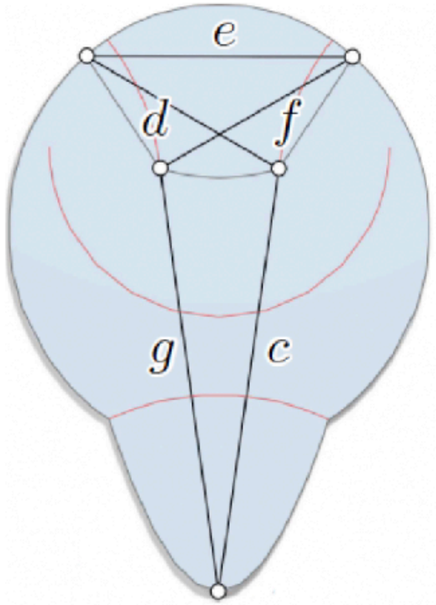

UCL
UCL ENGINEERING
Change the world

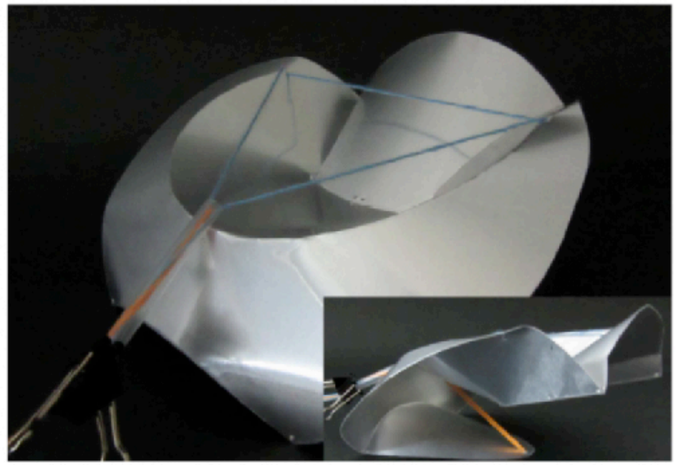

UCL ENGINEERING
Change the world
UCL



Concept Chair

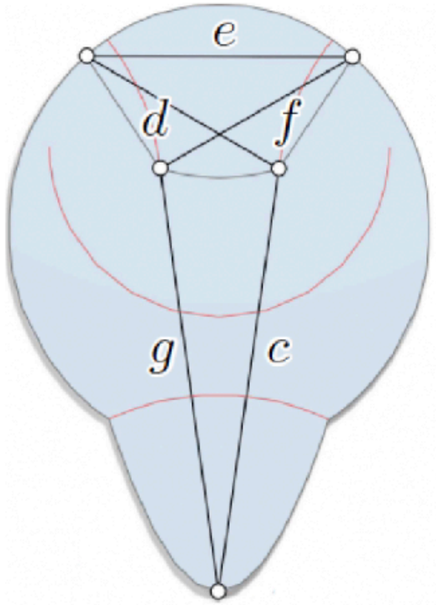
Sequence: CONCEPT CHAIR

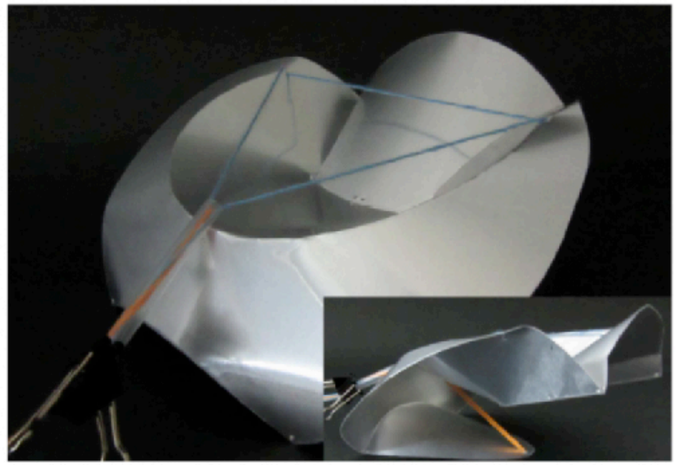




Concept Chair

Sequence: CONCEPT CHAIR





Overview

Overview

Geometric Modeling

- techniques and algorithms for representing and processing geometric objects

Overview

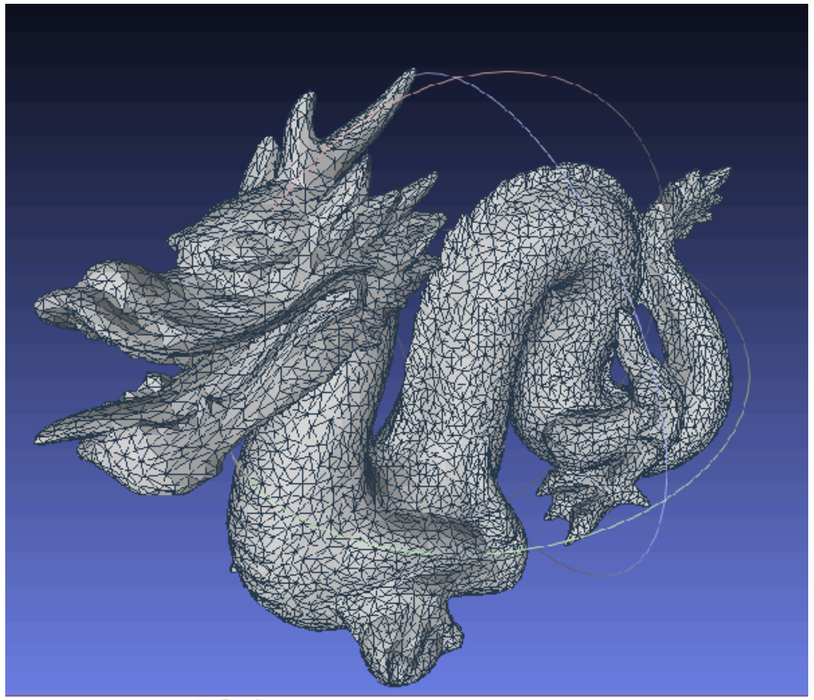
Geometric Modeling

- techniques and algorithms for representing and processing geometric objects

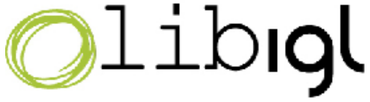
In this course, we will focus on **triangle meshes**

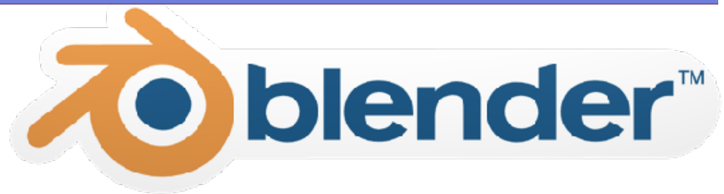
- **why** are triangle meshes a suitable representation for geometry processing?
- **what** are the central processing algorithms?
- **how** can they be implemented efficiently?

Looking at Meshes

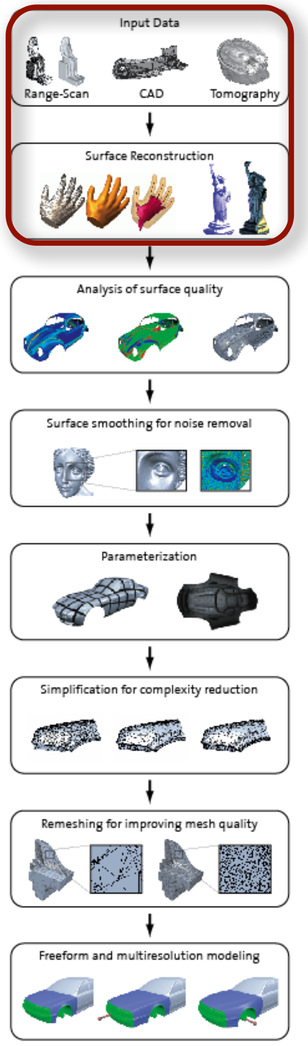


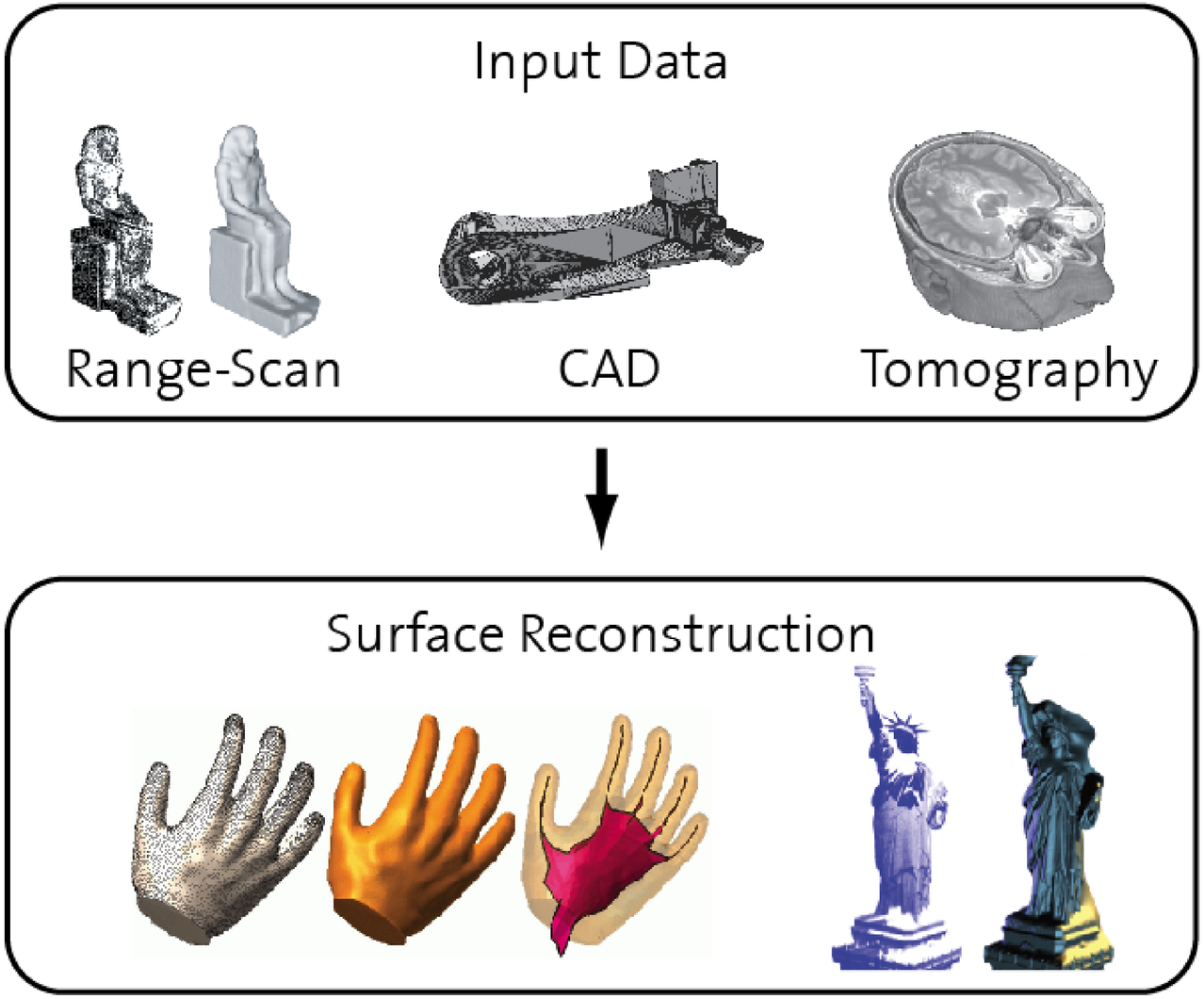
OpenMesh



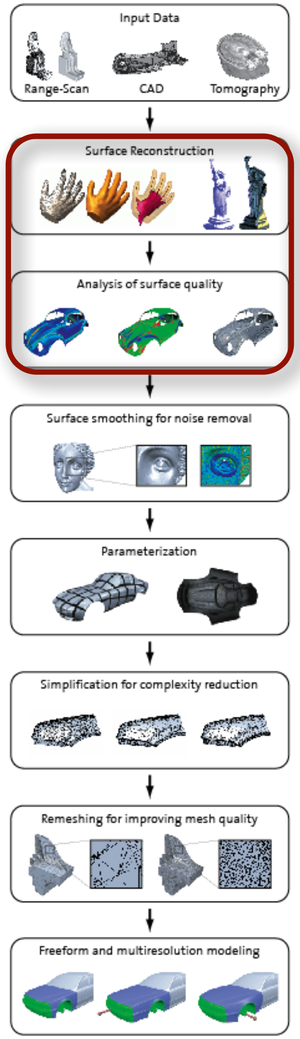


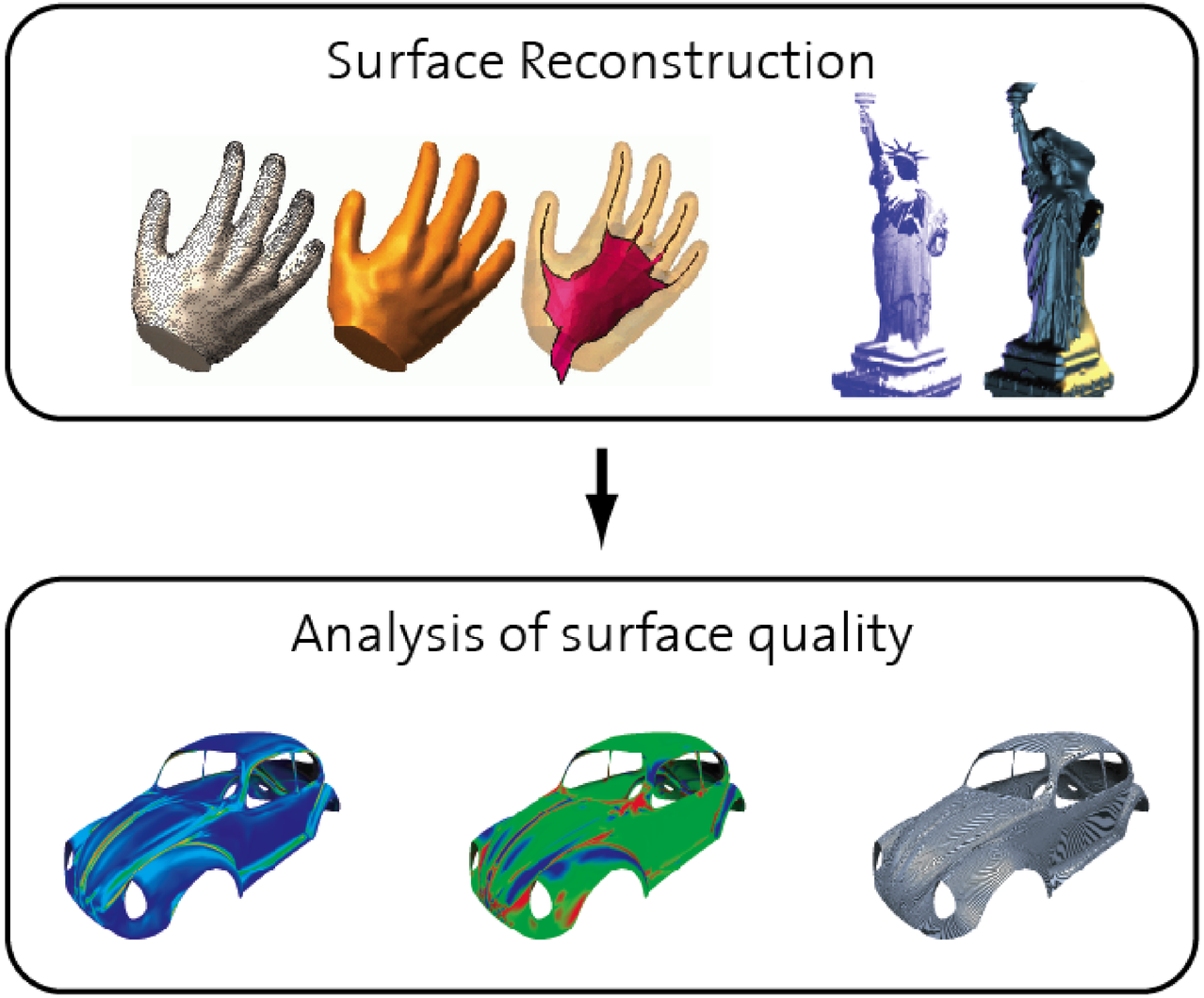
Geometry Processing Pipeline



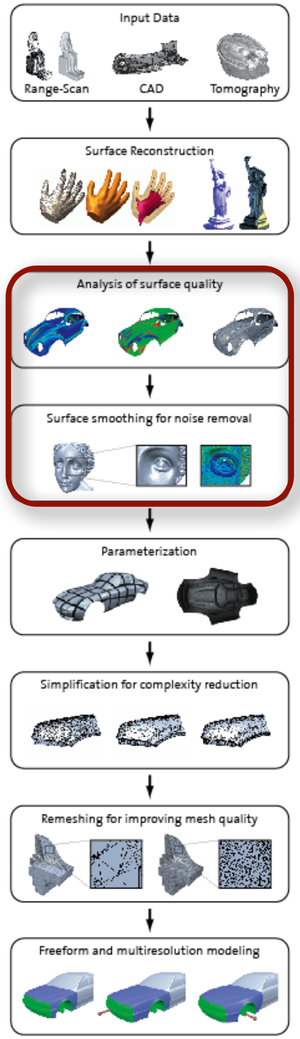


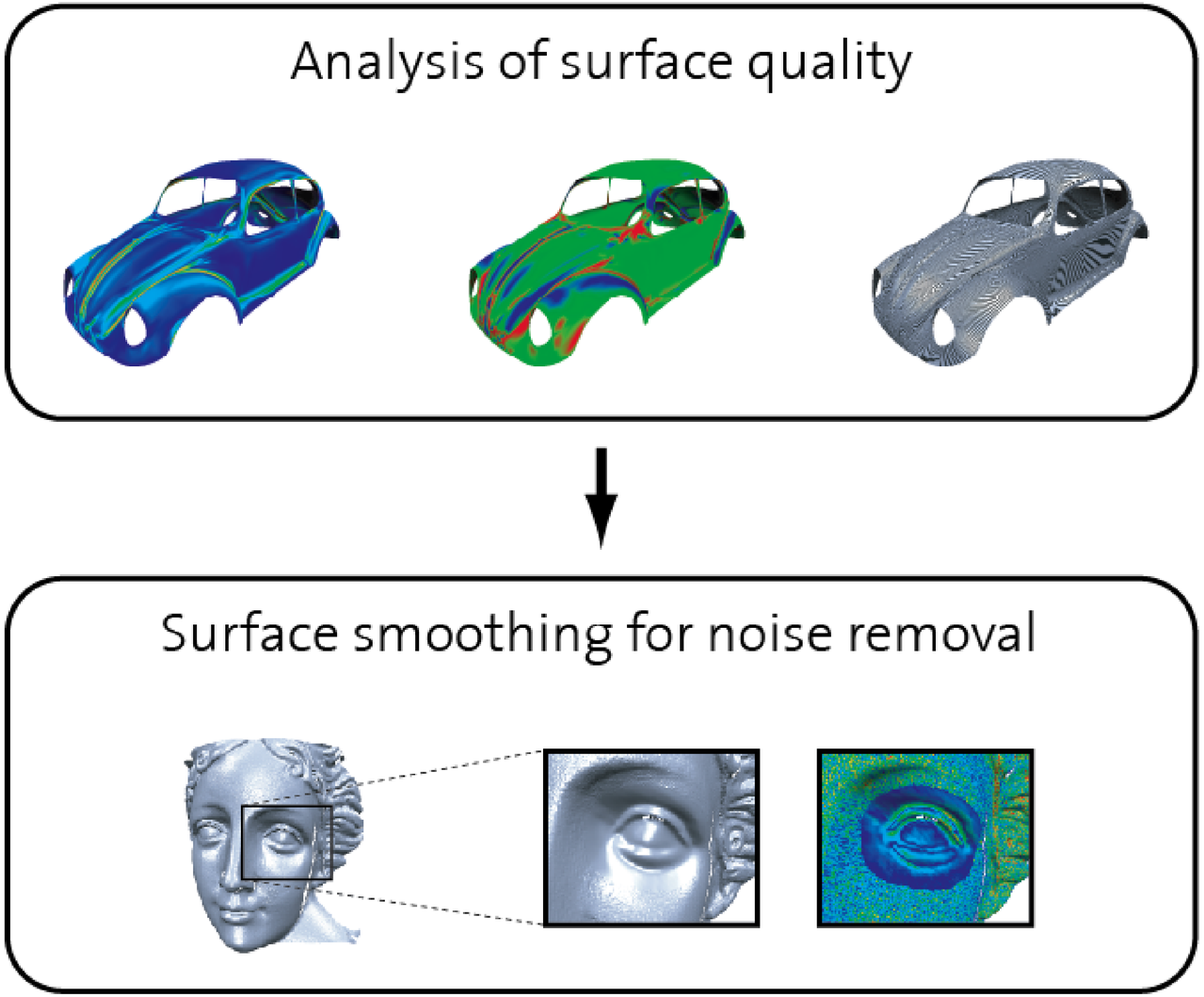
Geometry Processing Pipeline



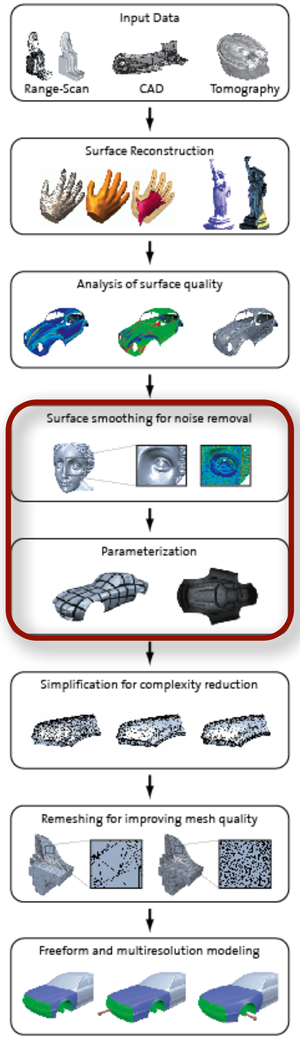


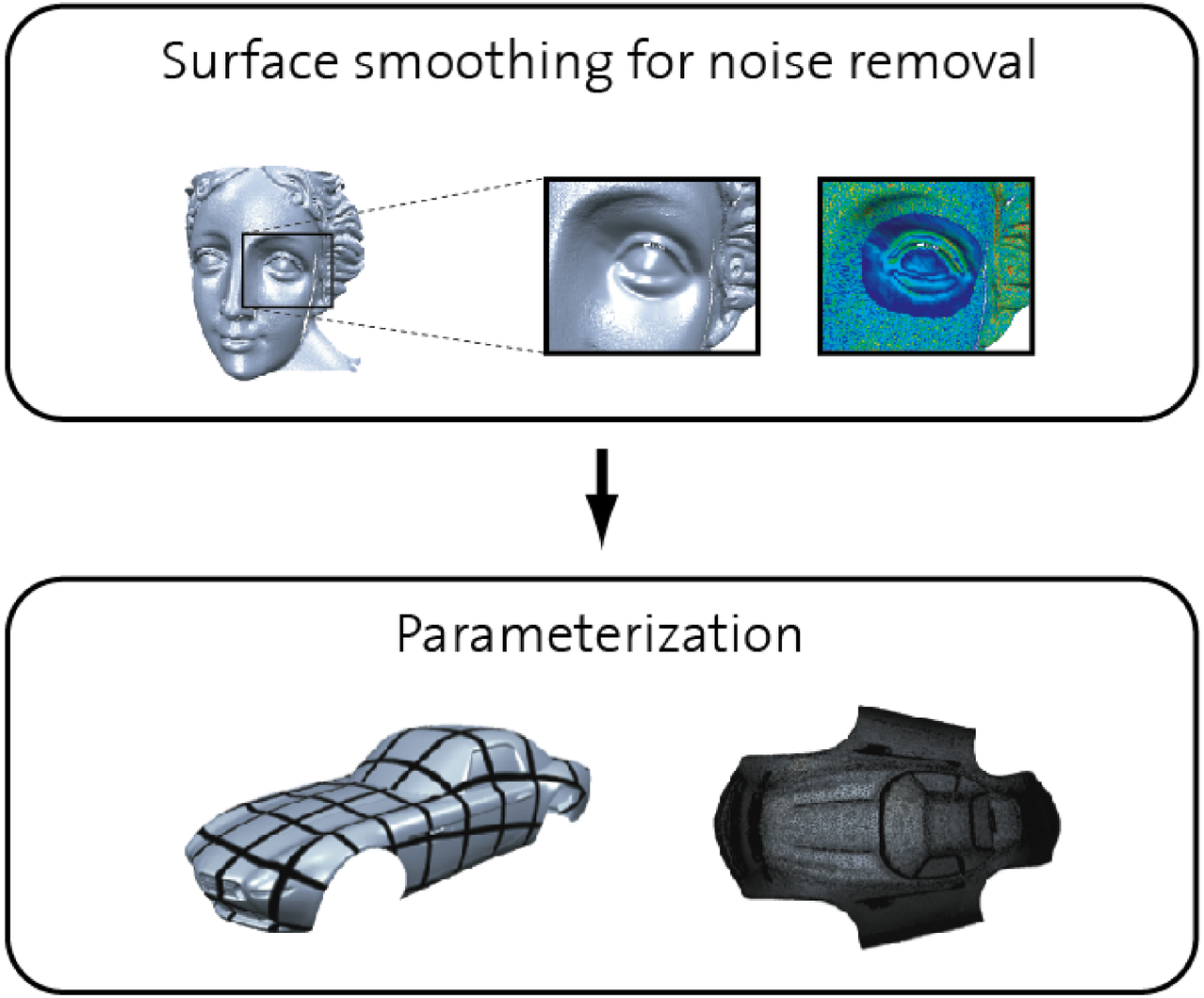
Geometry Processing Pipeline



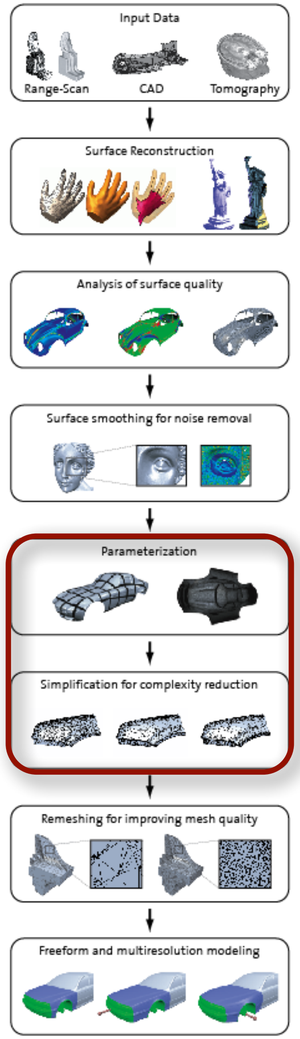


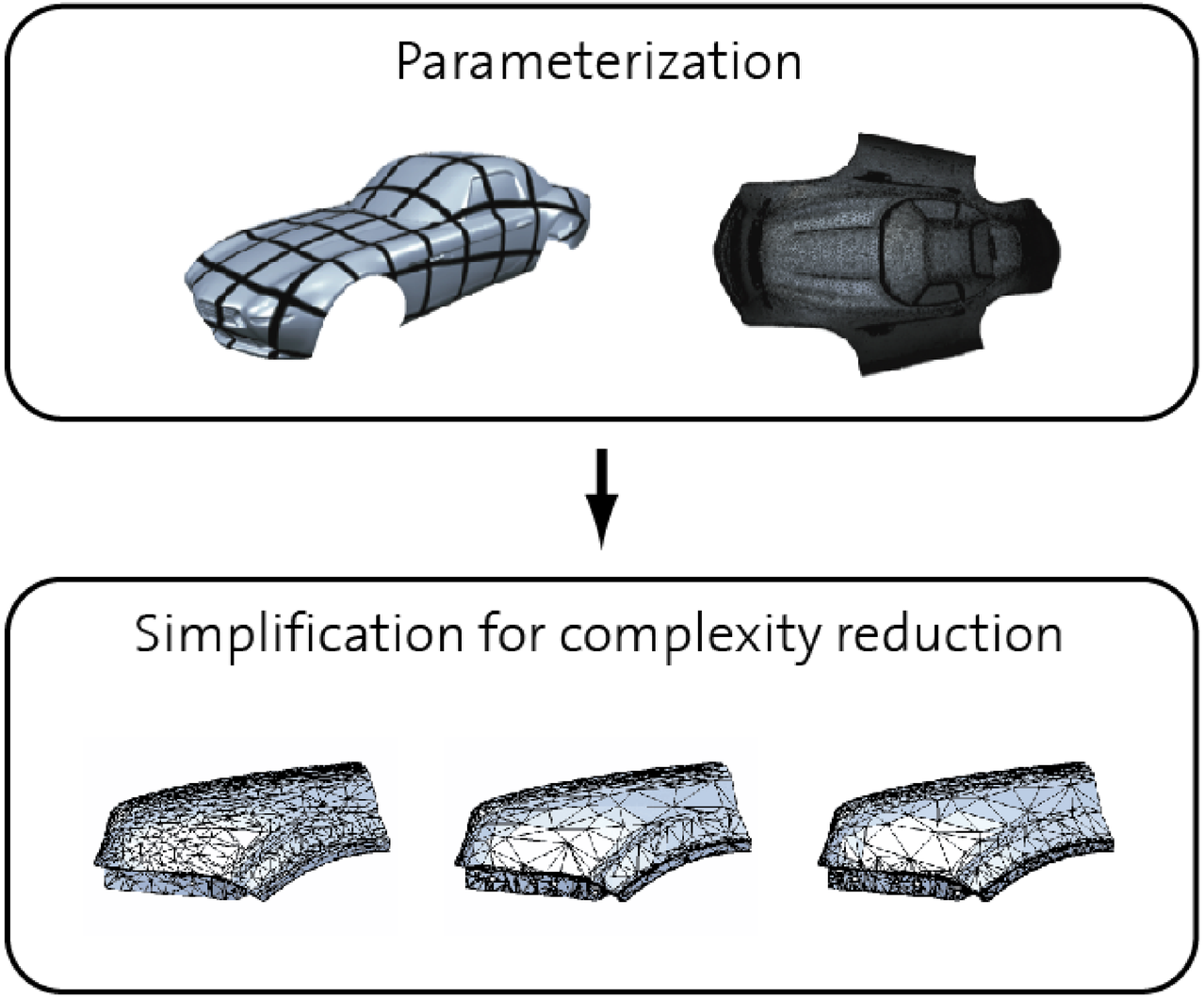
Geometry Processing Pipeline



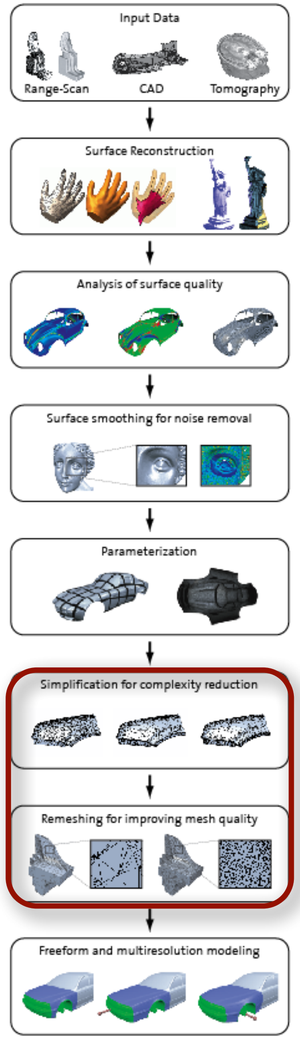


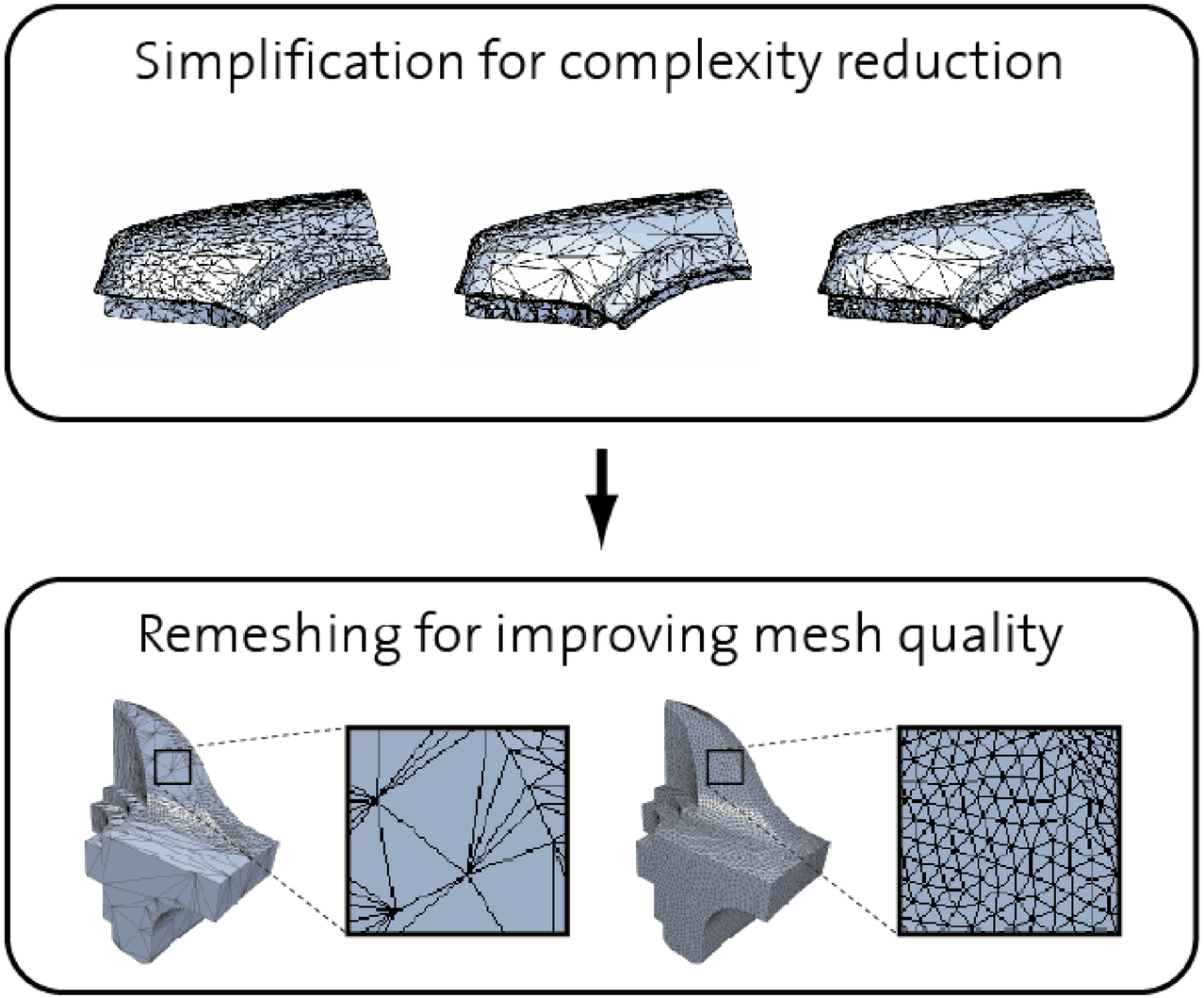
Geometry Processing Pipeline



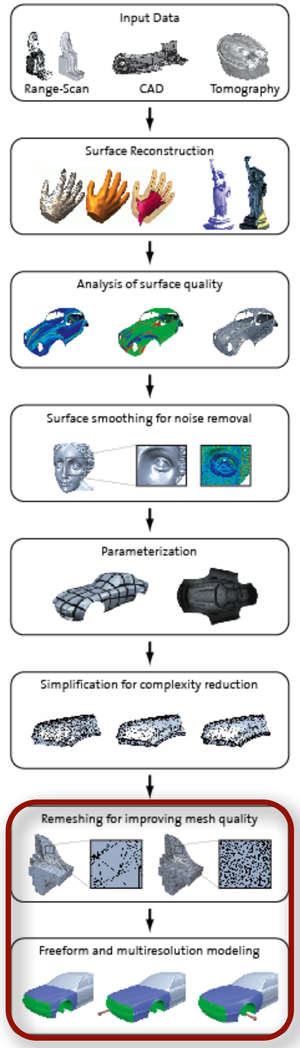


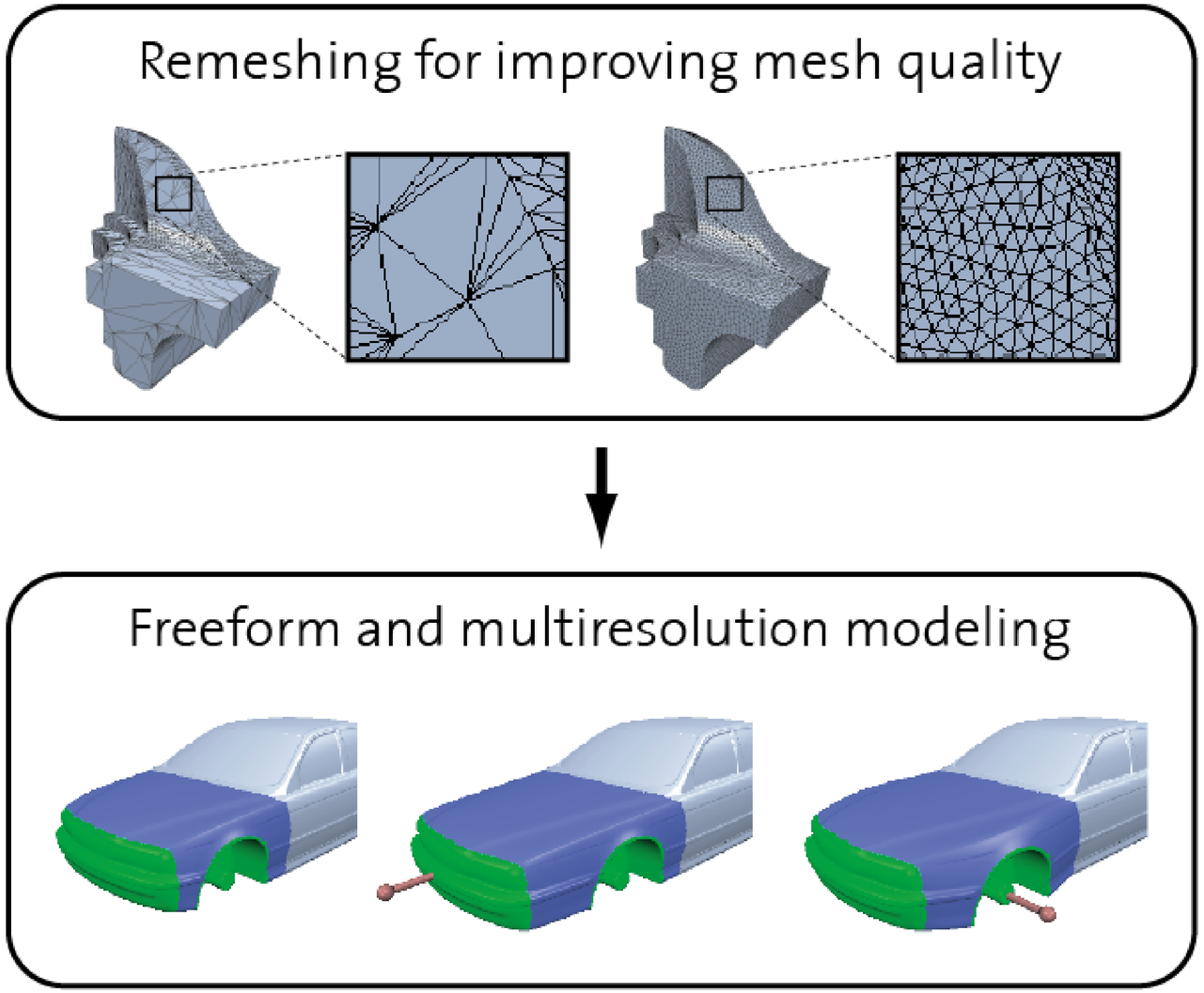
Geometry Processing Pipeline



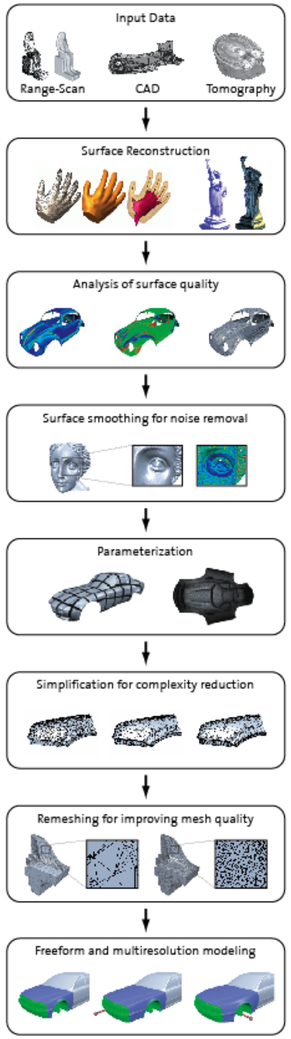


Geometry Processing Pipeline

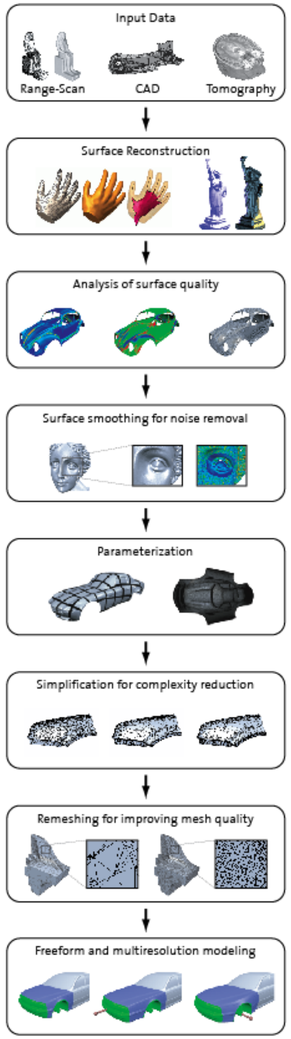




Geometry Processing Pipeline



Geometry Processing Pipeline



Course Objectives

Course Objectives

After attending the course, you should be able to

Course Objectives

After attending the course, you should be able to

- ***define*** and ***relate*** the basic concepts, tools, and algorithms in geometric modeling and digital geometry processing

Course Objectives

After attending the course, you should be able to

- ***define*** and ***relate*** the basic concepts, tools, and algorithms in geometric modeling and digital geometry processing
- critically ***analyze*** and ***assess*** current research on surface representations and geometric modeling and apply the proposed methods in your own work

Course Objectives

After attending the course, you should be able to

- ***define*** and ***relate*** the basic concepts, tools, and algorithms in geometric modeling and digital geometry processing
- critically ***analyze*** and ***assess*** current research on surface representations and geometric modeling and apply the proposed methods in your own work
- ***design*** and ***implement*** individual components of a geometric modeling system

Exercises

Programming assignments

- based on OpenMesh/Meshlab/Trimesh/others
- cover the core stages of the geometry processing pipeline
- framework including mesh viewer will be provided (Python, C/C++)

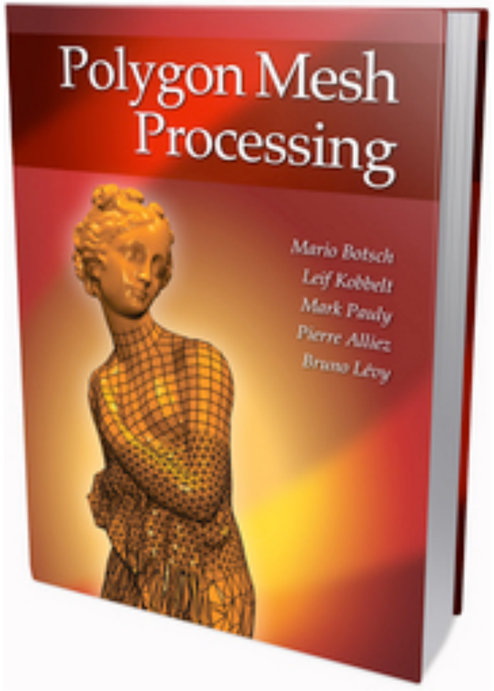
Final project (groups of 2)

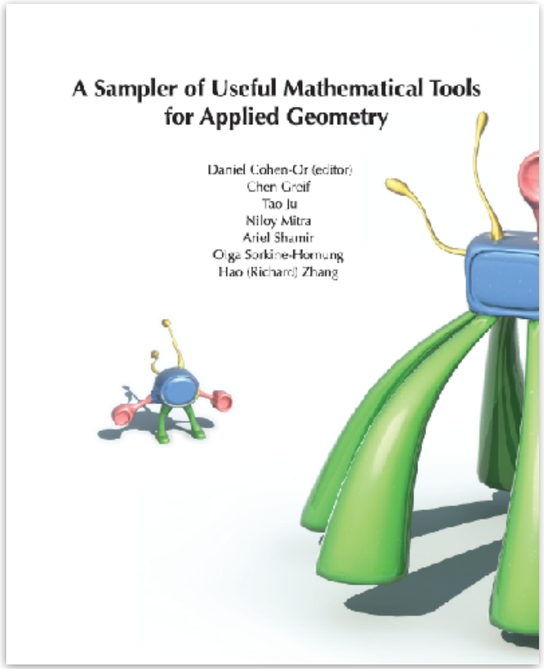
Integral part of the lecture

- important for achieving course objectives

Literature

- Botsch, Kobbelt, Pauly, Alliez, Levy:
  Polygon Mesh Processing, AK Peters, 2010
- Cohen-Or, Greif, Ju, Mitra, Shamir, Sorkine, Zhang:
  A Sampler of Useful Mathematical Tools for Applied Geometry, CRC, 2015





Outline

- Parametric Approximations
- Polygon Meshes
- Data structures

Outline

- **Parametric Approximations**
- Polygon Meshes
- Data structures

Parametric Representation

Parametric Representation

- Surface is the **range** of a function

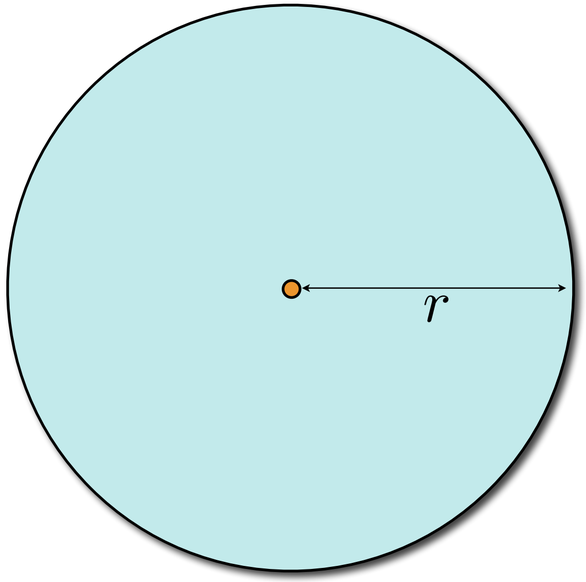
$$\mathbf{f} : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad \mathcal{S}_\Omega = \mathbf{f}(\Omega)$$

$$\mathbf{c} : \Lambda \in \mathbb{R} \rightarrow \mathbb{R}^2, \quad \mathcal{C}_\Lambda = \mathbf{c}(\Lambda)$$

- 2D example: Circle

$$\mathbf{f} : [0, 2\pi] \rightarrow \mathbb{R}^2$$

$$\mathbf{f}(t) = \begin{pmatrix} r\cos(t) \\ r\sin(t) \end{pmatrix}$$



Parametric Representation

- Surface is the range of a function

$$\mathbf{f} : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad \mathcal{S}_\Omega = \mathbf{f}(\Omega)$$

- 2D example: Island coast line

$$\mathbf{f} : [0, 2\pi] \rightarrow \mathbb{R}^2$$

$$\mathbf{f}(t) = \begin{pmatrix} ??? \\ ??? \end{pmatrix}$$

Piecewise Approximation

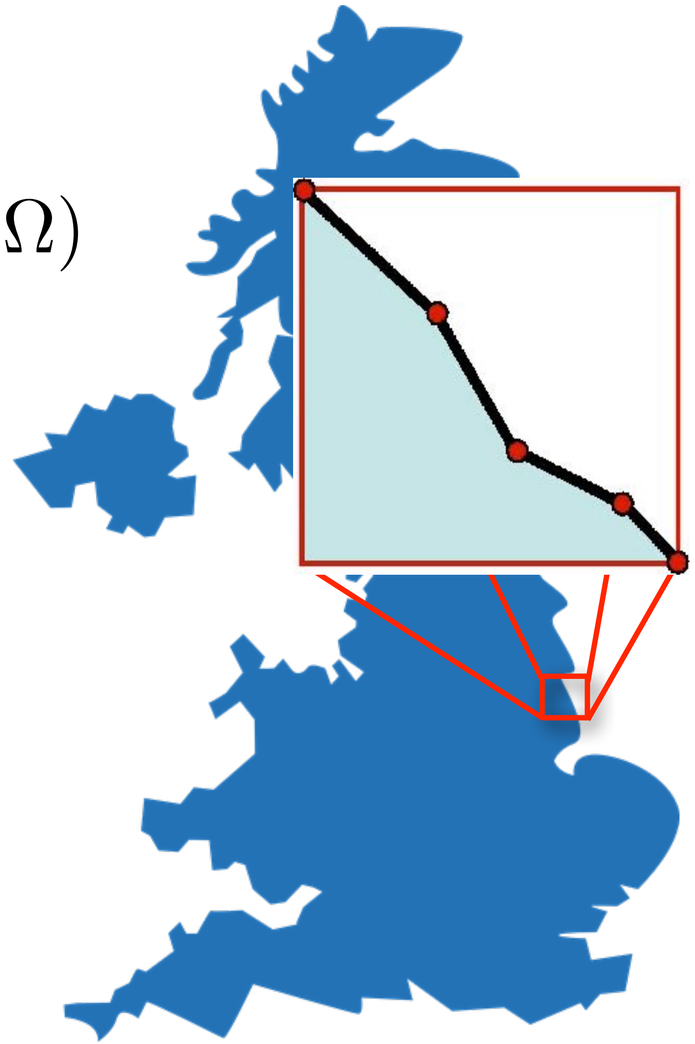
- Surface is the range of a function

$$\mathbf{f} : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad \mathcal{S}_\Omega = \mathbf{f}(\Omega)$$

- 2D example: Island coast line

$$\mathbf{f} : [0, 2\pi] \rightarrow \mathbb{R}^2$$

$$\mathbf{f}(t) = \begin{pmatrix} ??? \\ ??? \end{pmatrix}$$



Polynomial Approximation

Polynomial Approximation

- Polynomials are computable functions

$$f(t) = \sum_{i=0}^{p} c_i \, t^i = \sum_{i=0}^{p} \tilde{c}_i \, \phi_i(t)$$

Polynomial Approximation

- Polynomials are computable functions

$$f(t) \;=\; \sum_{i=0}^{p} c_i\, t^i \;=\; \sum_{i=0}^{p} \tilde{c}_i\, \phi_i(t)$$

- **Taylor expansion** up to degree *p*

$$g(h) \;=\; \sum_{i=0}^{p} \frac{1}{i!}\, g^{(i)}(0)\; h^i \;+\; O\big(h^{p+1}\big)$$

Polynomial Approximation

- Polynomials are computable functions

$$f(t) \;=\; \sum_{i=0}^{p} c_i\, t^i \;=\; \sum_{i=0}^{p} \tilde{c}_i\, \phi_i(t)$$

- **Taylor expansion** up to degree *p*

$$g(h) \;=\; \sum_{i=0}^{p} \frac{1}{i!}\, g^{(i)}(0)\; h^i \;+\; O\big(h^{p+1}\big)$$

- Error for approximating a function *g* by polynomial *f*

$$f(t_i) = g(t_i)\,, \quad 0 \le t_0 < \cdots < t_p \le h$$

$$|f(t) - g(t)| \;\le\; \frac{1}{(p+1)!}\, \max f^{(p+1)} \prod_{i=0}^{p} (t - t_i) \;=\; O\Big(h^{(p+1)}\Big)$$

Polynomial Approximation

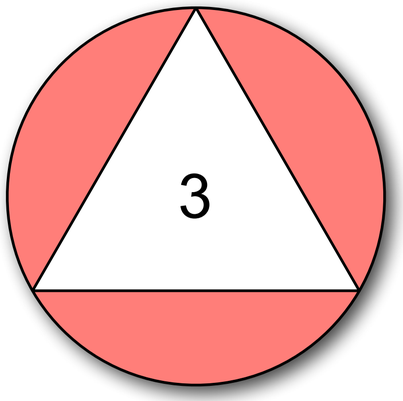
- **Approximation error** is $O(h^{p+1})$

- **Improve** approximation quality by
  - increasing *p* ... higher order polynomials
  - decreasing *h* ... smaller / more segments

- Issues
  - _ smoothness of the target data $\max_t f^{p+1}(t)$
  - smoothness conditions between segments

Polygonal Meshes

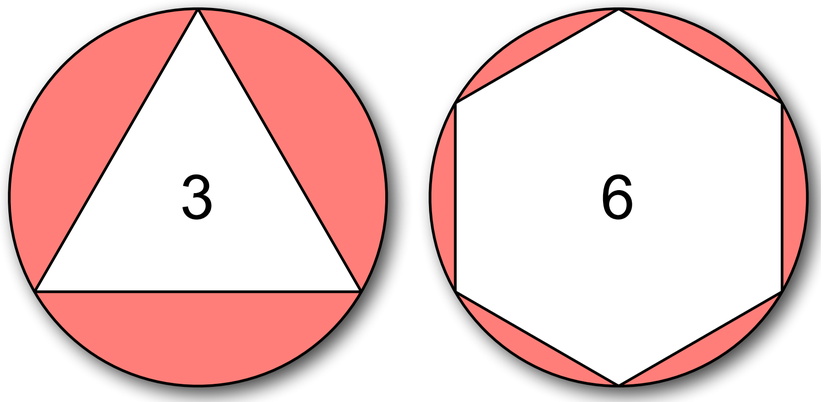
- **Polygonal meshes** are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$



25%

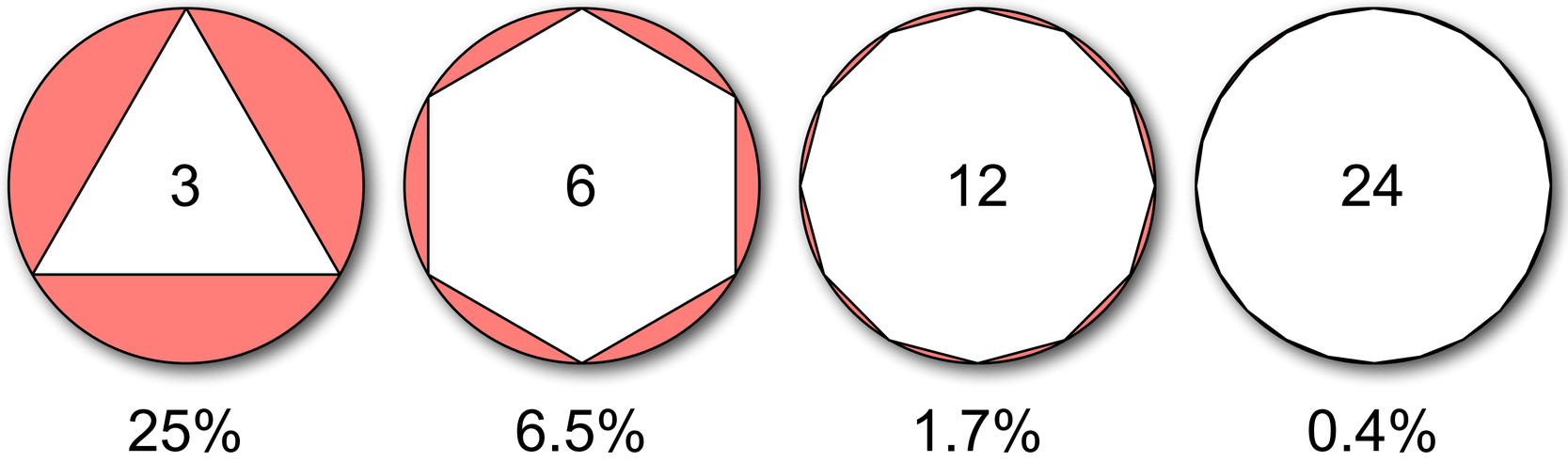
Polygonal Meshes

- **Polygonal meshes** are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$



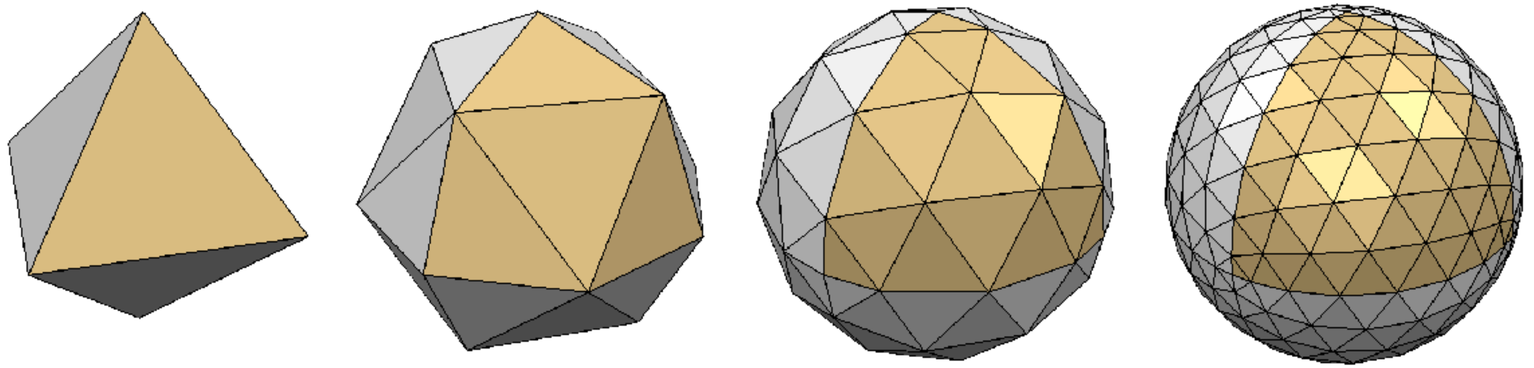
Polygonal Meshes

- **Polygonal meshes** are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$



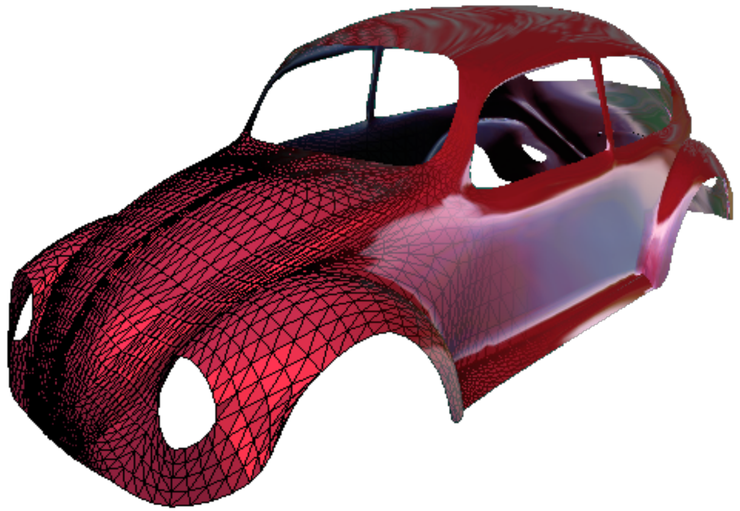
Polygonal Meshes

- Polygonal meshes are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inverse proportional to #faces



Polygonal Meshes

- Polygonal meshes are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inverse proportional to #faces
  - Arbitrary topology surfaces

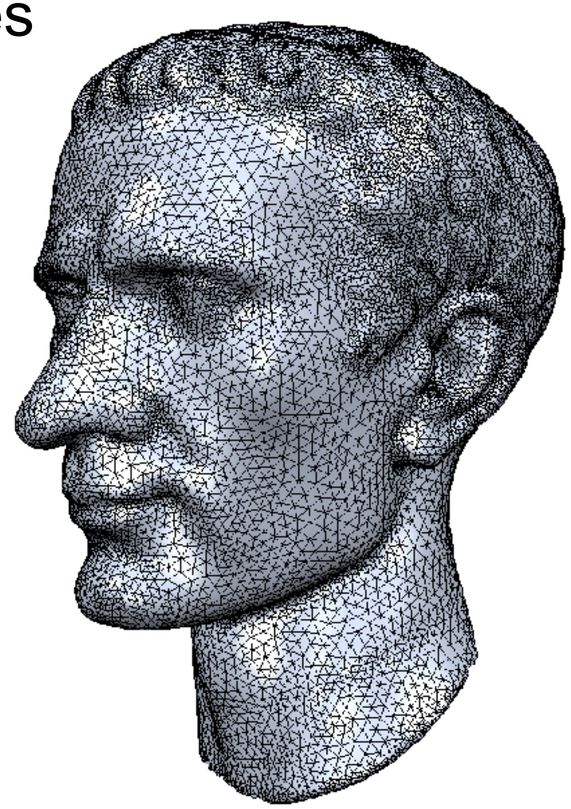


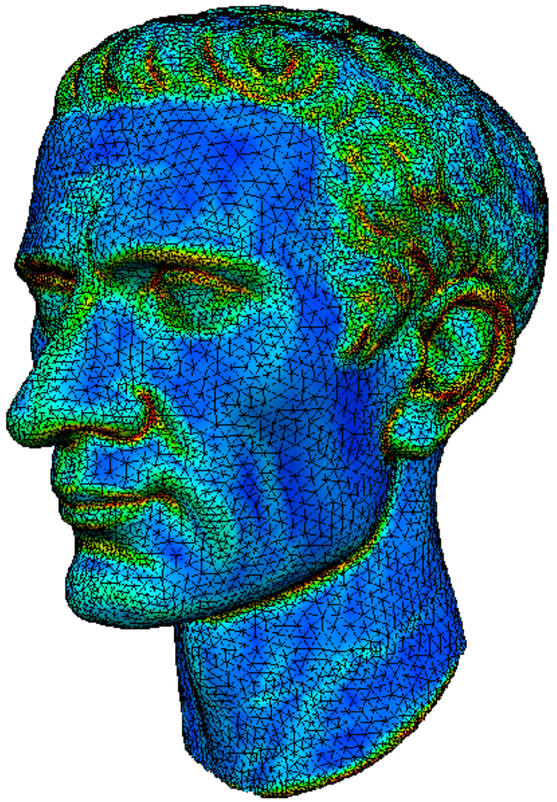
Polygonal Meshes

- Polygonal meshes are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inverse proportional to #faces
  - Arbitrary topology surfaces
  - Piecewise smooth surfaces

Polygonal Meshes

- Polygonal meshes are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inverse proportional to #faces
  - Arbitrary topology surfaces
  - Piecewise smooth surfaces
  - Curvature adaptive sampling



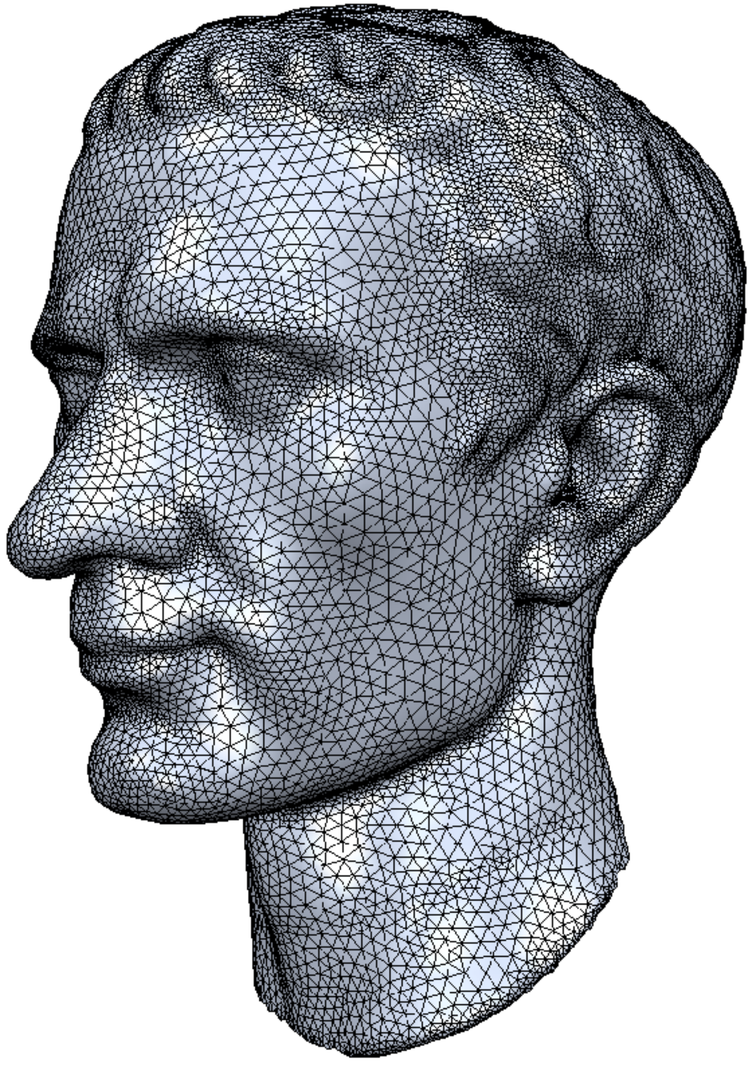


Polygonal Meshes

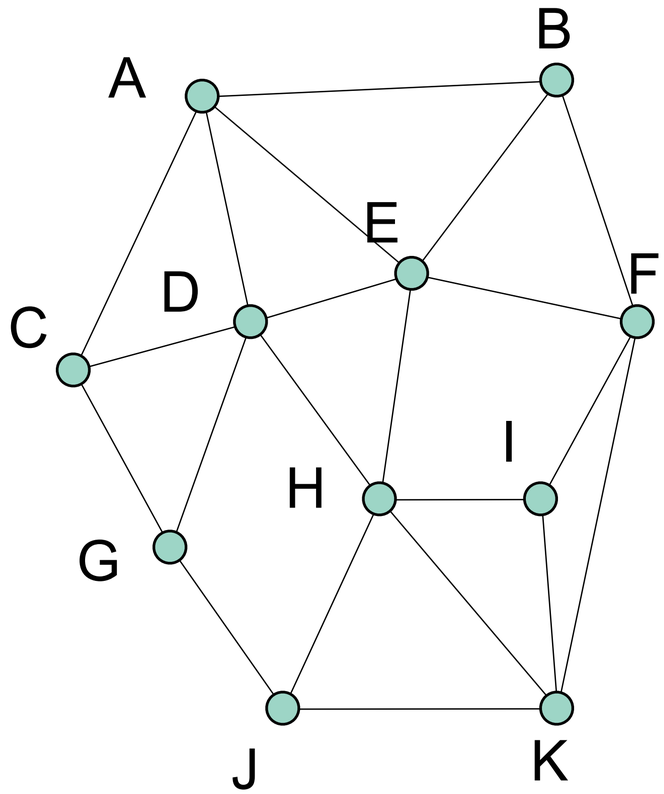
- Polygonal meshes are a good compromise
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inverse proportional to #faces
  - Arbitrary topology surfaces
  - Piecewise smooth surfaces
  - Curvature adaptive sampling
  - Efficient GPU-based rendering

Outline

- Parametric Approximations
- **Polygon Meshes**
- Data structures

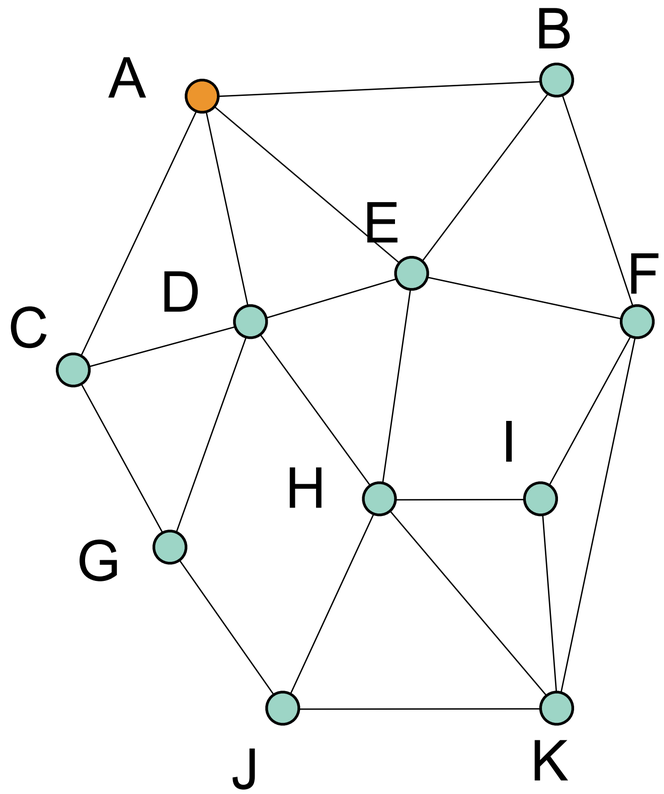


Graph Definitions



Graph $\{V, E\}$

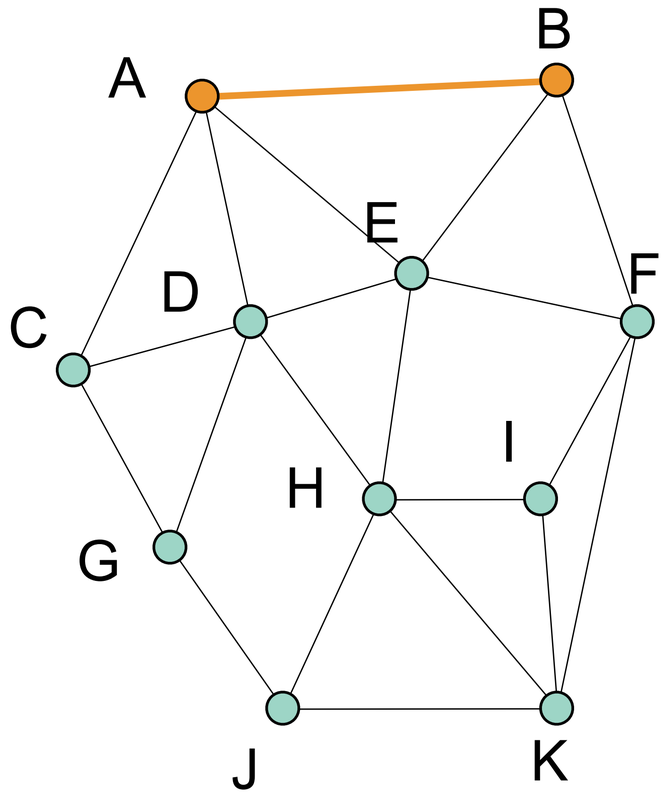
Graph Definitions



Graph $\{V, E\}$

Vertices $V = \{A, B, C, \ldots, K\}$

Graph Definitions

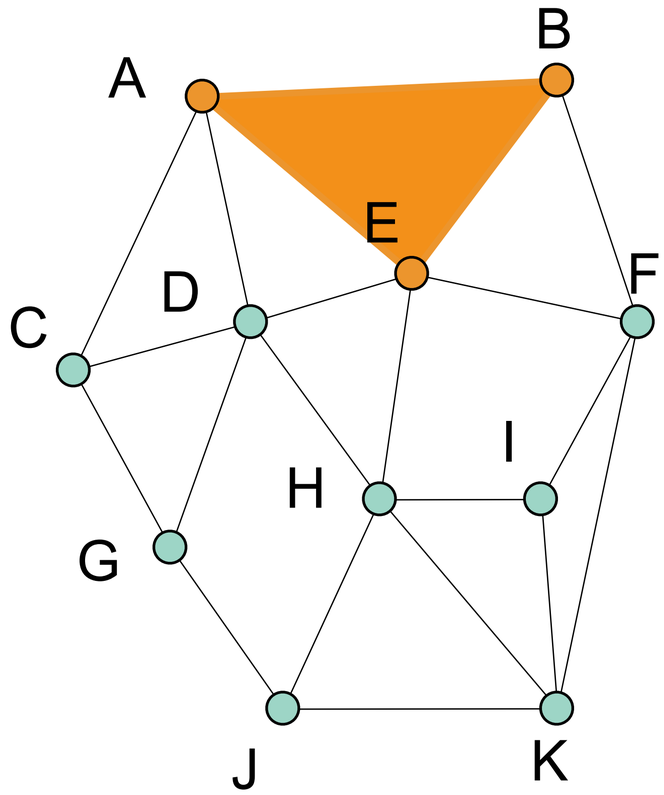


Graph $\{V, E\}$

Vertices $V = \{A, B, C, \ldots, K\}$

Edges $E = \{(AB), (AE), (CD), \ldots\}$

Graph Definitions



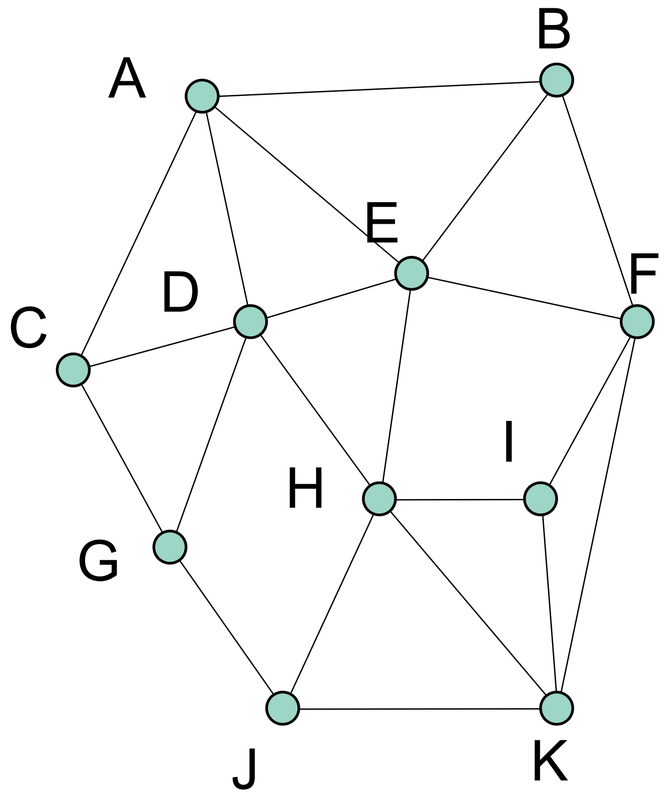
Graph $\{V, E\}$

Vertices $V = \{A, B, C, \ldots, K\}$

Edges $E = \{(AB), (AE), (CD), \ldots\}$

Faces $F = \{(ABE), (EBF), (EFIH), \ldots\}$

Graph Definitions



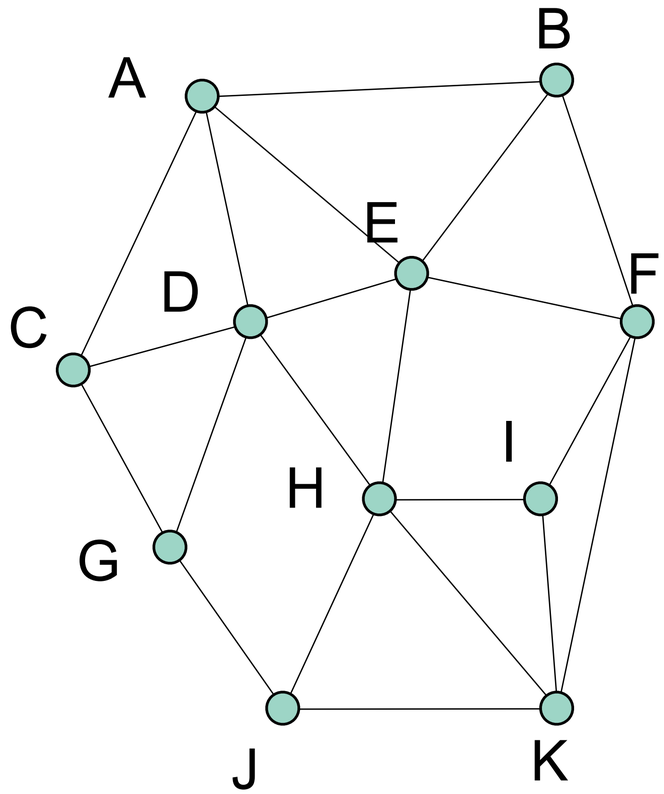
Vertex degree or **valence**:
number of incident edges.

deg(A) = 4

deg(E) = 5

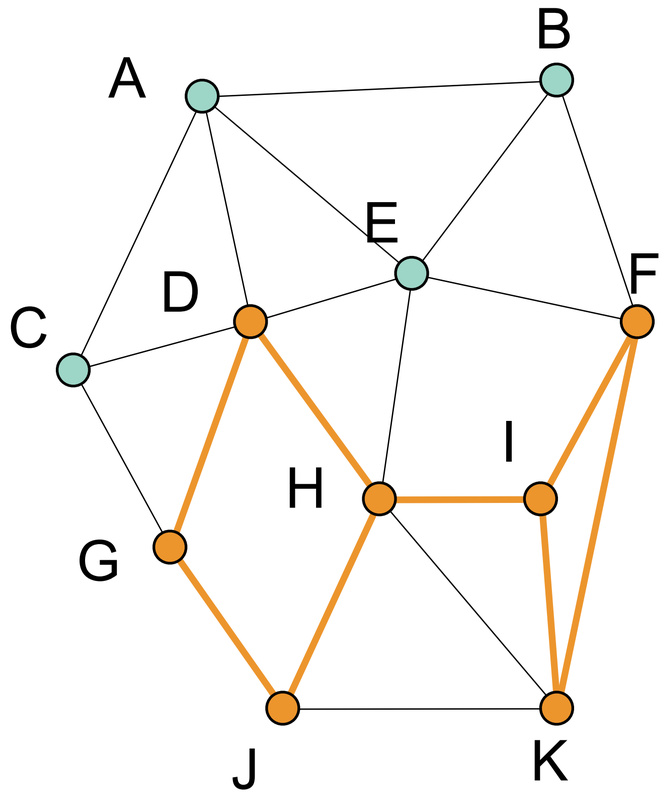
For triangle mesh, **average** valence is 6.

Connectivity



Connected:
Path of edges connecting every two vertices.

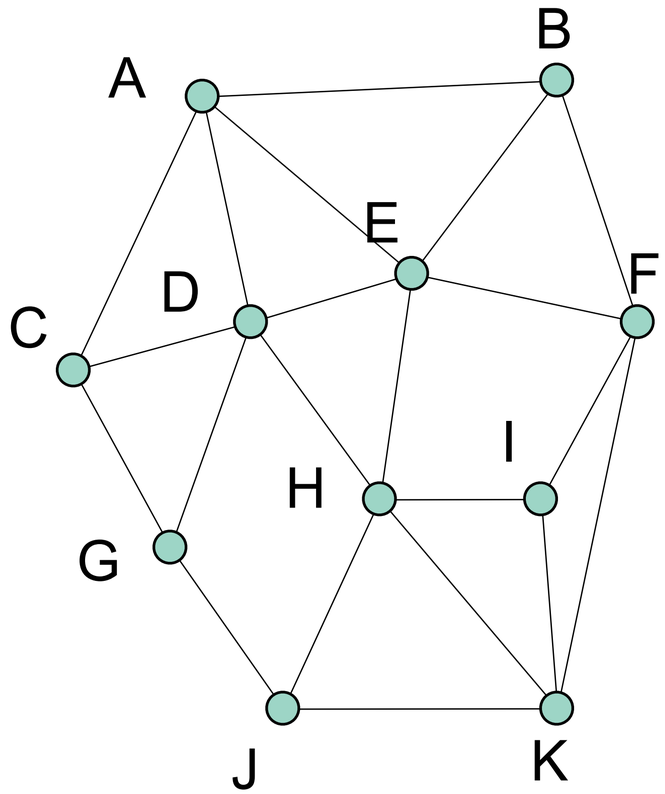
Connectivity



Connected:
Path of edges connecting every two vertices.

Subgraph: Graph $\{V', E'\}$ is a subgraph of graph $\{V, E\}$ if $V' \subset V$ with $E' \subset E$ *and* incident on V'.

Connectivity

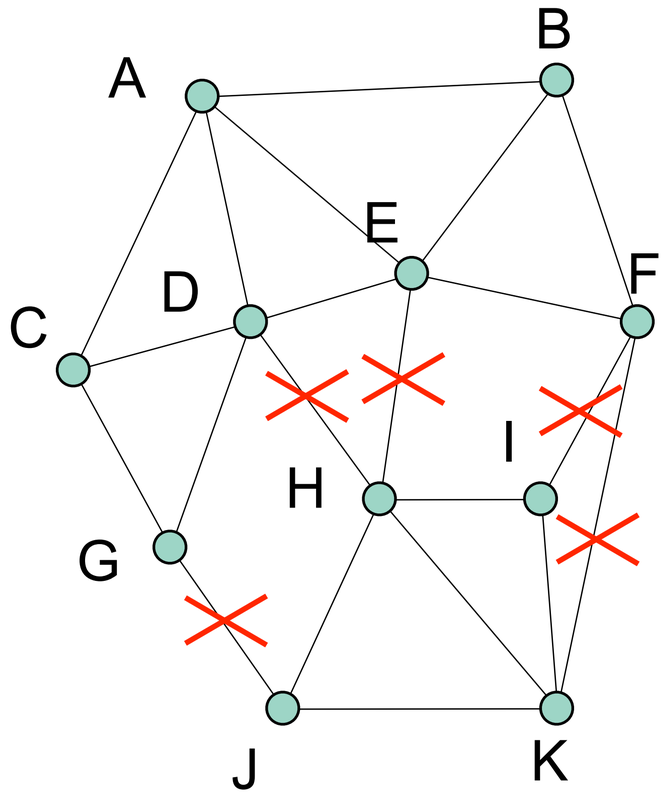


Connected:
Path of edges connecting every two vertices.

Subgraph: Graph $\{V', E'\}$ is a subgraph of graph $\{V, E\}$ if $V' \subset V$ with $E' \subset E$ *and* incident on V'.

Connected component:
Maximally connected subgraph.

Connectivity

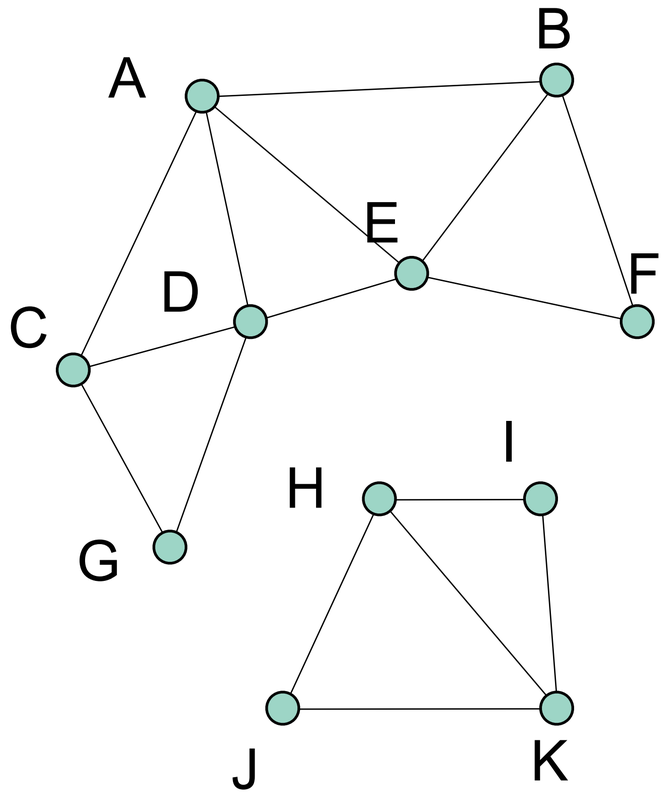


Connected:
Path of edges connecting every two vertices.

Subgraph: Graph $\{V', E'\}$ is a subgraph of graph $\{V, E\}$ if $V' \subset V$ with $E' \subset E$ *and* incident on V'.

Connected component:
Maximally connected subgraph.

Connectivity



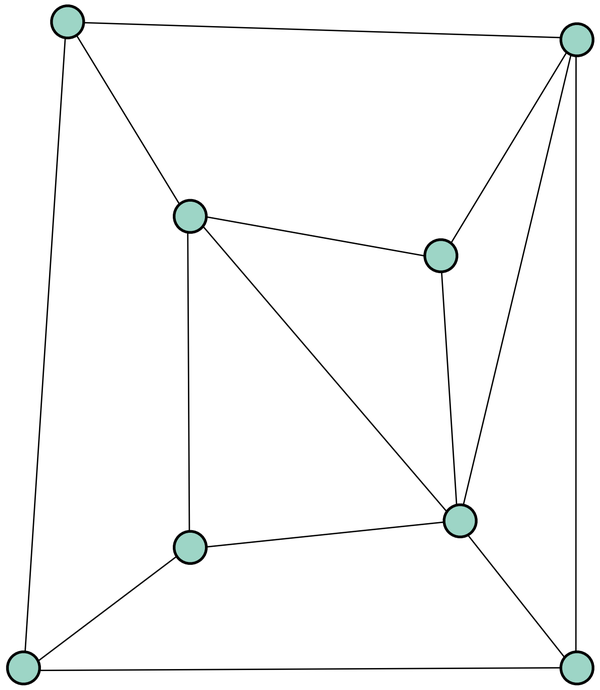
Connected:
Path of edges connecting every two vertices.

Subgraph: Graph $\{V', E'\}$ is a subgraph of graph $\{V, E\}$ if $V' \subset V$ with $E' \subset E$ *and* incident on V'.

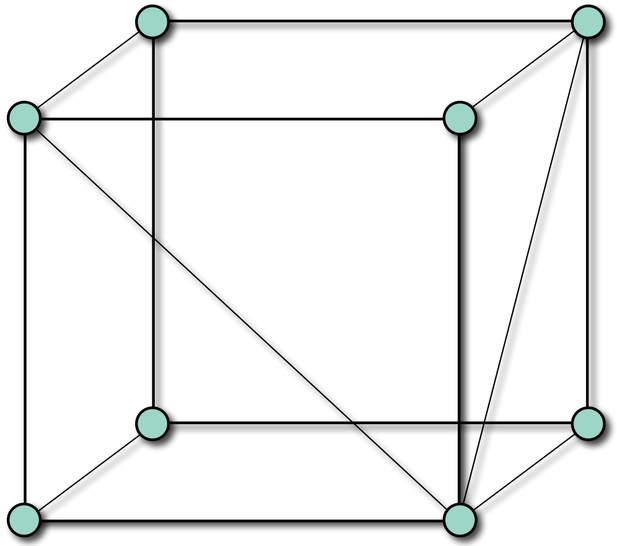
Connected component:
Maximally connected subgraph.

Graph Embedding

Embedding: Graph is embedded in $\mathbb{R}^d$, if each vertex is assigned a position in $\mathbb{R}^d$.



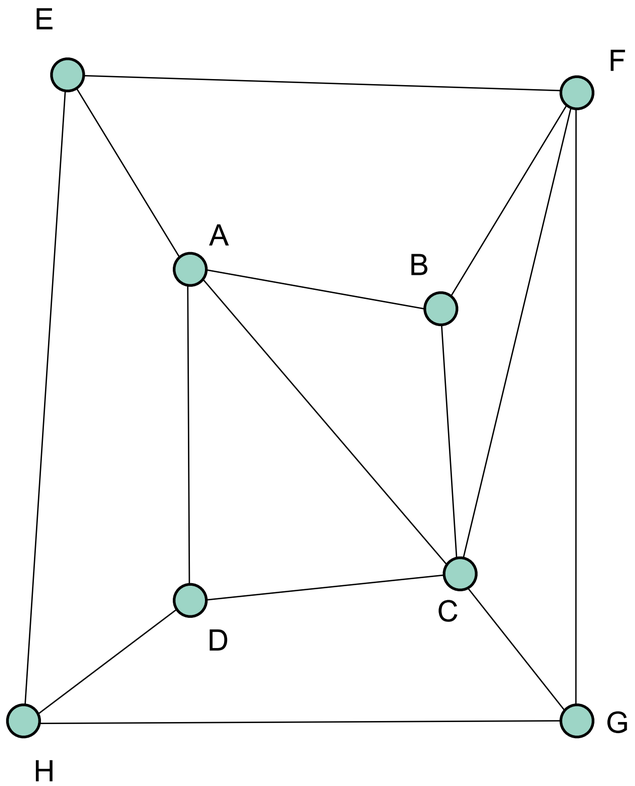
Embedded in $\mathbb{R}^2$



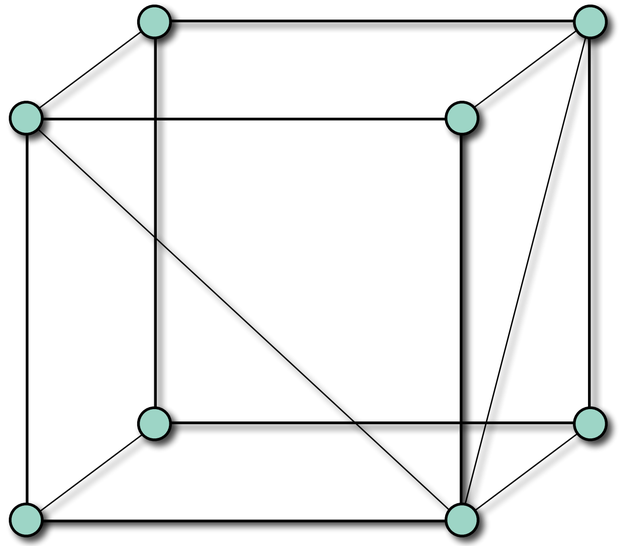
Embedded in $\mathbb{R}^3$

Graph Embedding

Embedding: Graph is embedded in $\mathbb{R}^d$, if each vertex is assigned a position in $\mathbb{R}^d$.



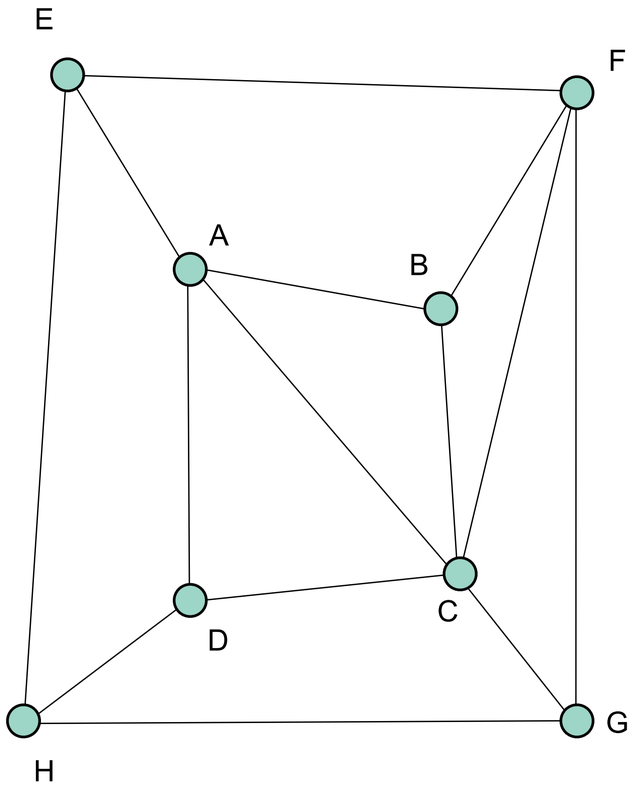
Embedded in $\mathbb{R}^2$



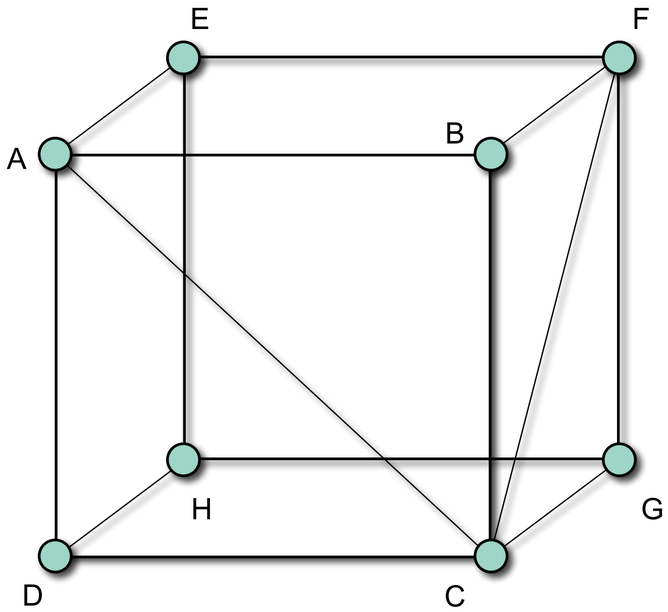
Embedded in $\mathbb{R}^3$

Graph Embedding

Embedding: Graph is embedded in $\mathbb{R}^d$, if each vertex is assigned a position in $\mathbb{R}^d$.



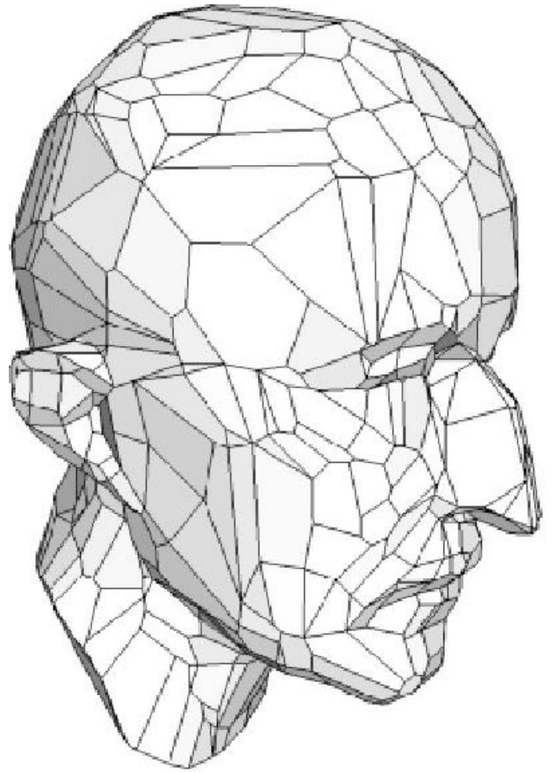
Embedded in $\mathbb{R}^2$



Embedded in $\mathbb{R}^3$

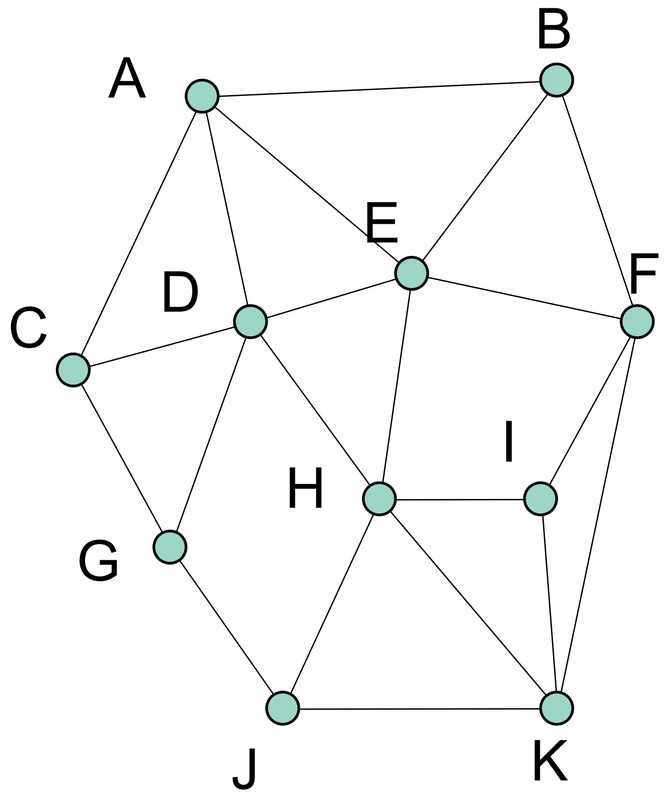
Graph Embedding

Embedding: Graph is embedded in $\mathbb{R}^d$, if each vertex is assigned a position in $\mathbb{R}^d$.

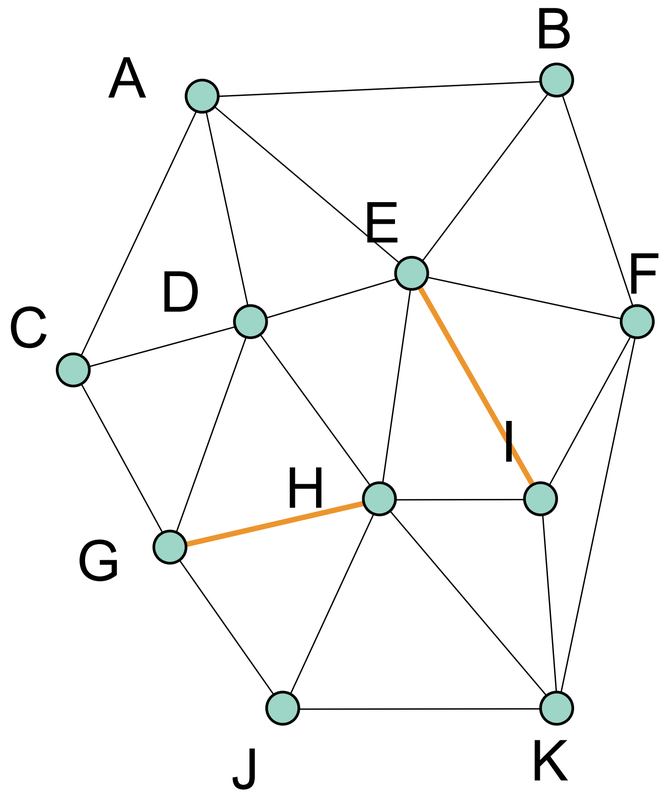


Embedded in $\mathbb{R}^3$

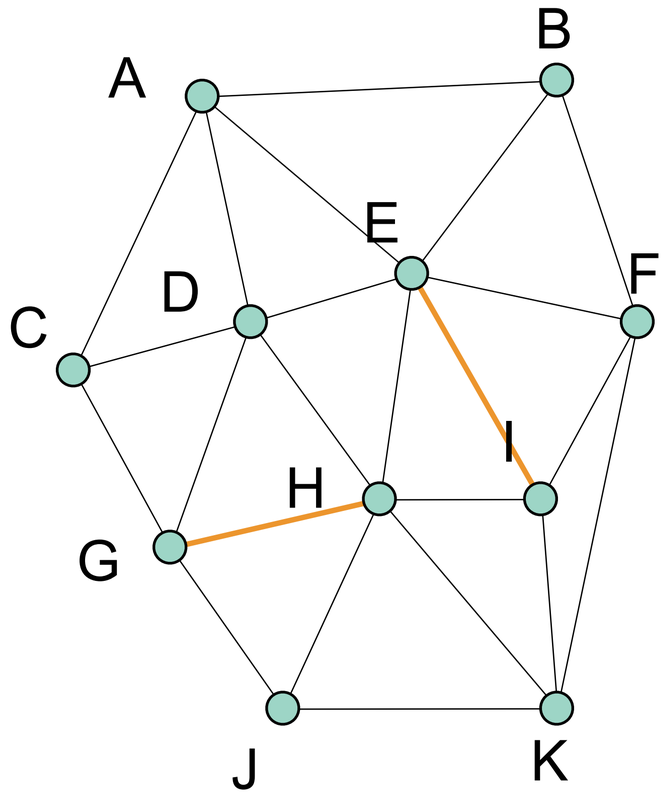
Triangulation



Triangulation

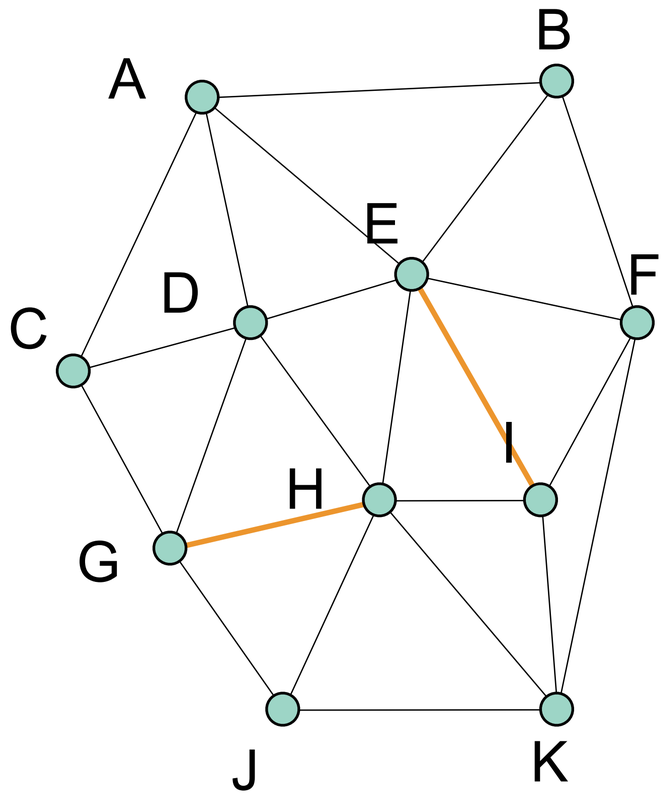


Triangulation



Triangulation: Graph where every face is a triangle.

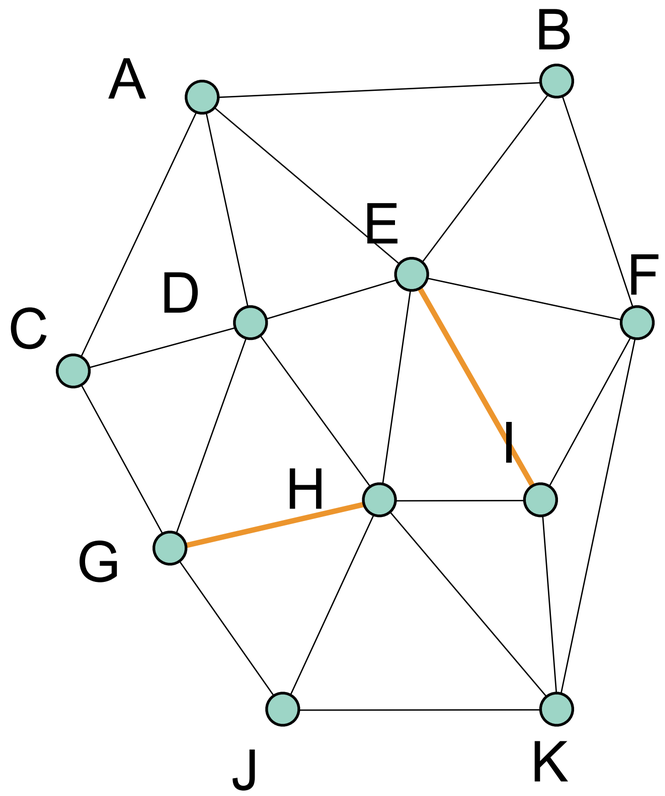
Triangulation



Triangulation: Graph where every face is a triangle.

Why...?

Triangulation

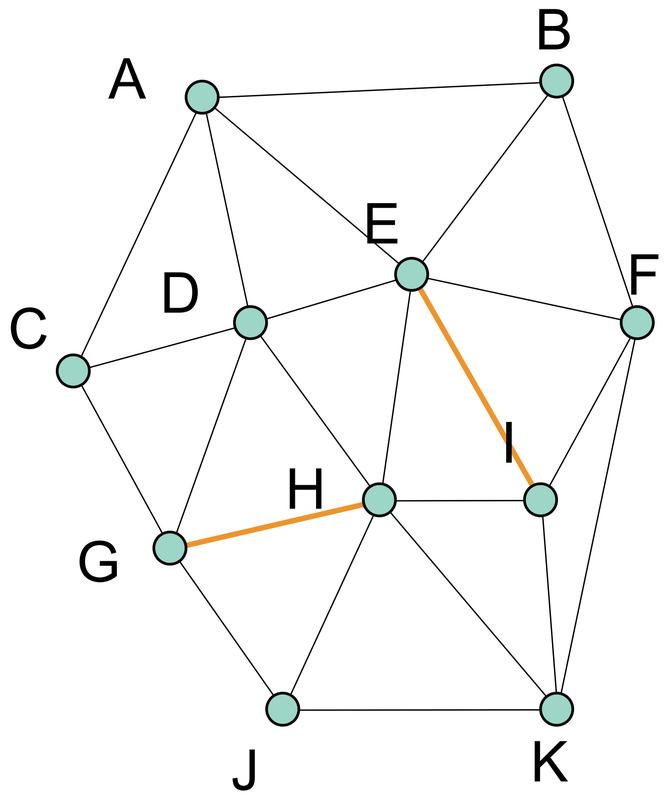


Triangulation: Graph where every face is a triangle.

Why...?

➡ simplifies data structures

Triangulation

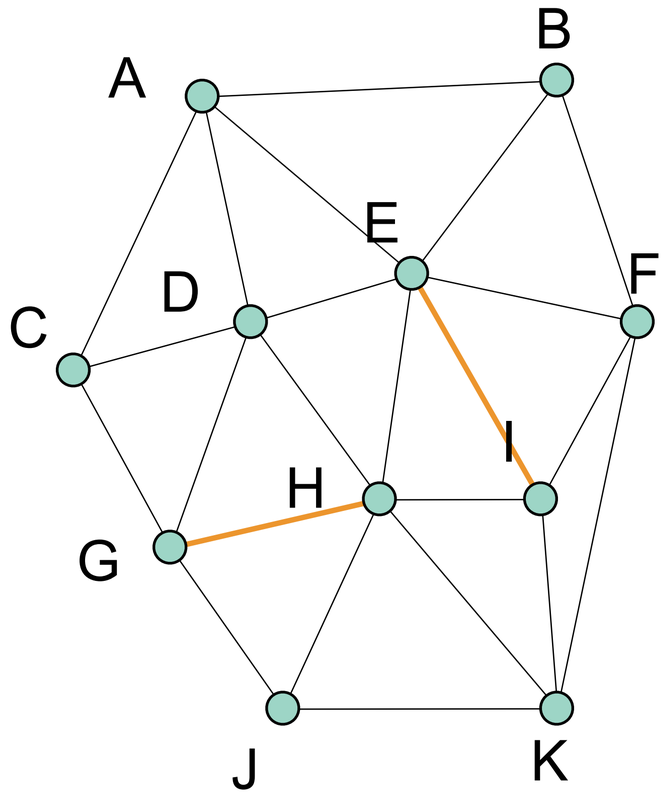


Triangulation: Graph where every face is a triangle.

Why...?

- ➡ simplifies data structures
- ➡ simplifies rendering

Triangulation

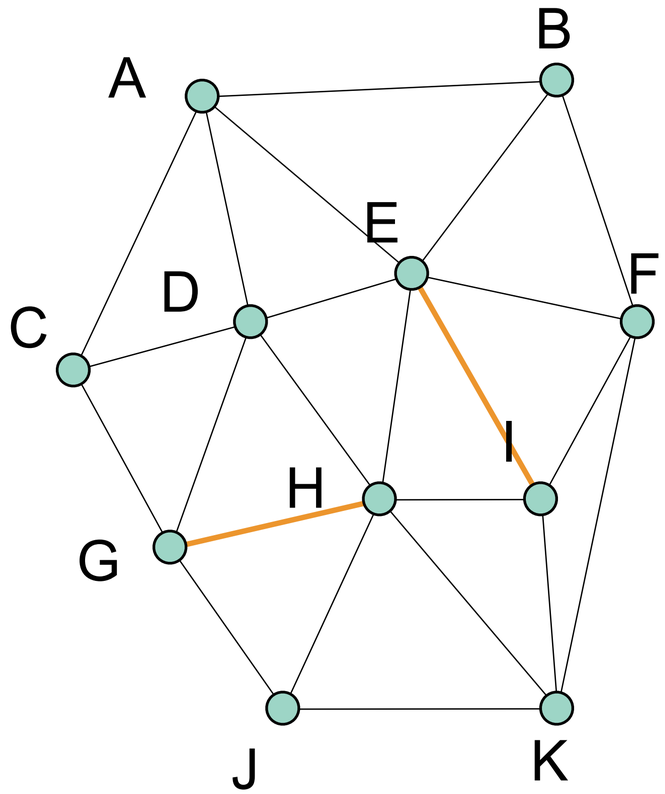


Triangulation: Graph where every face is a triangle.

Why...?

- ➡ simplifies data structures
- ➡ simplifies rendering
- ➡ simplifies algorithms

Triangulation

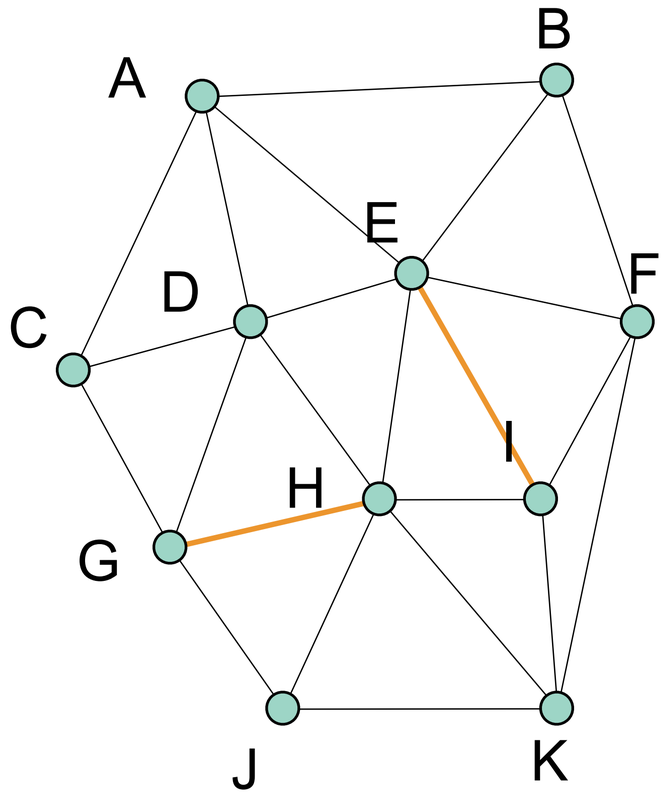


Triangulation: Graph where every face is a triangle.

Why...?

- ➡ simplifies data structures
- ➡ simplifies rendering
- ➡ simplifies algorithms
- ➡ by definition, triangle is **planar**

Triangulation



Triangulation: Graph where every face is a triangle.

Why...?

- ➡ simplifies data structures
- ➡ simplifies rendering
- ➡ simplifies algorithms
- ➡ by definition, triangle is **planar**
- ➡ any polygon can be triangulated (not necessarily uniquely)

Outline

- Parametric Approximations
- Polygon Meshes
- **Data structures**

Mesh Data Structures

- How to store **geometry** & **connectivity (topology)**?
- Compact storage
  - File formats
- Efficient algorithms on meshes
  - Identify time-critical operations
  - All vertices/edges of a face
  - All **incident/neighboring** vertices/edges/faces of a vertex

Data Structures

Data Structures

- What should be stored?

Data Structures

- What should be stored?
  - **Geometry**: 3D coordinates

Data Structures

- What should be stored?
  - **Geometry**: 3D coordinates
  - Attributes

Data Structures

- What should be stored?
  - **Geometry**: 3D coordinates
  - Attributes
    - e.g., normal, color, texture coordinate

Data Structures

- What should be stored?
  - **Geometry**: 3D coordinates
  - Attributes
    - e.g., normal, color, texture coordinate
    - Per vertex, per face, per edge

Data Structures

- What should be stored?
  - **Geometry**: 3D coordinates
  - Attributes
    - e.g., normal, color, texture coordinate
    - Per vertex, per face, per edge
  - **Connectivity (topology)**

Data Structures

- What should be stored?
  - **Geometry**: 3D coordinates
  - Attributes
    - e.g., normal, color, texture coordinate
    - Per vertex, per face, per edge
  - **Connectivity (topology)**
    - What is adjacent to what

Data Structures

Data Structures

- What should it support?

Data Structures

- What should it support?
  - Rendering

Data Structures

- What should it support?
  - Rendering
  - Queries

Data Structures

- What should it support?
  - Rendering
  - Queries
    - What are the vertices of the face #37?

Data Structures

- What should it support?
  - Rendering
  - Queries
    - What are the vertices of the face #37?
    - Is vertex #6 adjacent to vertex #12?

Data Structures

- What should it support?
  - Rendering
  - Queries
    - What are the vertices of the face #37?
    - Is vertex #6 adjacent to vertex #12?
    - Which faces are adjacent to face #7?

Data Structures

- What should it support?
  - Rendering
  - Queries
    - What are the vertices of the face #37?
    - Is vertex #6 adjacent to vertex #12?
    - Which faces are adjacent to face #7?
  - Modifications

Data Structures

- What should it support?
  - Rendering
  - Queries
    - What are the vertices of the face #37?
    - Is vertex #6 adjacent to vertex #12?
    - Which faces are adjacent to face #7?
  - Modifications
    - Remove/add a vertex/face

Data Structures

- What should it support?
  - Rendering
  - Queries
    - What are the vertices of the face #37?
    - Is vertex #6 adjacent to vertex #12?
    - Which faces are adjacent to face #7?
  - Modifications
    - Remove/add a vertex/face
    - Vertex split, edge collapse

Data Structures

Data Structures

- Is it a good data structure?

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)
  - Time to answer a query

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)
  - Time to answer a query
  - Time to perform an operation

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)
  - Time to answer a query
  - Time to perform an operation
  - Space complexity

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)
  - Time to answer a query
  - Time to perform an operation
  - Space complexity
  - Redundancy

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)
  - Time to answer a query
  - Time to perform an operation
  - Space complexity
  - Redundancy
  - Time to update changes in geometry and/or topology

Data Structures

- Is it a good data structure?
  - Time to construct (preprocessing)
  - Time to answer a query
  - Time to perform an operation
  - Space complexity
  - Redundancy
  - Time to update changes in geometry and/or topology
  - *Is this differentiable?*

Face Set (STL)

- `Face list`:
  - 3 positions

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

Face Set (STL)

- `Face list:`
  - 3 positions

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

36 B/f = 72 B/v

no connectivity!

Shared Vertex (OBJ, OFF)

- `Indexed Face List`
  - Vertex: position
  - Face: vertex indices

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

Triangles
i_{11} i_{12} i_{13}
...
...
...
...
i_{F1} i_{F2} i_{F3}

Shared Vertex (OBJ, OFF)

- `Indexed Face List`
  - Vertex: position
  - Face: vertex indices

Vertices
$x_1\ y_1\ z_1$
...
$x_V\ y_V\ z_V$

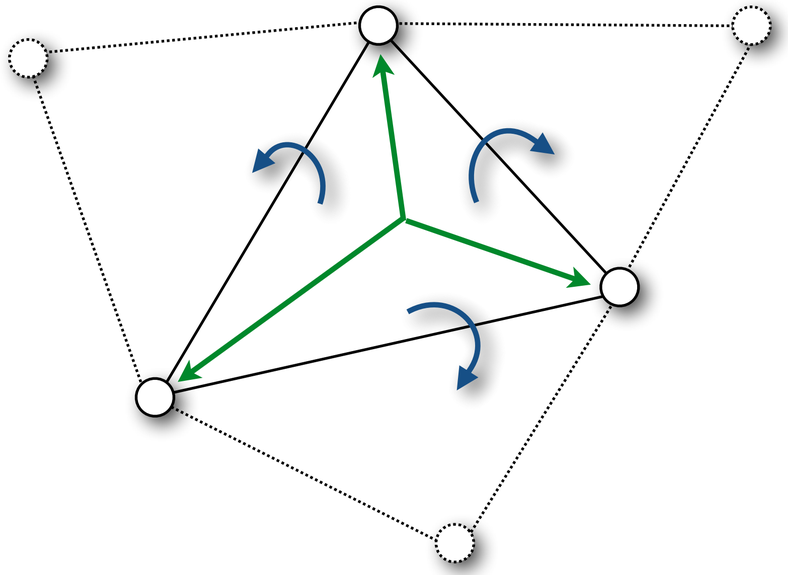
Triangles
$i_{11}\ i_{12}\ i_{13}$
...
...
...
...
$i_{F1}\ i_{F2}\ i_{F3}$

12 B/v + 12 B/f = 36 B/v

no explicit neighborhood information

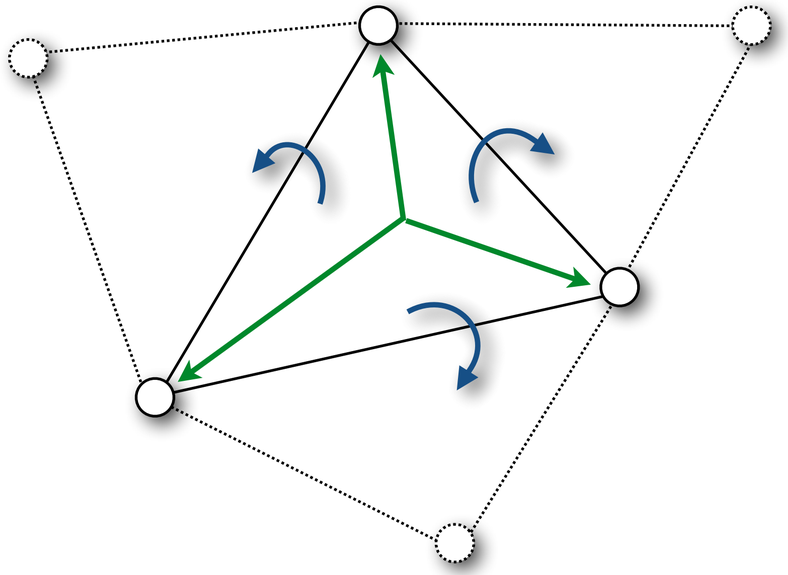
Face-Based Connectivity

- Vertex:
  - position
  - 1 face
- Face:
  - 3 vertices
  - 3 face neighbors



Face-Based Connectivity

- Vertex:
  - position
  - 1 face
- Face:
  - 3 vertices
  - 3 face neighbors

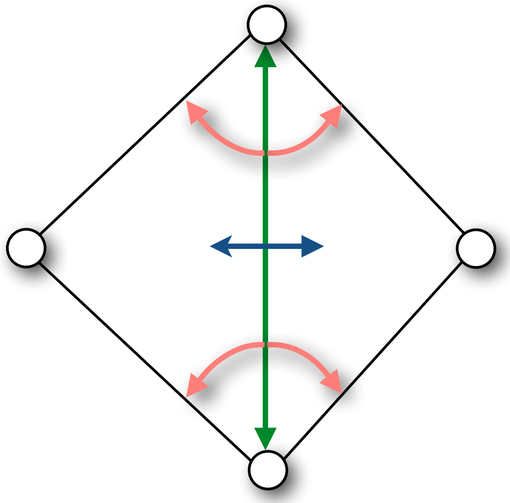


64 B/v

no edges!

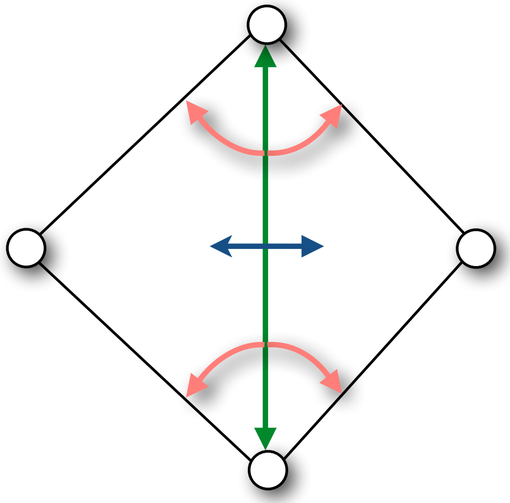
Edge-Based Connectivity

- Vertex
  - position
  - 1 edge
- Edge
  - 2 vertices
  - 2 faces
  - 4 edges
- Face
  - 1 edge



Edge-Based Connectivity

- Vertex
  - position
  - 1 edge
- Edge
  - 2 vertices
  - 2 faces
  - 4 edges
- Face
  - 1 edge

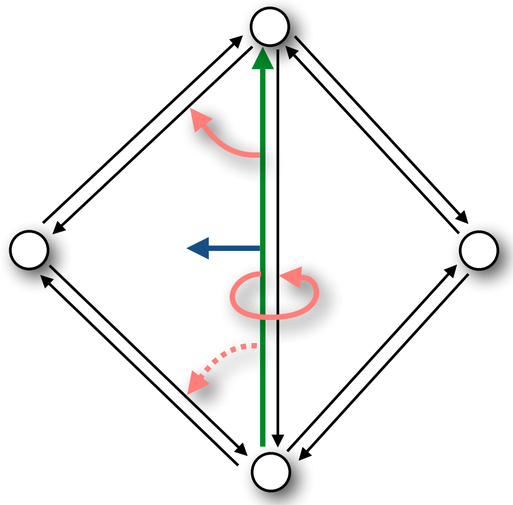


120 B/v

edge orientation?

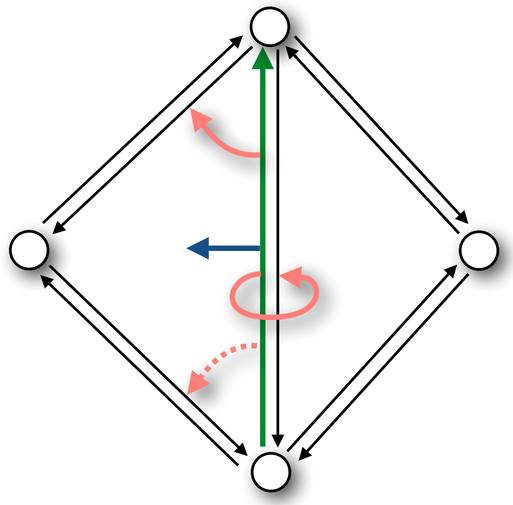
Halfedge-Based Connectivity

- Vertex
  - position
  - 1 halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- Face
  - 1 halfedge



Halfedge-Based Connectivity

- Vertex
  - position
  - 1 halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- Face
  - 1 halfedge

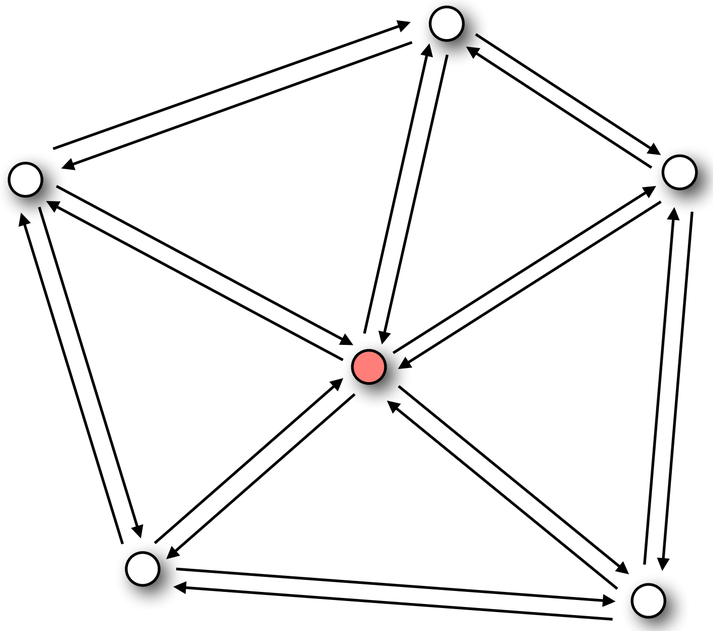


96 to 144 B/v

no case distinctions
during traversal

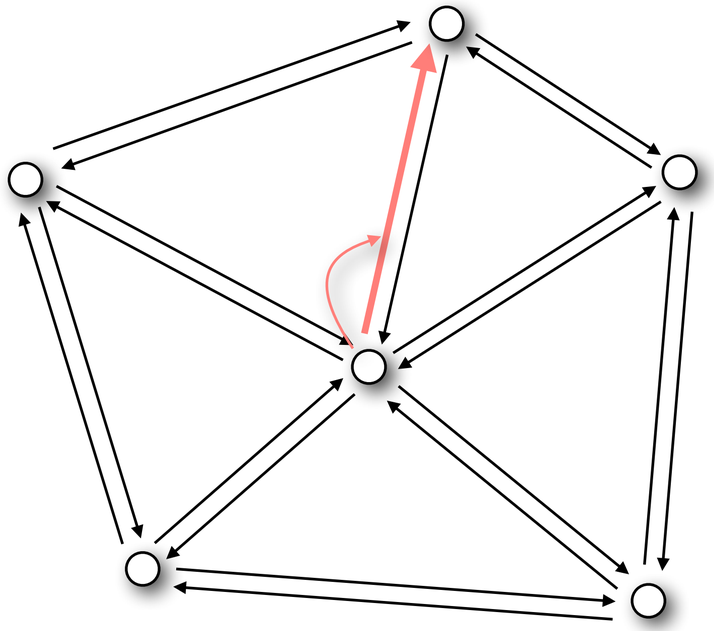
One-Ring Traversal

1. Start at vertex



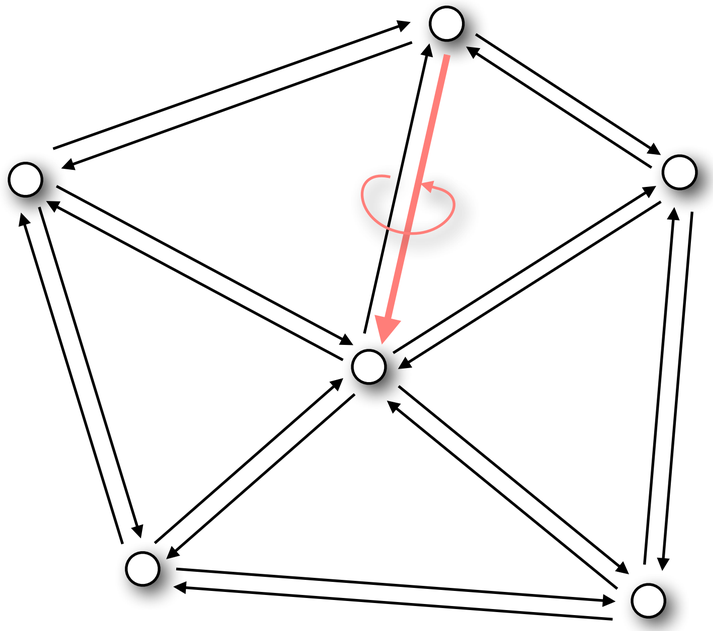
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



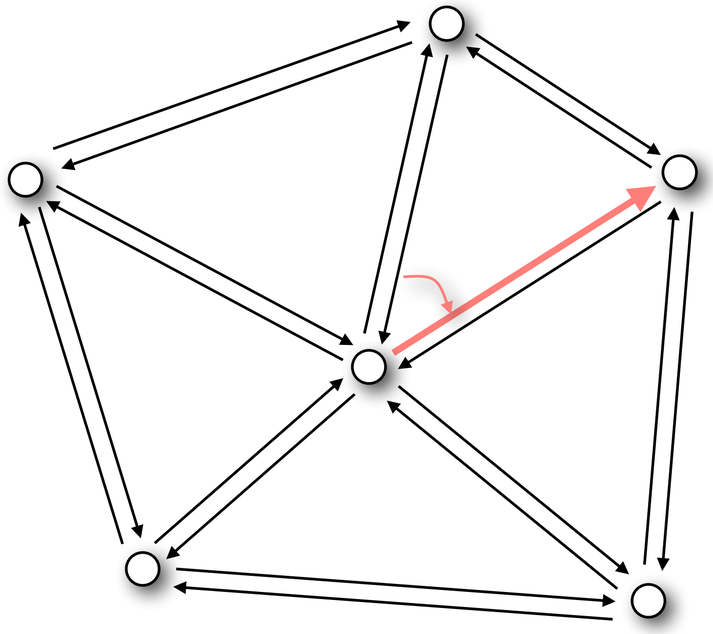
One-Ring Traversal

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



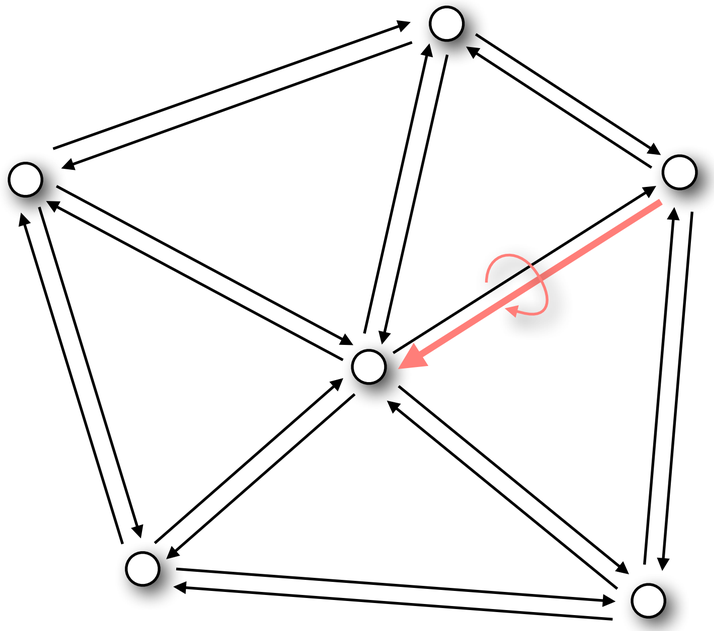
One-Ring Traversal

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



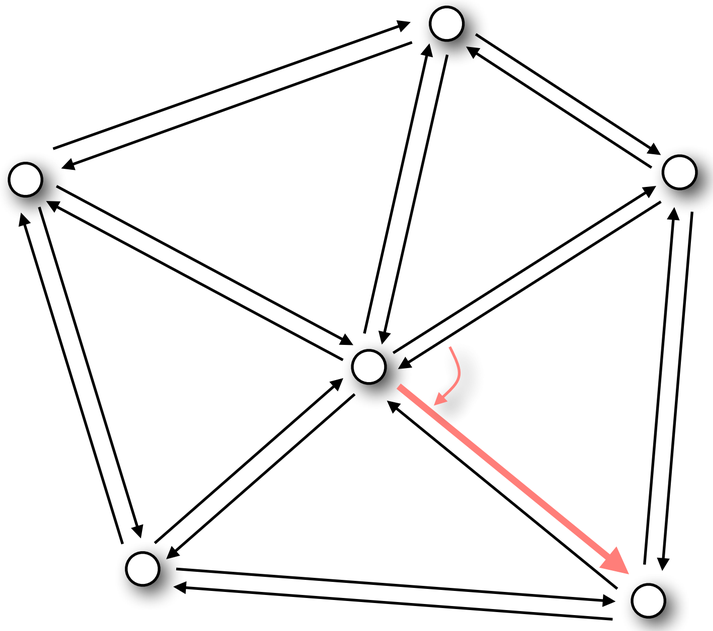
One-Ring Traversal

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. ...

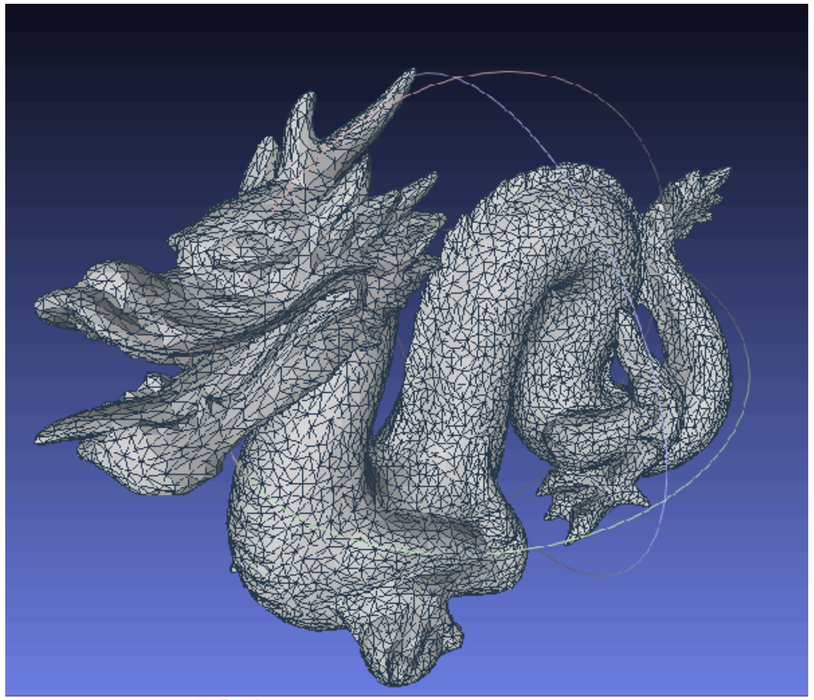


Halfedge-Based Libraries

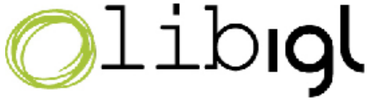
- CGAL
  - www.cgal.org
  - Computational geometry
  - Free for non-commercial use

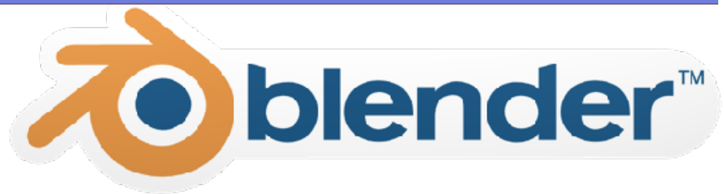
- OpenMesh
  - www.openmesh.org
  - Mesh processing
  - Free, LGPL licence

Looking at Meshes



OpenMesh





Literature

- Book #1: Chapters 2,3;
- Book #2: Chapters 1,2
- Kettner, *Using generic programming for designing a data structure for polyhedral surfaces*, Symp. on Comp. Geom., 1998
- Campagna et al., *Directed Edges - A Scalable Representation for Triangle Meshes*, Journal of Graphics Tools 4(3), 1998
- Botsch et al., *OpenMesh - A generic and efficient polygon mesh data structure*, OpenSG Symp. 2002