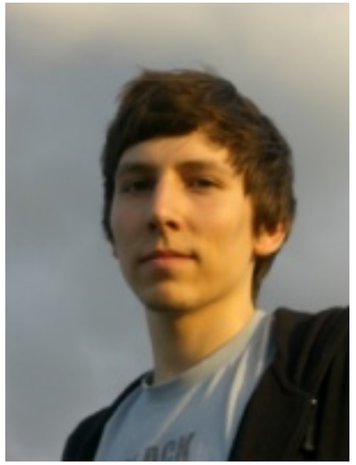
Geometric Computing with Python

Sebastian Koch, **Teseo** Schneider, Francis Williams,
Chencheng Li, **Daniele** Panozzo

https://geometryprocessing.github.io/geometric-computing-python/

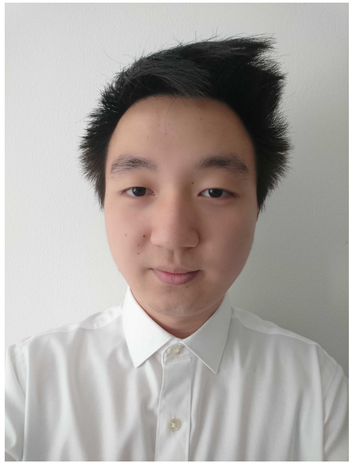
Who are we?



Sebastian Koch

Teseo Schneider

Francis Williams



ChenCheng Li

Daniele Panozzo

Course Goals

- Learn how to design, program, and analyze algorithms for **geometric computing**
- Hands-on experience with shape modeling and geometry processing algorithms
- Learn how to batch process large collections of geometric data and integrate it in deep learning pipelines

Geometric Computing

Geometric Computing

Discrete Differential Geometry

- Surface and volumes representation
- Differential properties and operators

Geometric Computing

Discrete Differential Geometry

- Surface and volumes representation
- Differential properties and operators

Numerical Method for PDEs

- Focus on real-time approximations
- Irregular domains

Geometric Computing

Discrete Differential Geometry

- Surface and volumes representation
- Differential properties and operators

High Performance Computing

- Vectorized computation
- Multi-core and distributed computation
- GPU accelerators

Numerical Method for PDEs

- Focus on real-time approximations
- Irregular domains

Geometric Computing

Discrete Differential Geometry

- Surface and volumes representation
- Differential properties and operators

High Performance Computing

- Vectorized computation
- Multi-core and distributed computation
- GPU accelerators

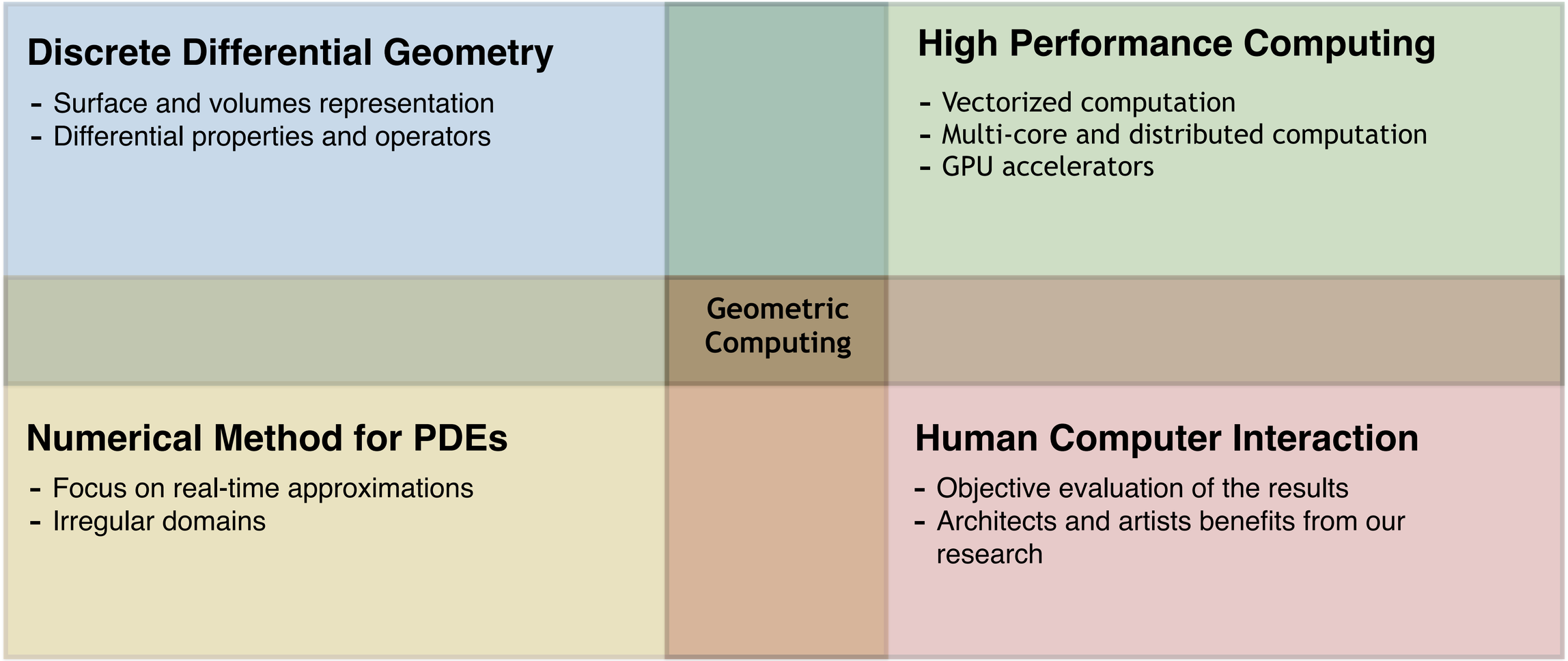
Numerical Method for PDEs

- Focus on real-time approximations
- Irregular domains

Human Computer Interaction

- Objective evaluation of the results
- Architects and artists benefits from our research

Geometric Computing



Geometric Computing

Big Data

Discrete Differential Geometry

- Surface and volumes representation
- Differential properties and operators

High Performance Computing

- Vectorized computation
- Multi-core and distributed computation
- GPU accelerators

Geometric Computing

Numerical Method for PDEs

- Focus on real-time approximations
- Irregular domains

Human Computer Interaction

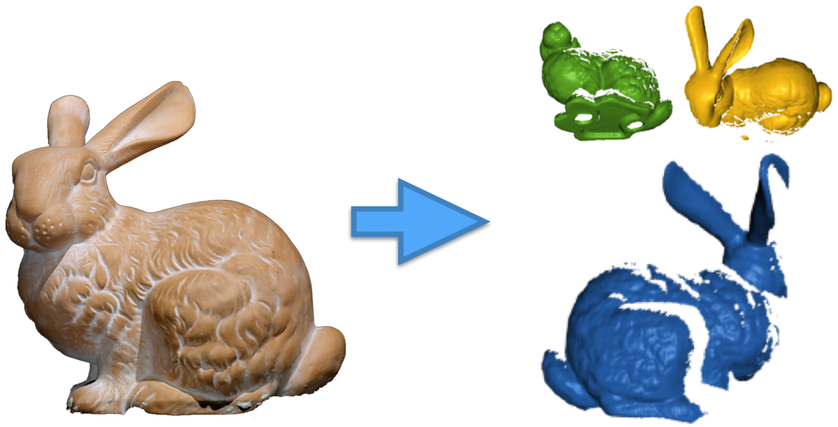
- Objective evaluation of the results
- Architects and artists benefits from our research

Geometric Computing

Geometric Computing

Physical Object

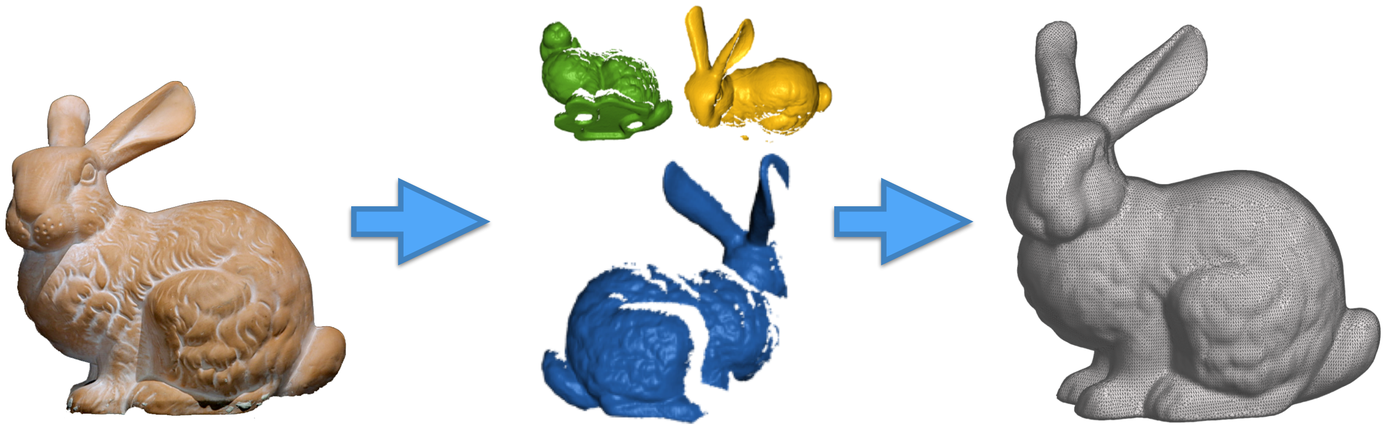
Geometric Computing



Physical Object

Range Images

Geometric Computing

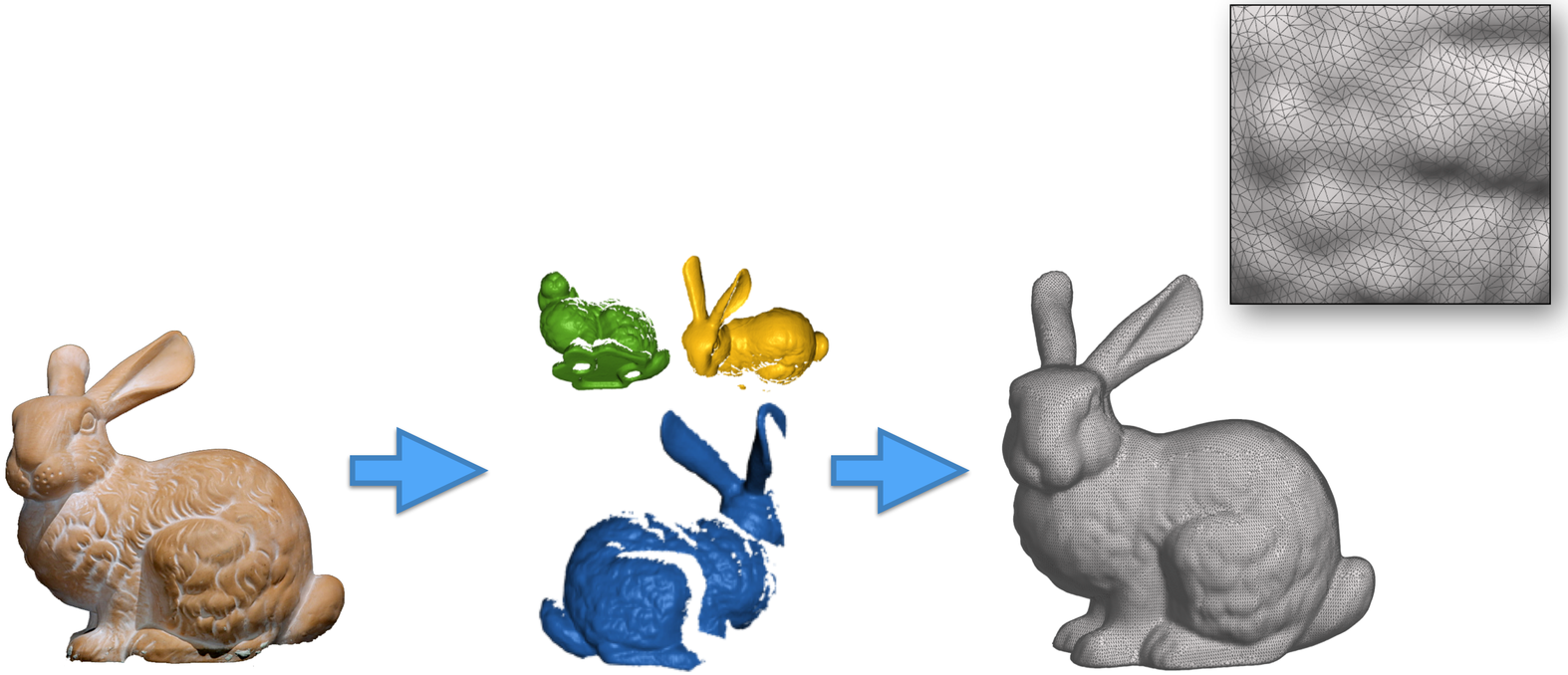


Physical Object

Range Images

Unstructured Model

Geometric Computing

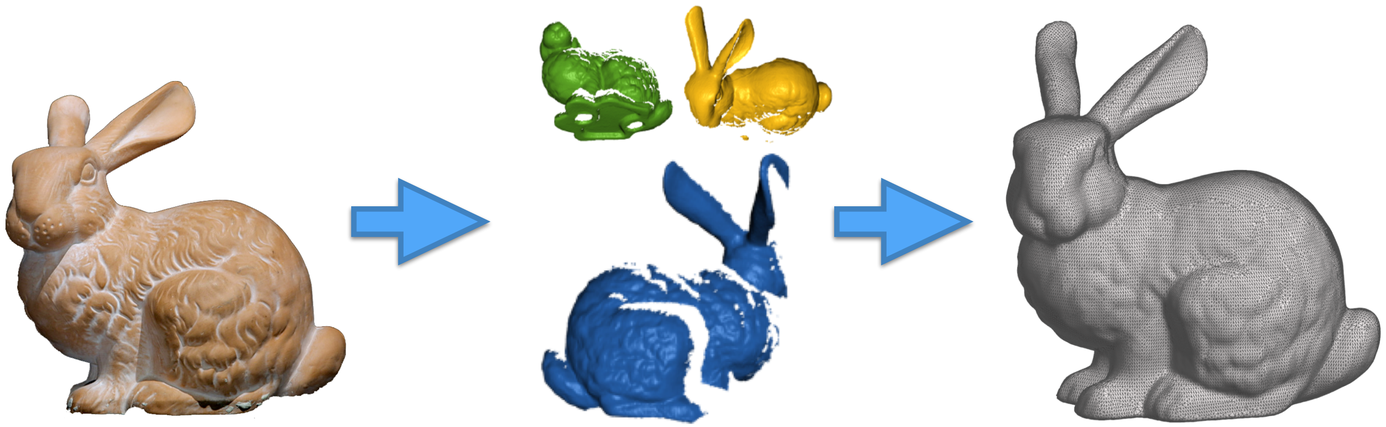


Physical Object

Range Images

Unstructured Model

Geometric Computing

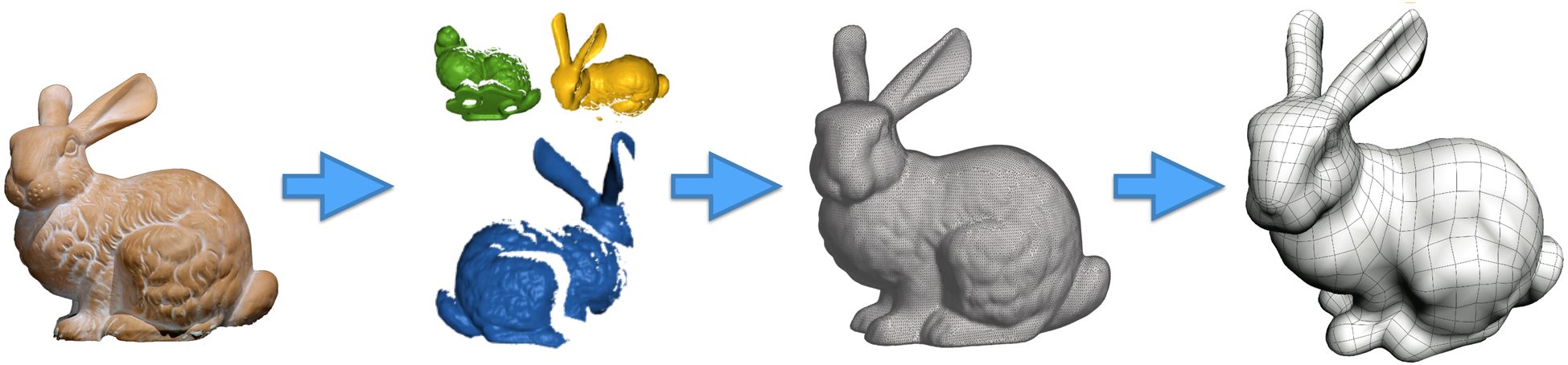


Physical Object

Range Images

Unstructured Model

Geometric Computing



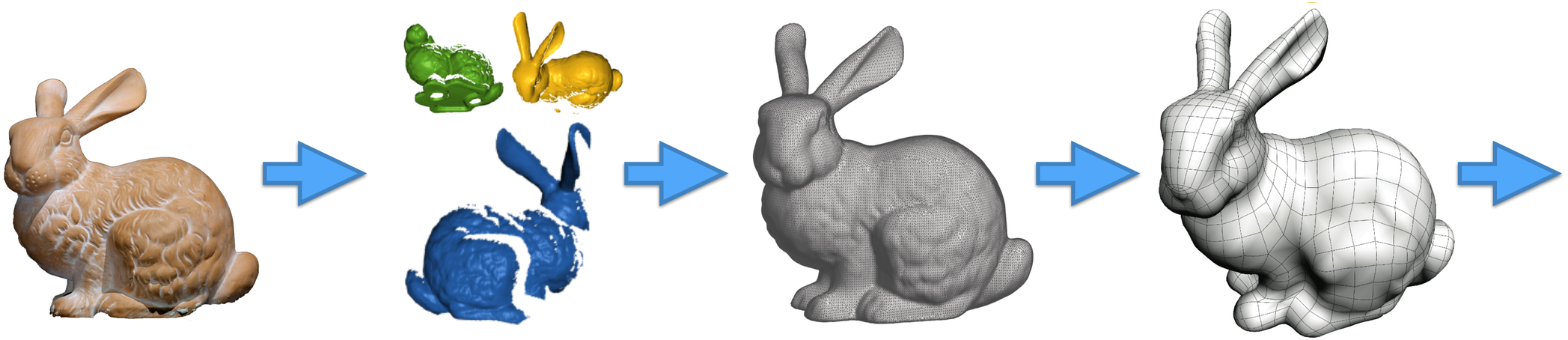
Physical Object

Range Images

Unstructured Model

Structured Model

Geometric Computing



Physical Object

Range Images

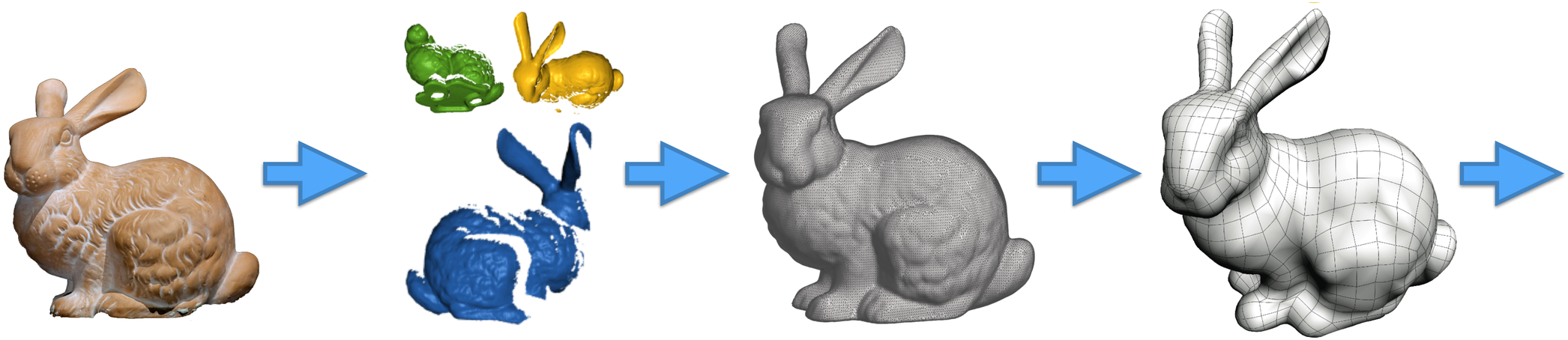
Unstructured Model

Structured Model

Applications

Geometric Computing

Animation



Physical Object

Range Images

Unstructured Model

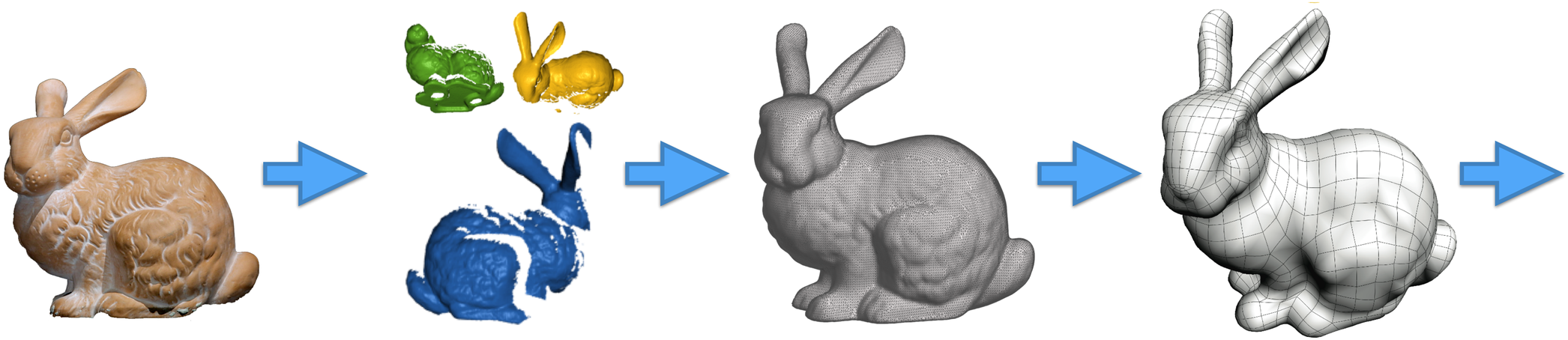
Structured Model

Applications

Geometric Computing

Animation

Physical Simulation



Physical Object

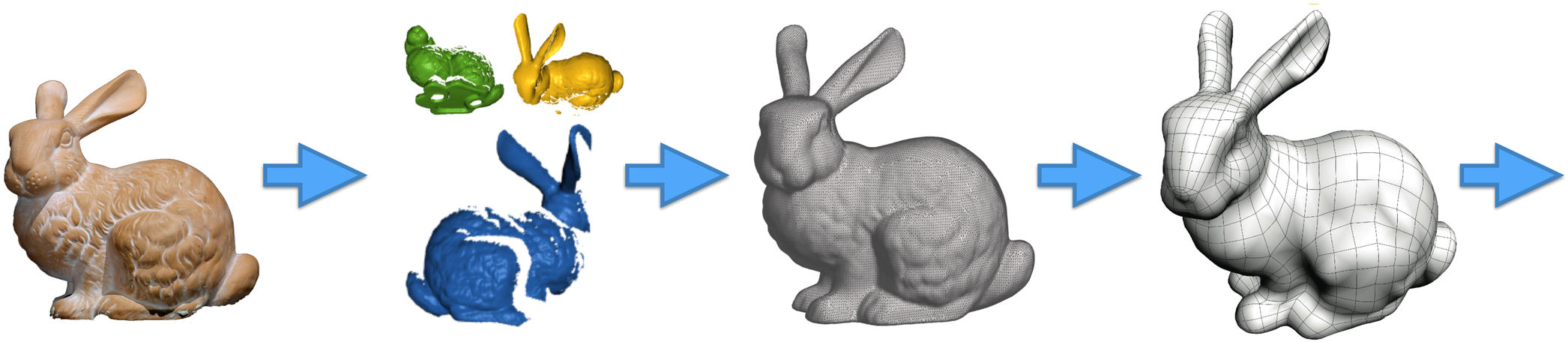
Range Images

Unstructured Model

Structured Model

Applications

Geometric Computing



Physical Object

Range Images

Unstructured Model

Structured Model

Animation

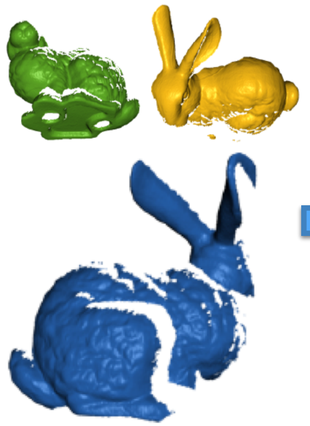
Physical Simulation

Fabrication

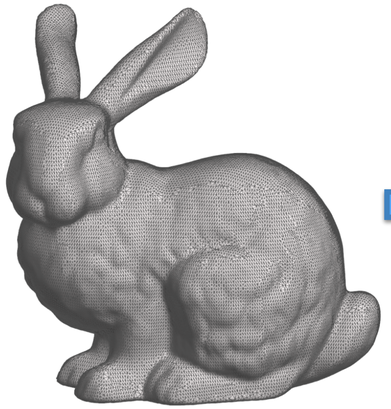
Applications

Geometric Computing

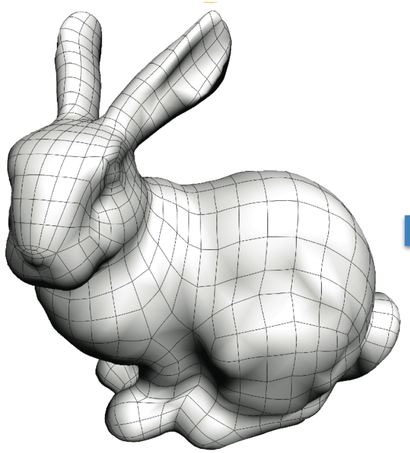
Physical Object



Range Images



Unstructured Model



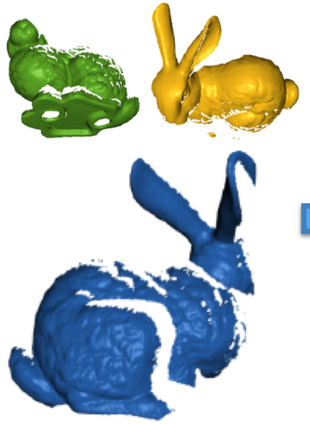
Structured Model

Applications

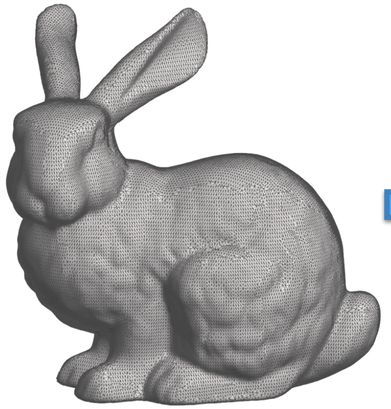
- **Animation**
- **Physical Simulation**
- **Fabrication**
- **Visual Effects**

Geometric Computing

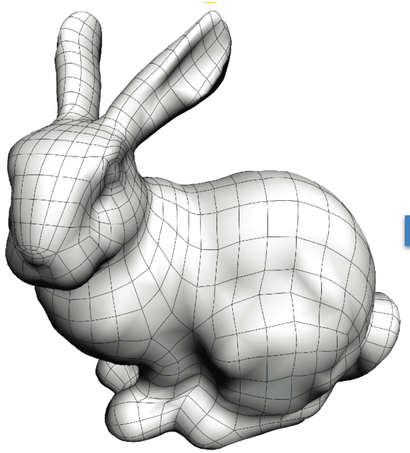
Physical Object



Range Images



Unstructured Model



Structured Model

Animation

Physical Simulation

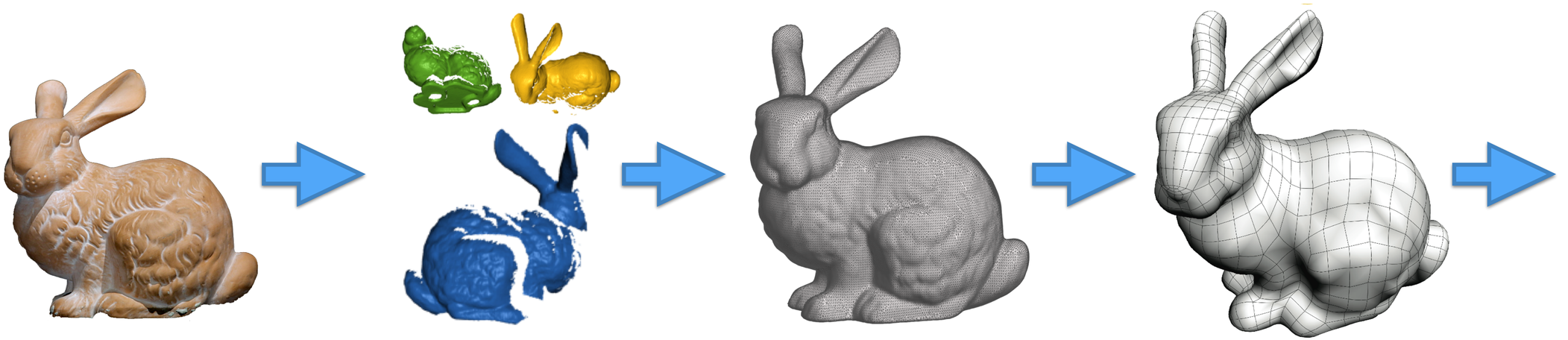
Fabrication

Visual Effects

Structural Analysis

Applications

Geometric Computing



Physical Object

Range Images

Unstructured Model

Structured Model

Animation

Physical Simulation

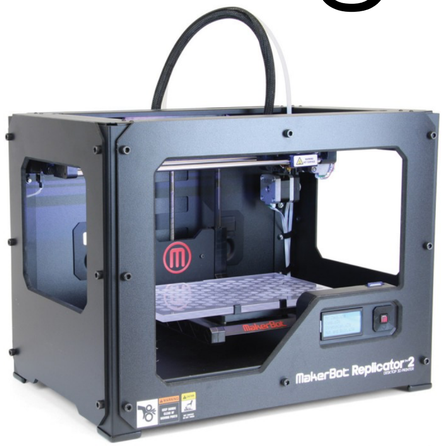
Fabrication

Visual Effects

Structural Analysis

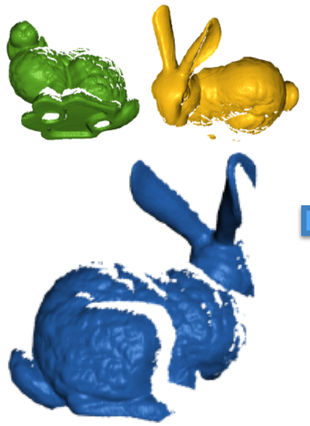
Applications

Geometric Computing

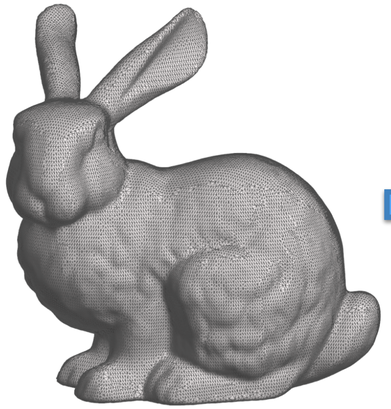


3D Printer

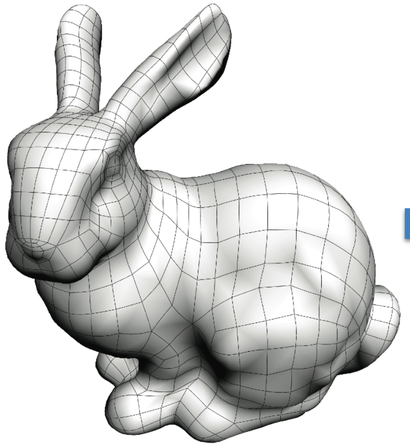
Physical Object



Range Images



Unstructured Model



Structured Model

Animation

Physical Simulation

Fabrication

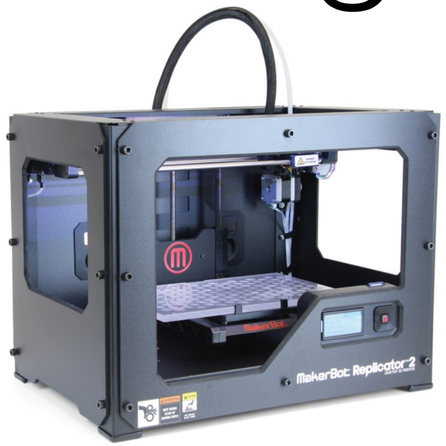
Visual Effects

Structural Analysis

Applications

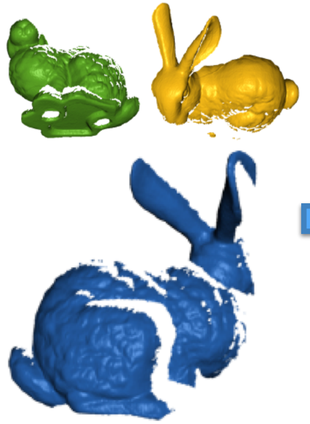
Geometric Computing

Physical Derived Object

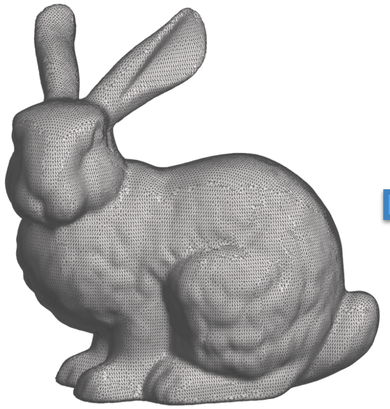


3D Printer

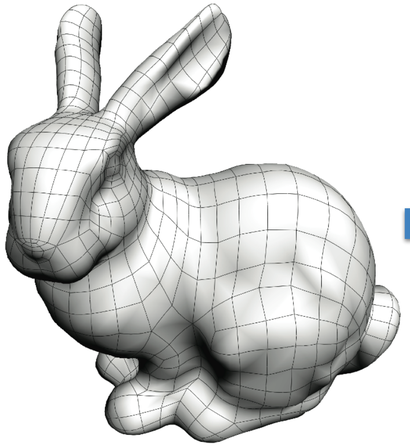
Physical Object



Range Images



Unstructured Model



Structured Model

Animation

Physical Simulation

Fabrication

Visual Effects

Structural Analysis

Applications

Course Overview

- **Daniele:** Introduction to Geometric Computing with Jupyter
- **Sebastian:** Geometric Computing and Geometric Deep Learning
- **Teseo:** Mesh Generation and Numerical Simulation
- Q&A

Getting Started

- The libraries used in this course are implemented in C++ for efficiency reasons, but are exposed to python for ease of integration

- All libraries are available on conda, they can be installed with:

```
conda config --add channels conda-forge

conda install meshplot
conda install igl
conda install wildmeshing
conda install polyfempy
```

Libraries Overview

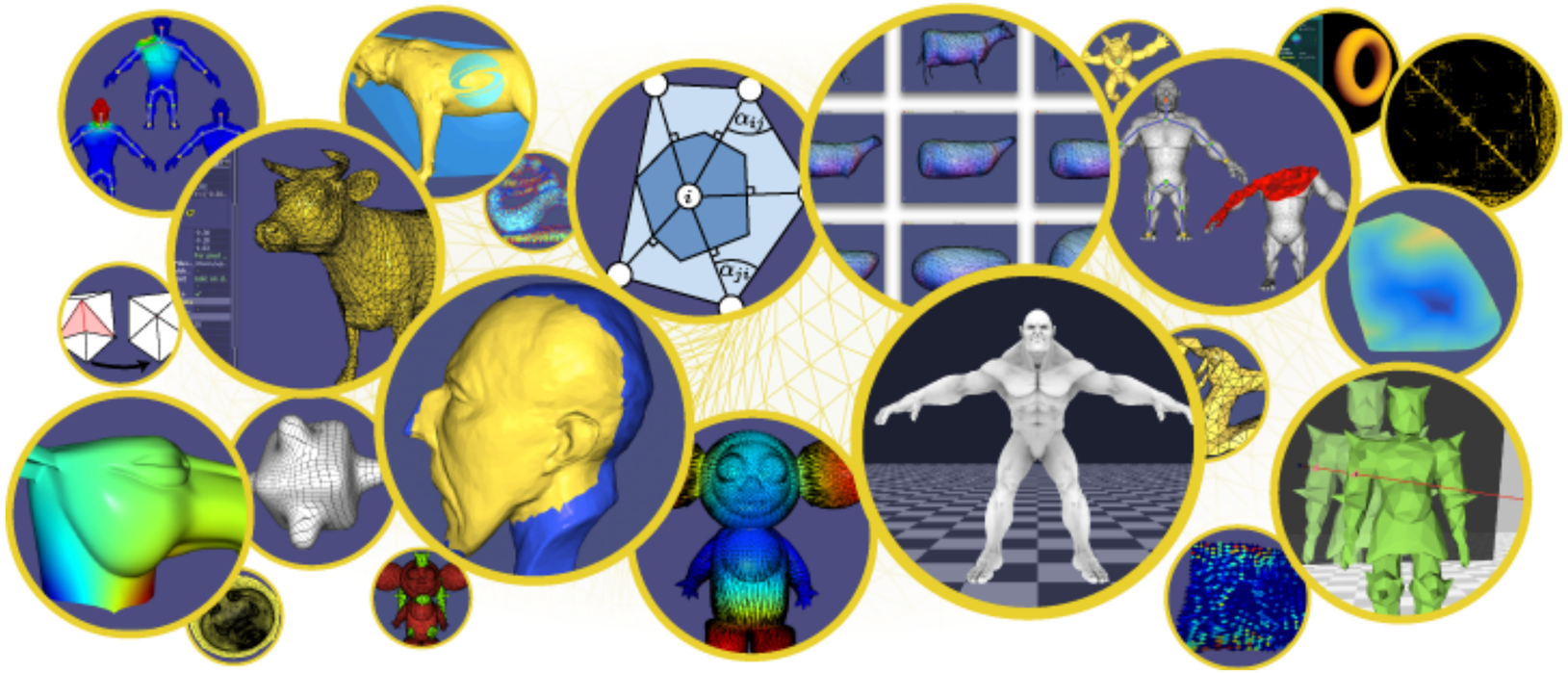
Cross Platform: Windows, MacOSX, Linux

MeshPlot

```
mp.plot(v, f)
```

https://skoch9.github.io/meshplot/

Interactive Geometry Library (libigl)



https://libigl.github.io

Wild Meshing (TetWild)

https://wildmeshing.github.io

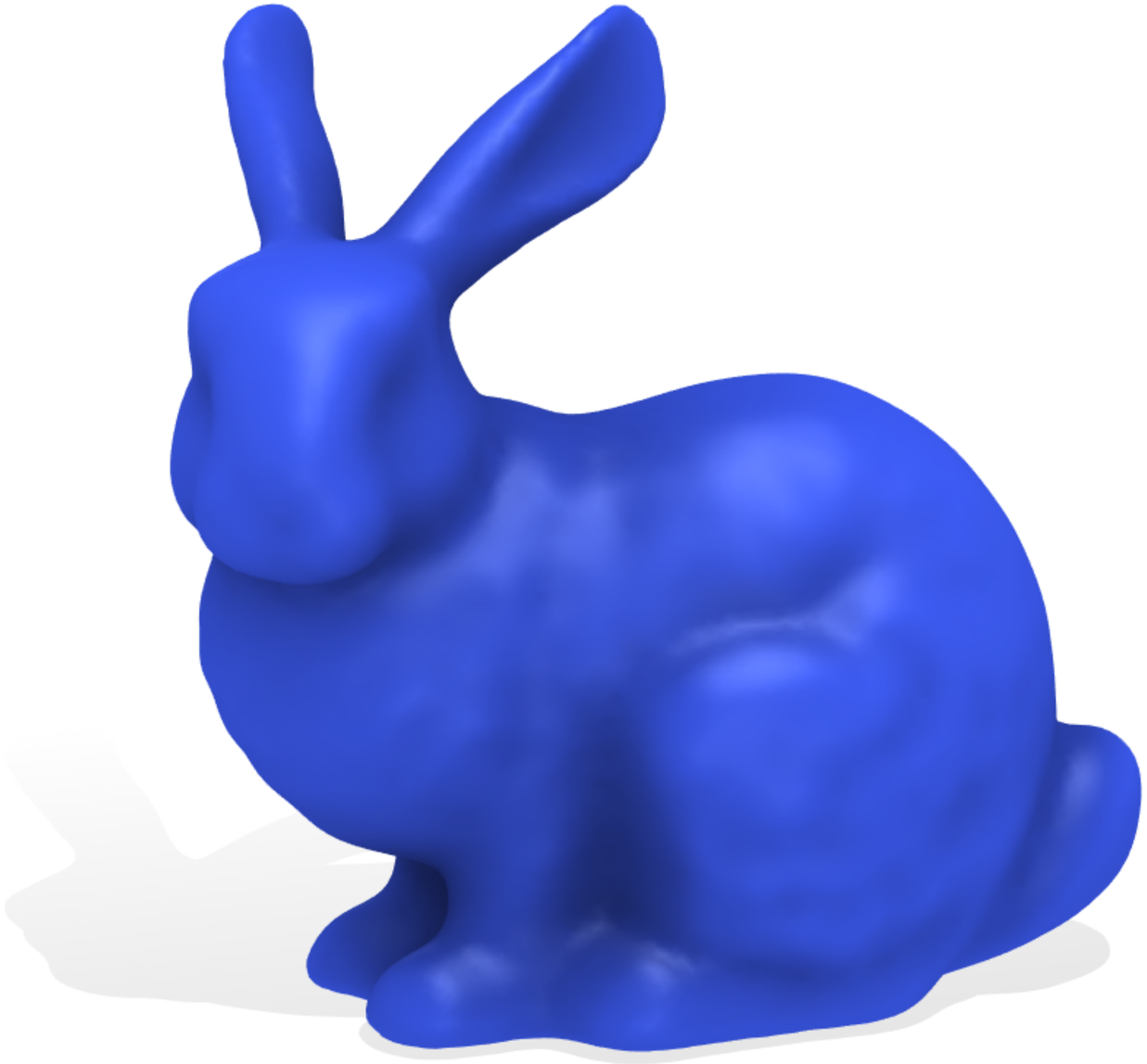
PolyFEM

https://polyfem.github.io

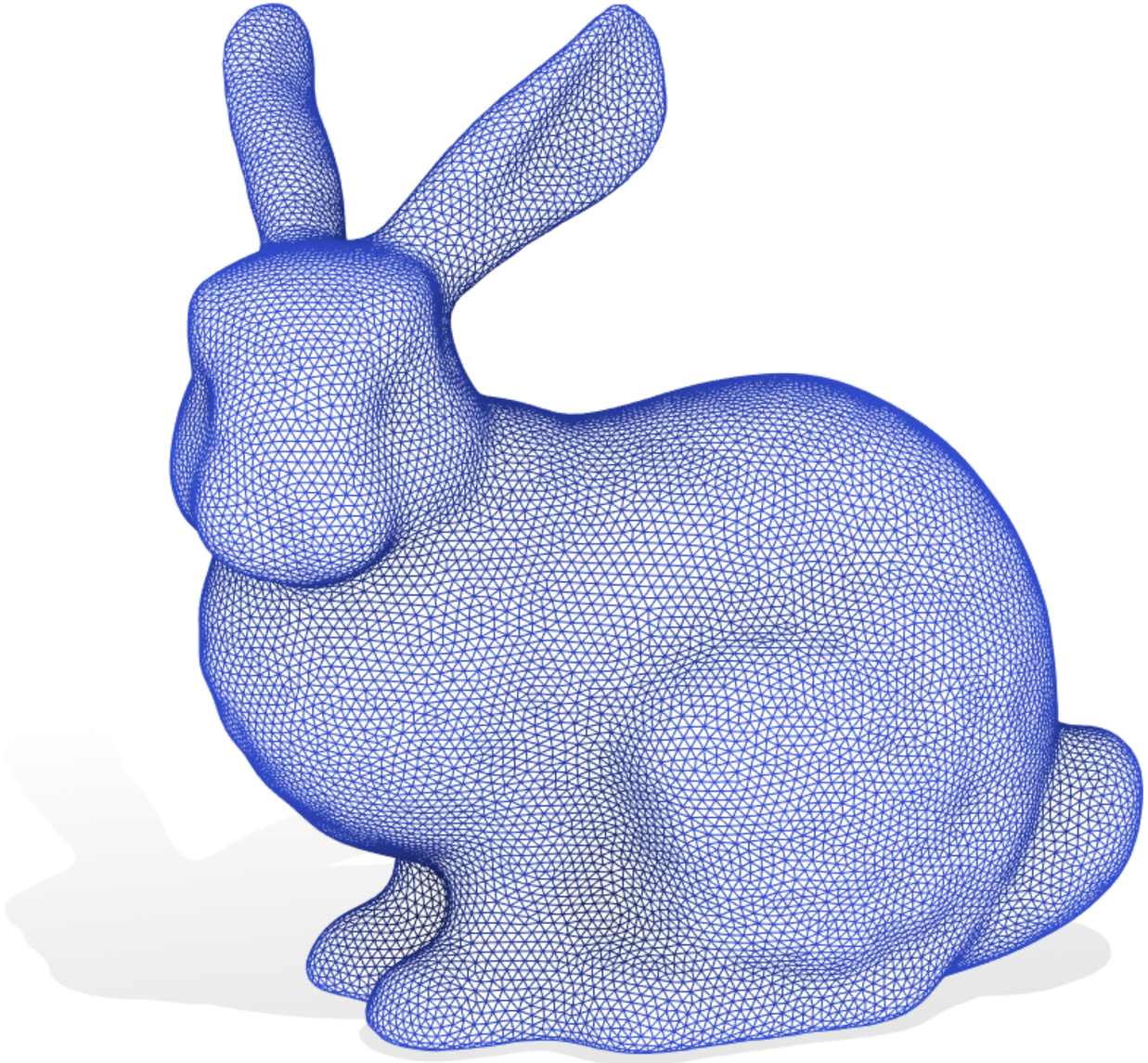
Data Structures (or lack thereof)

Acknowledgement: Alec Jacobson

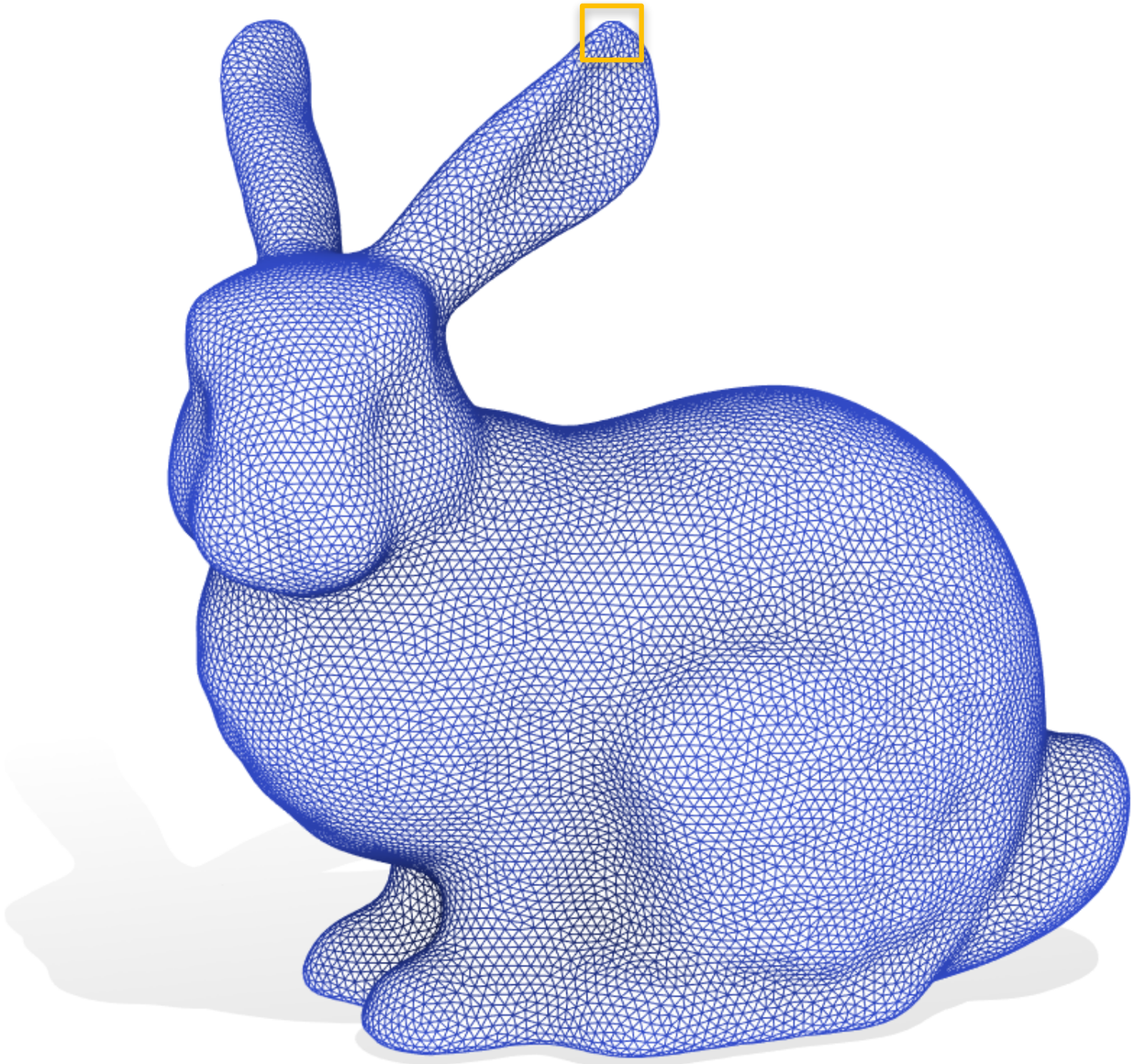
Triangle meshes discretize surfaces…

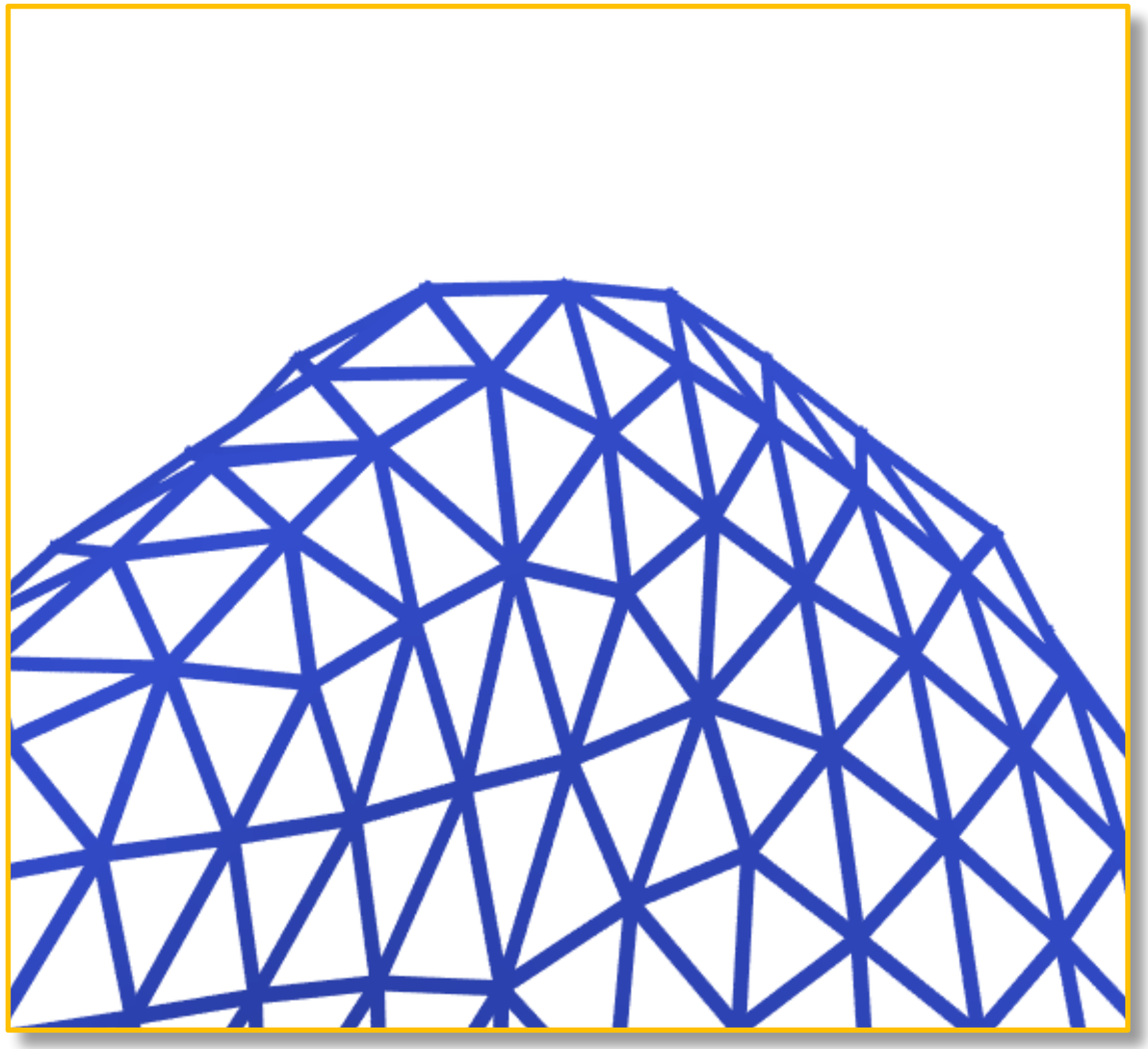


Triangle meshes discretize surfaces…

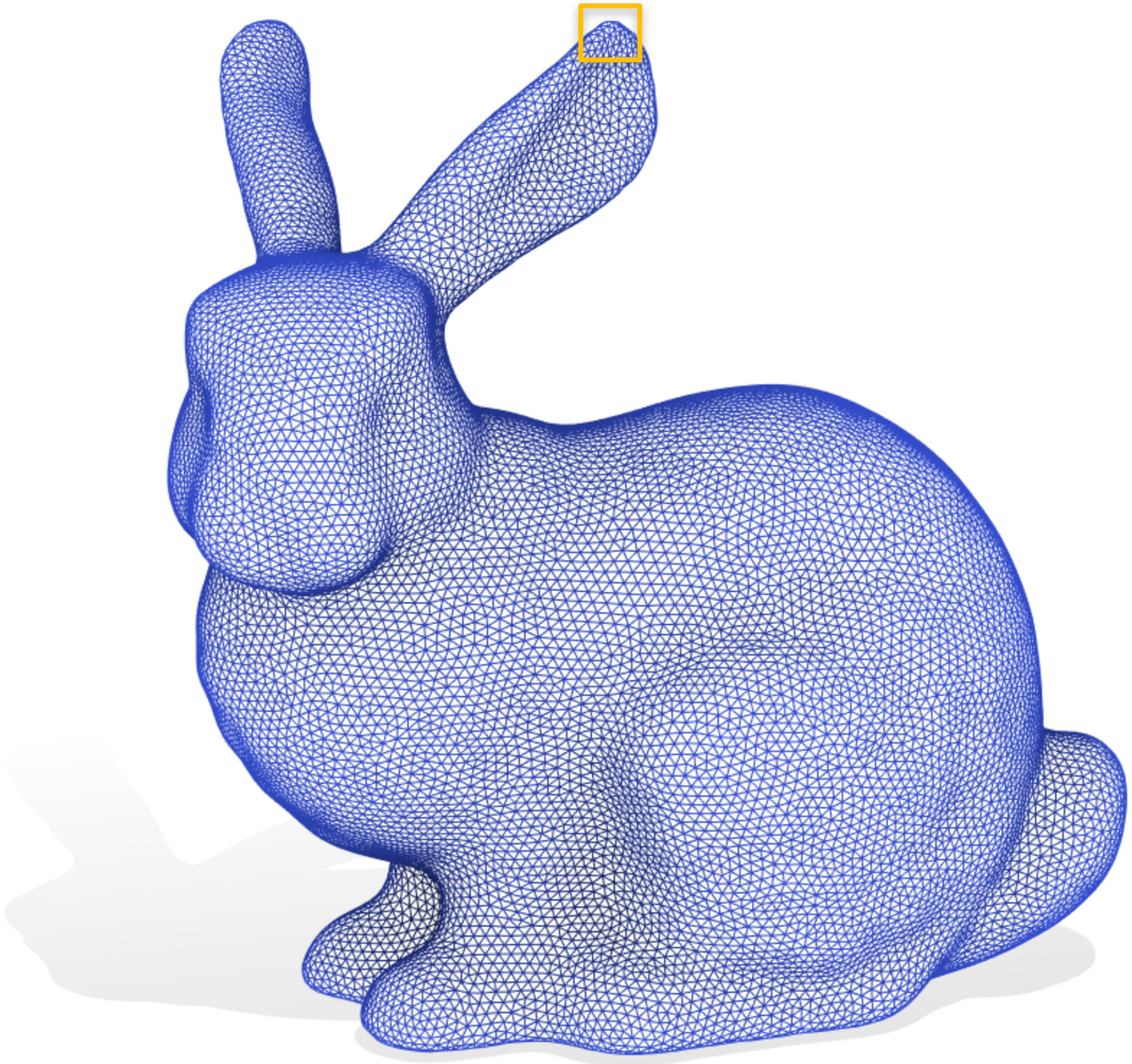


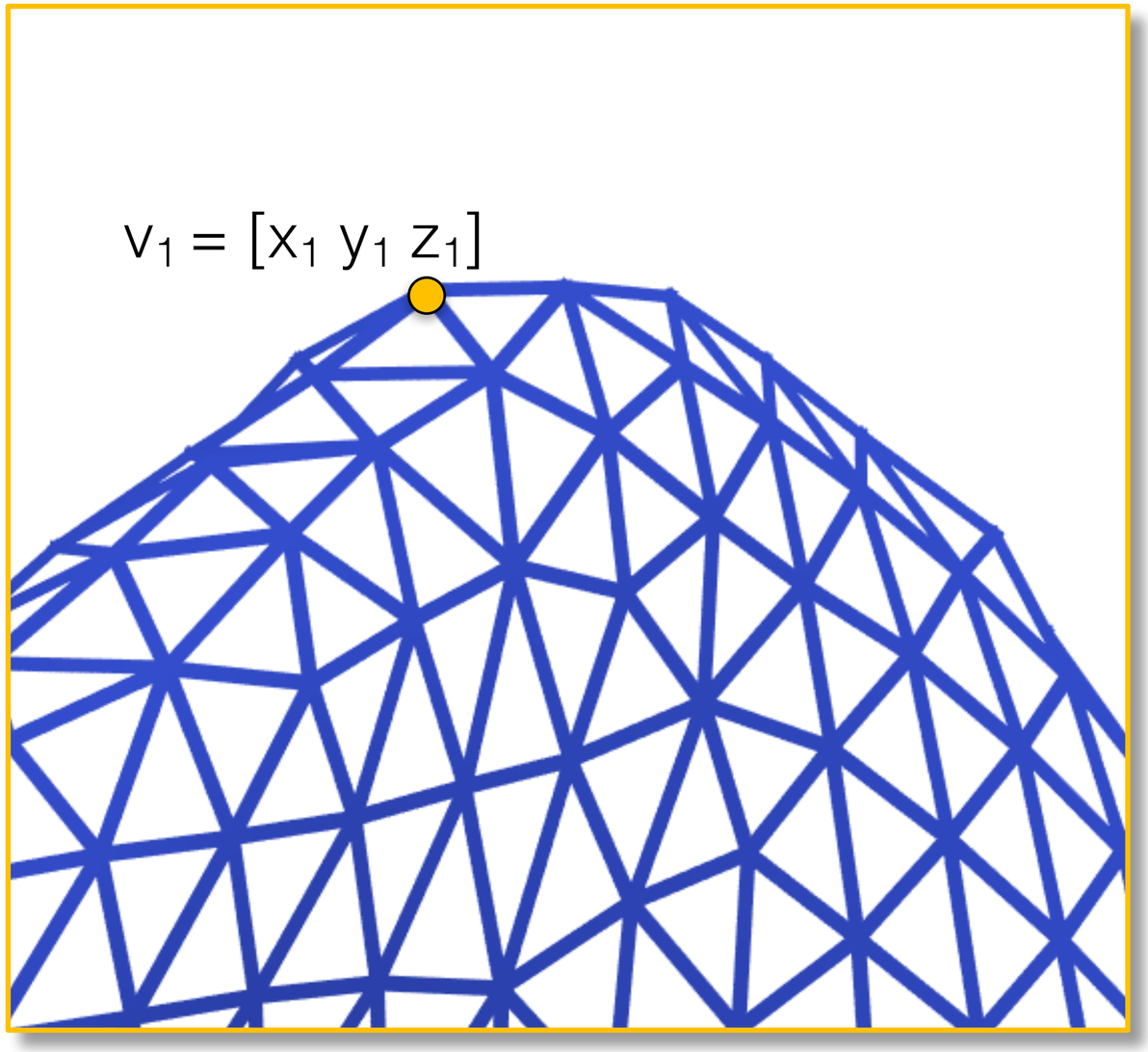
Triangle meshes discretize surfaces…



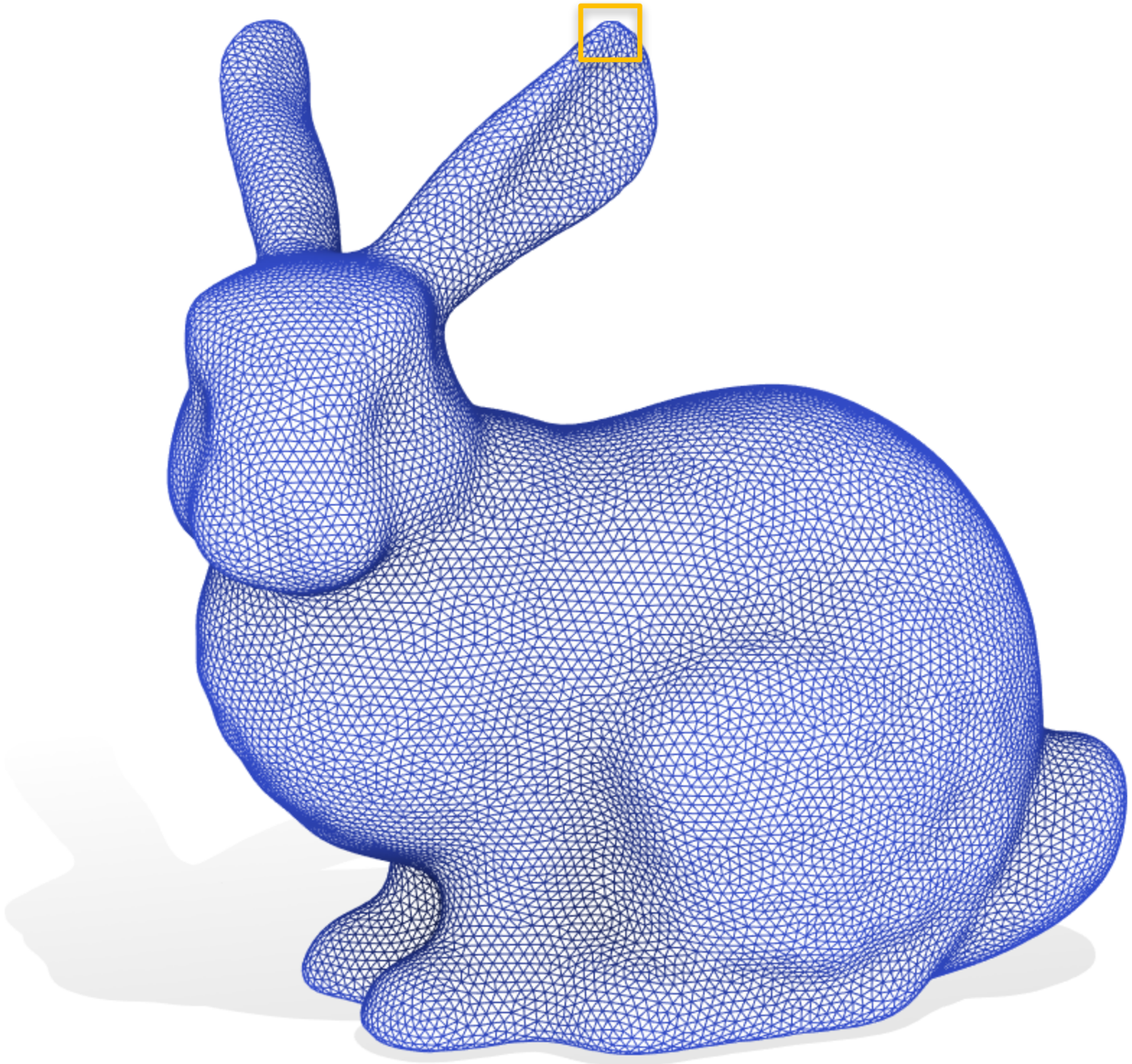


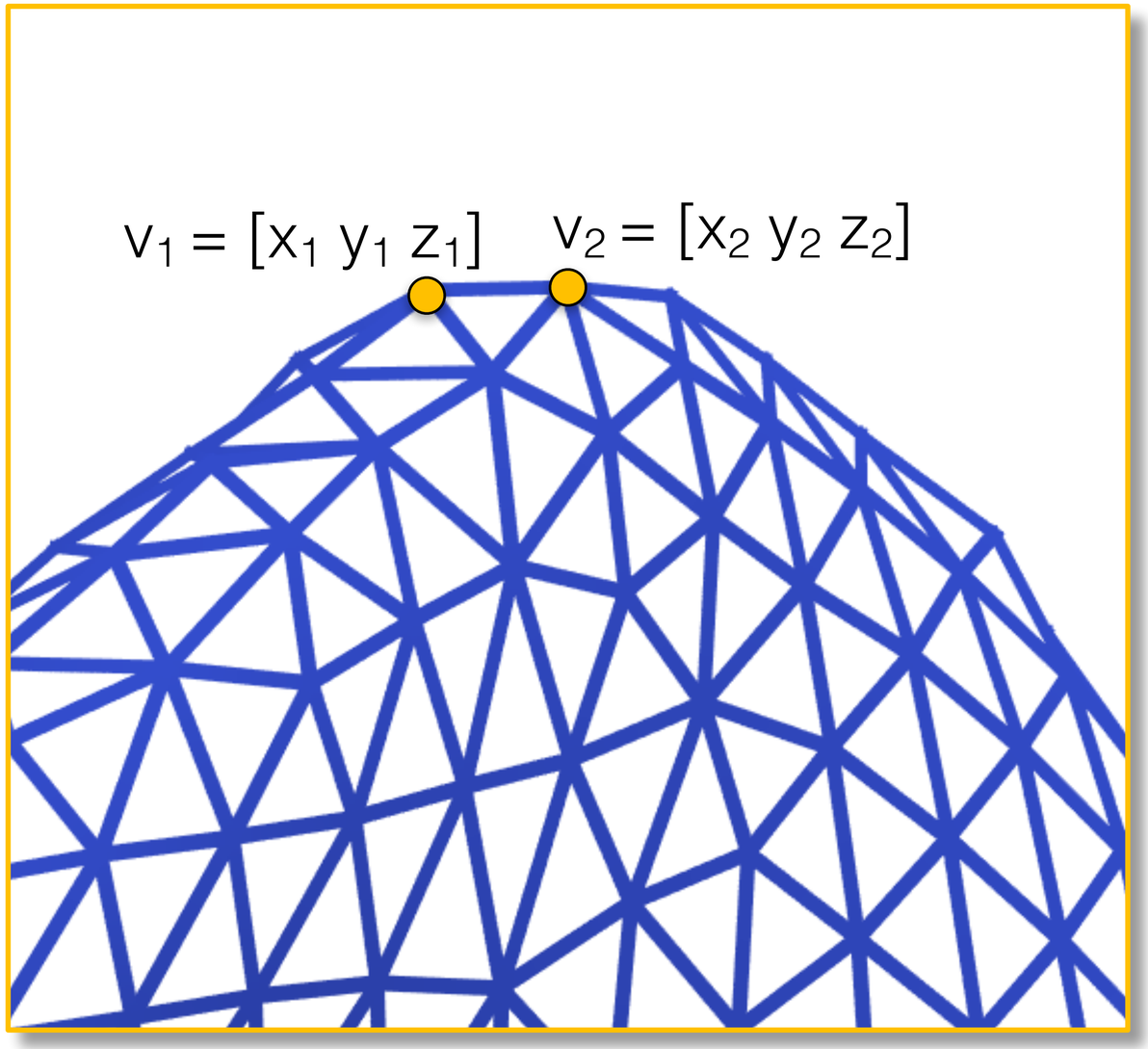
Triangle meshes discretize surfaces…



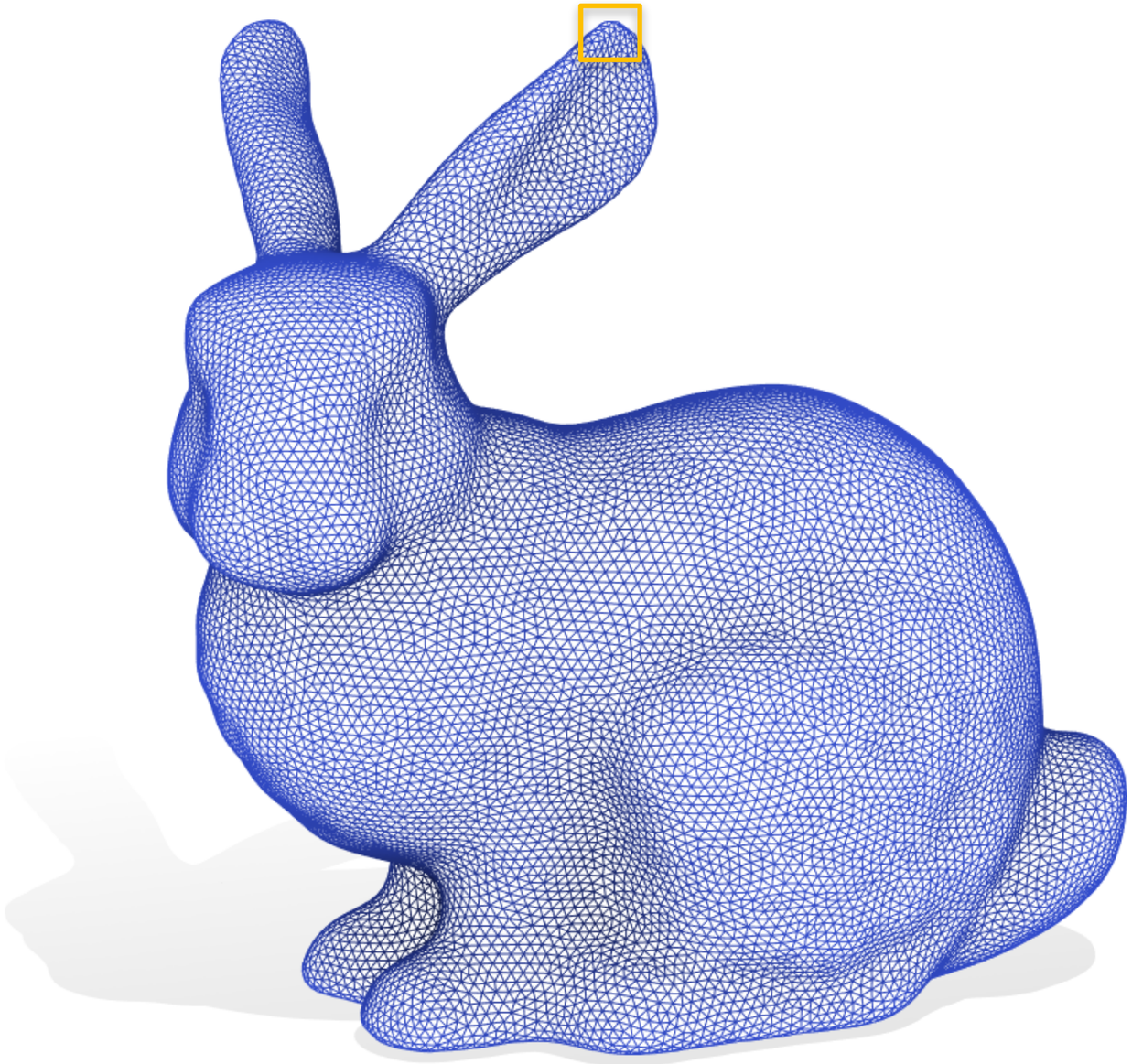


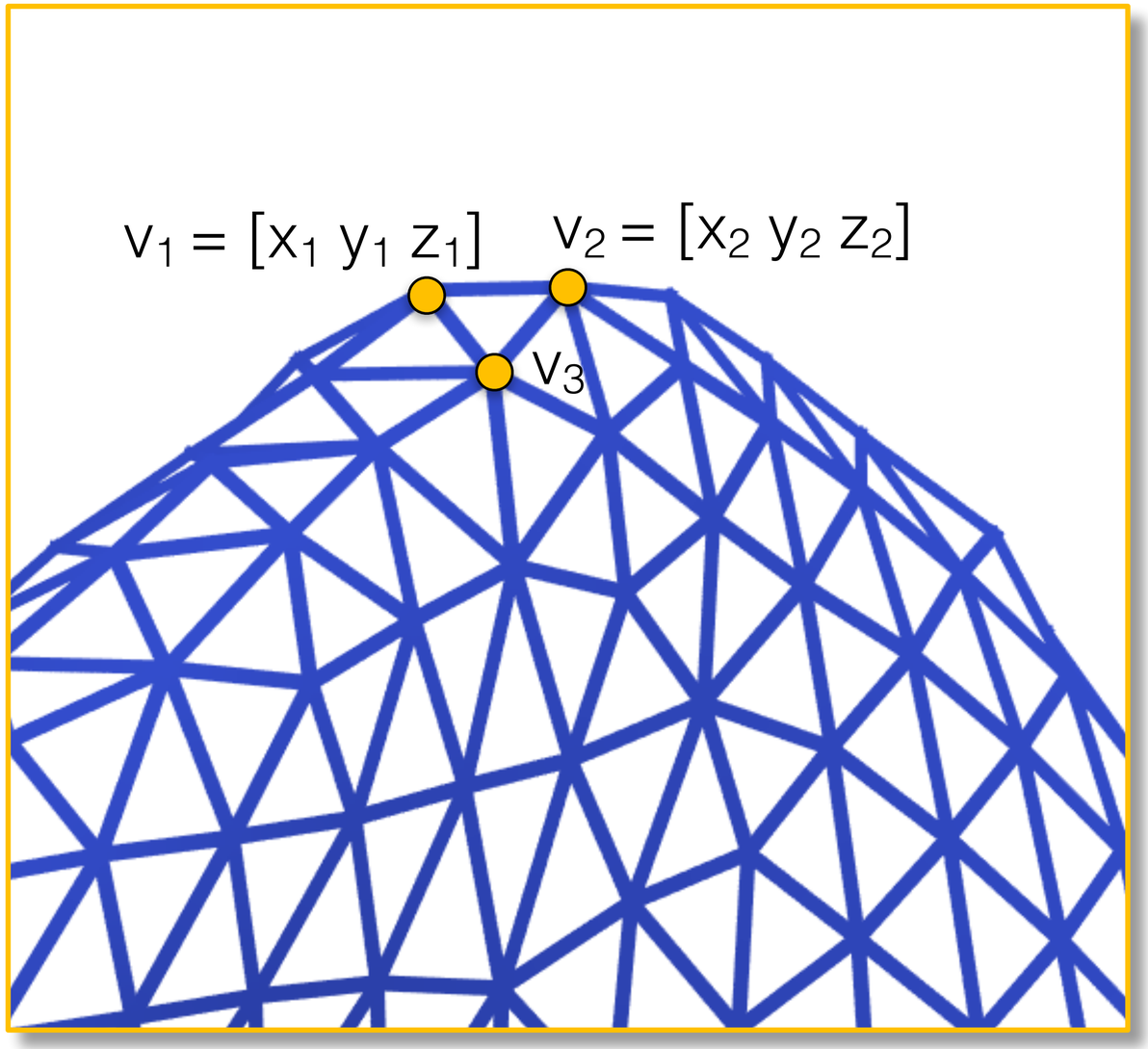
Triangle meshes discretize surfaces…



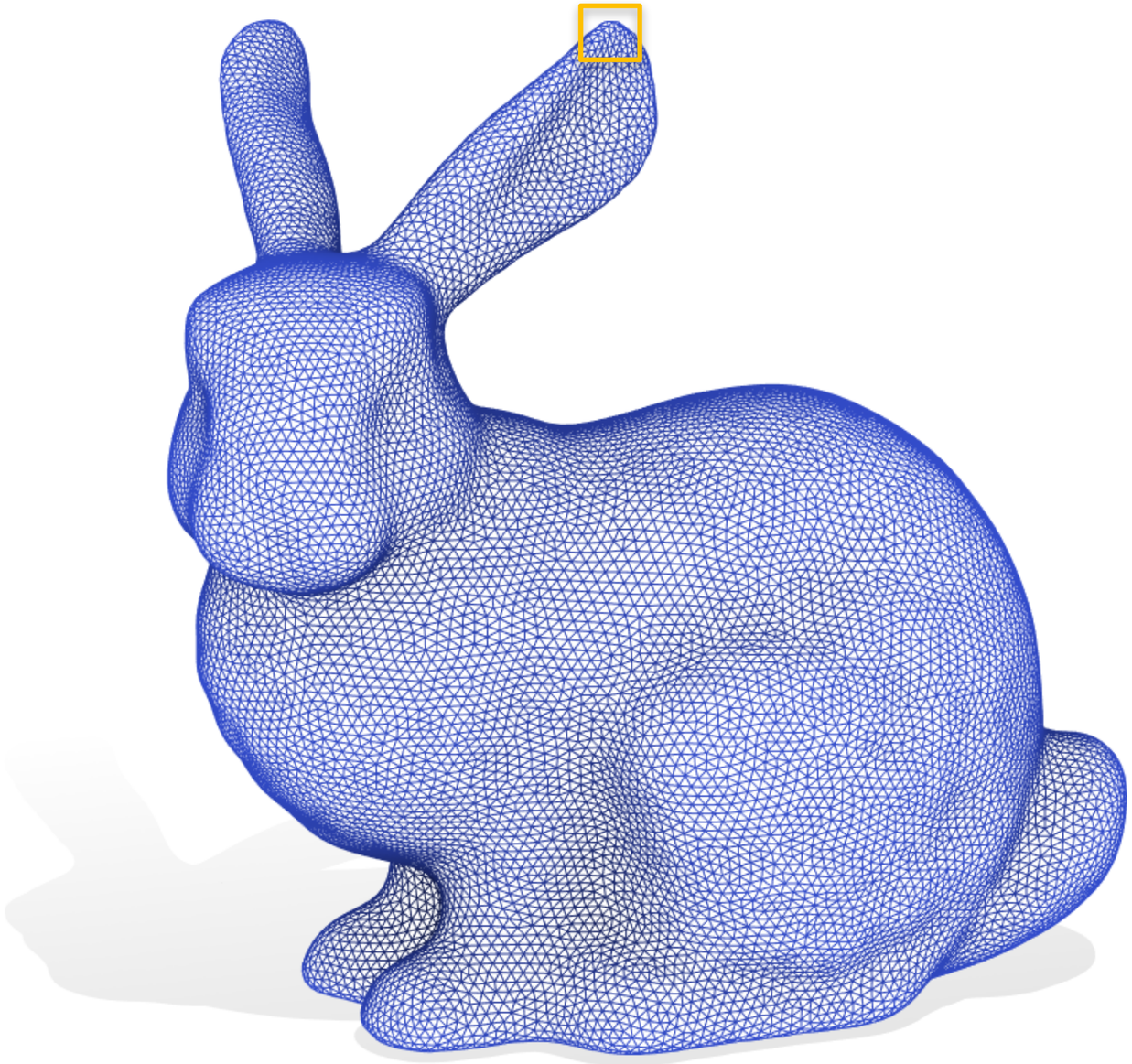


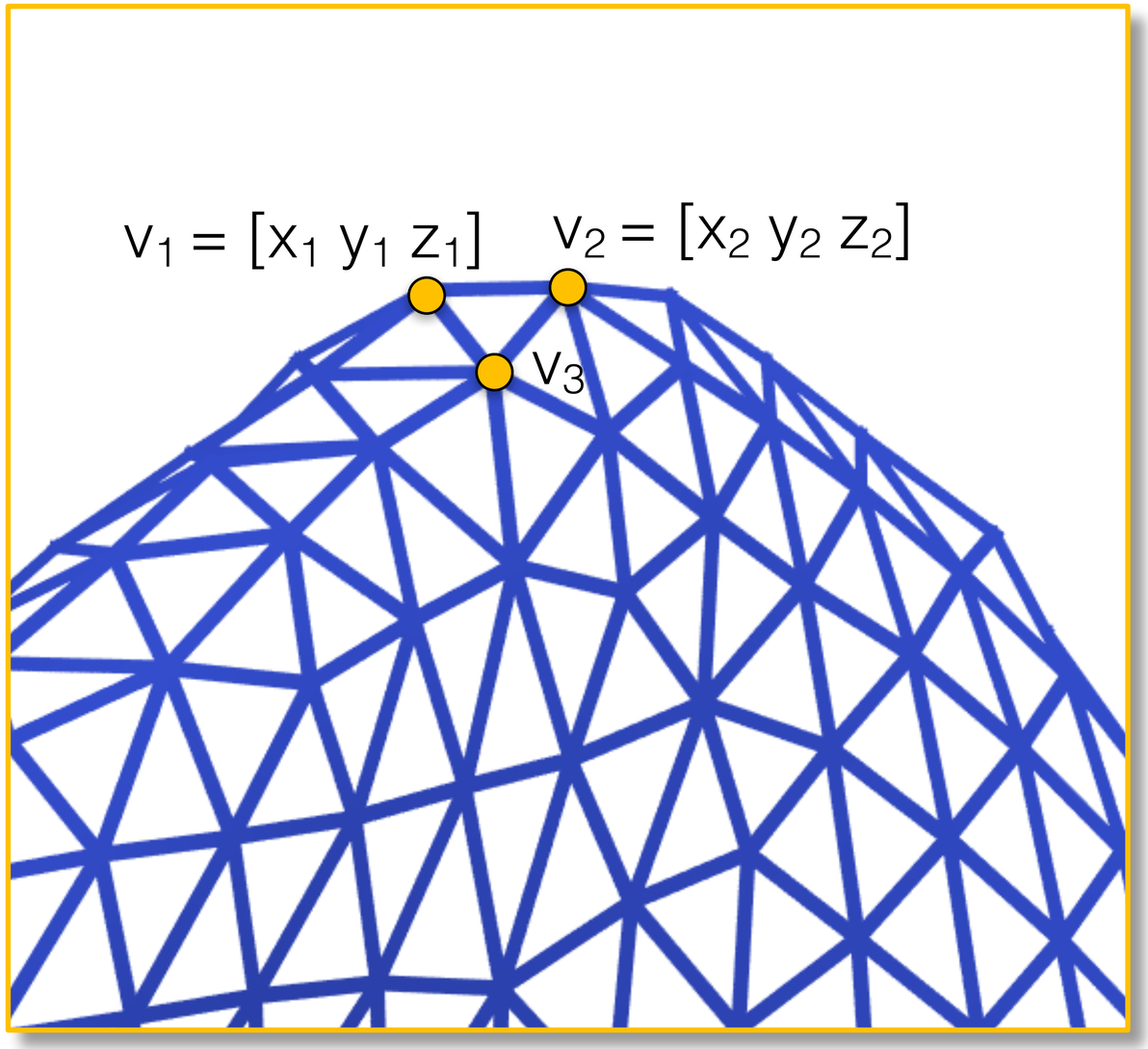
Triangle meshes discretize surfaces…



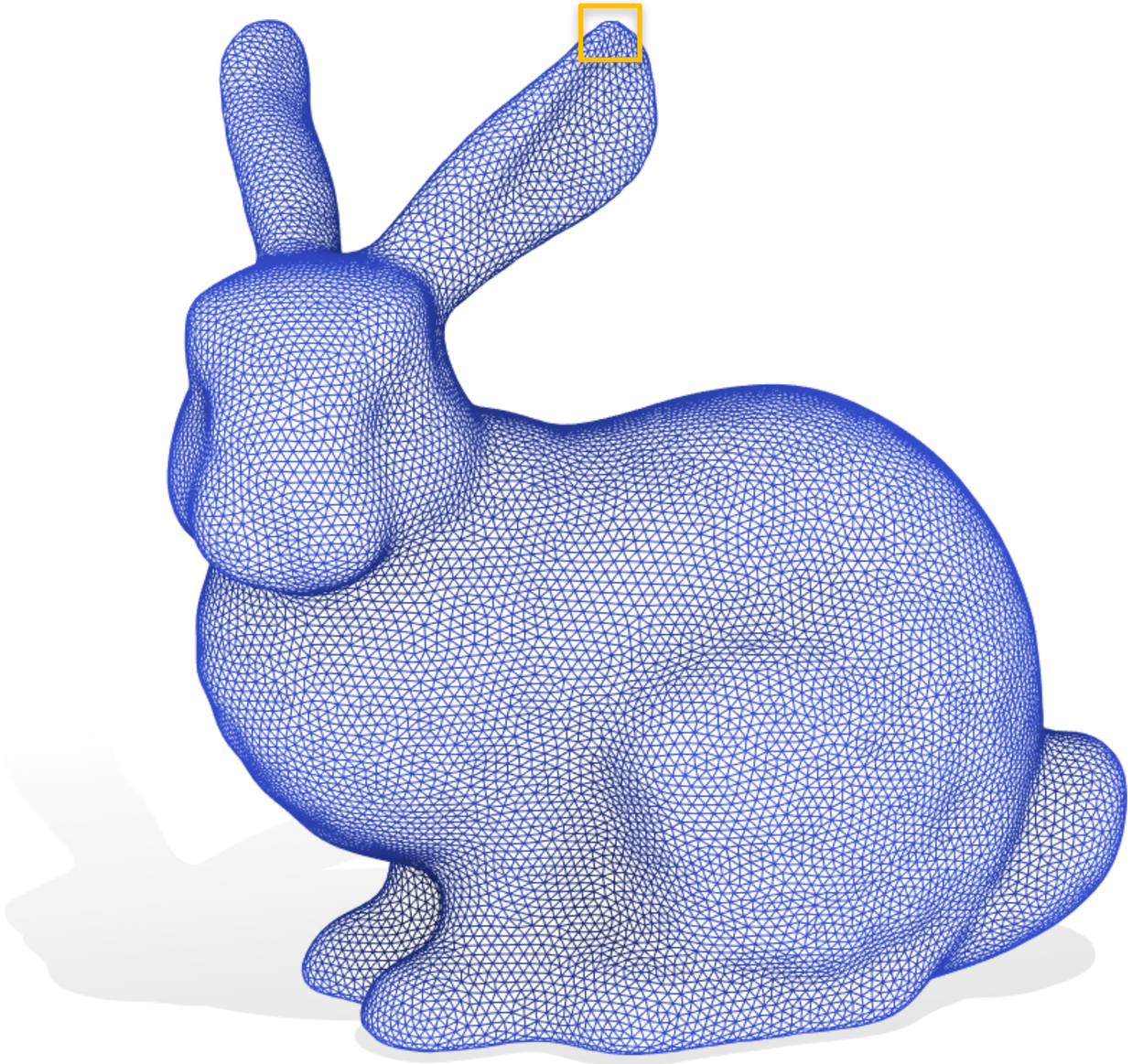


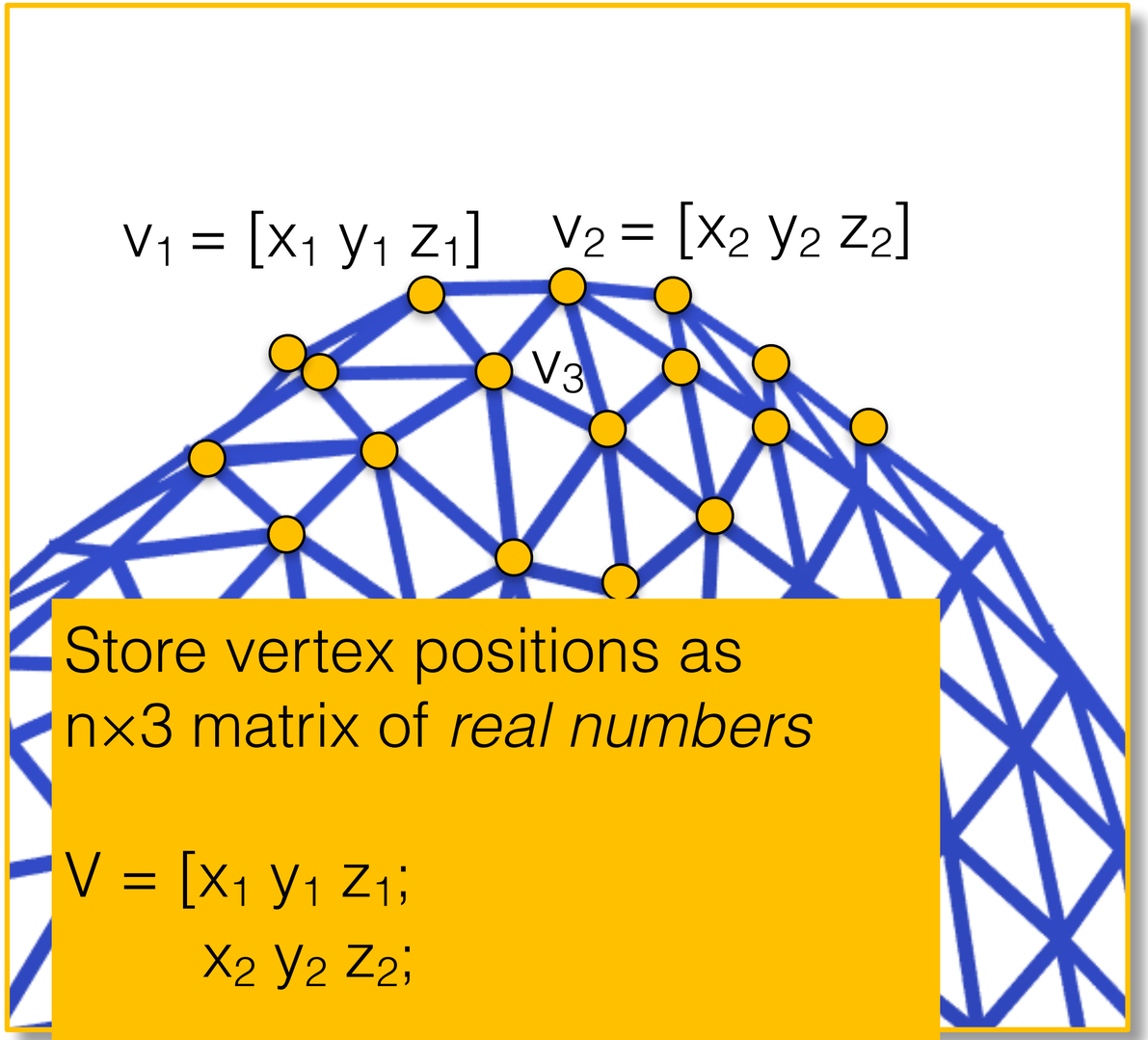
Triangle meshes discretize surfaces…





Triangle meshes discretize surfaces…

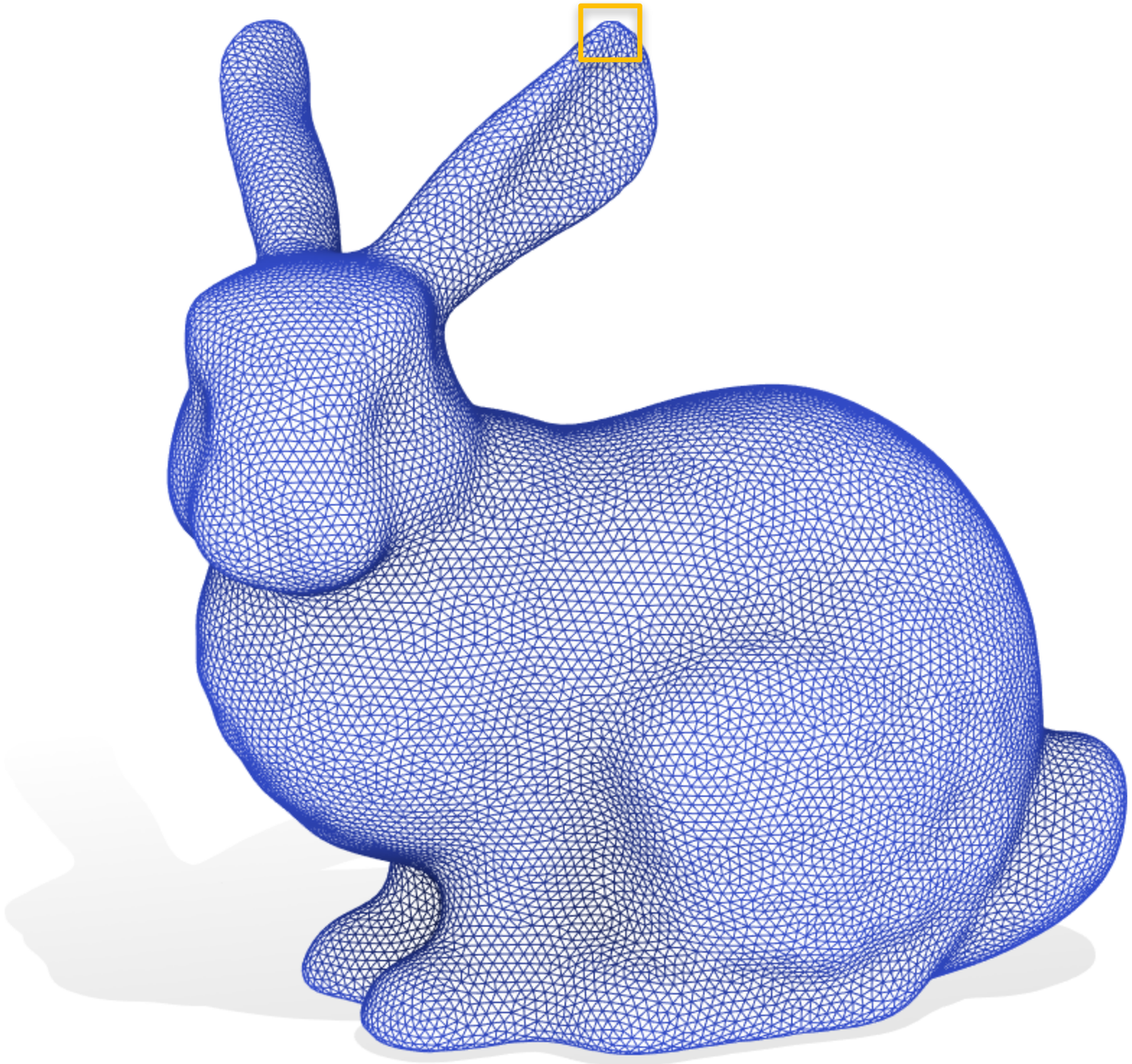


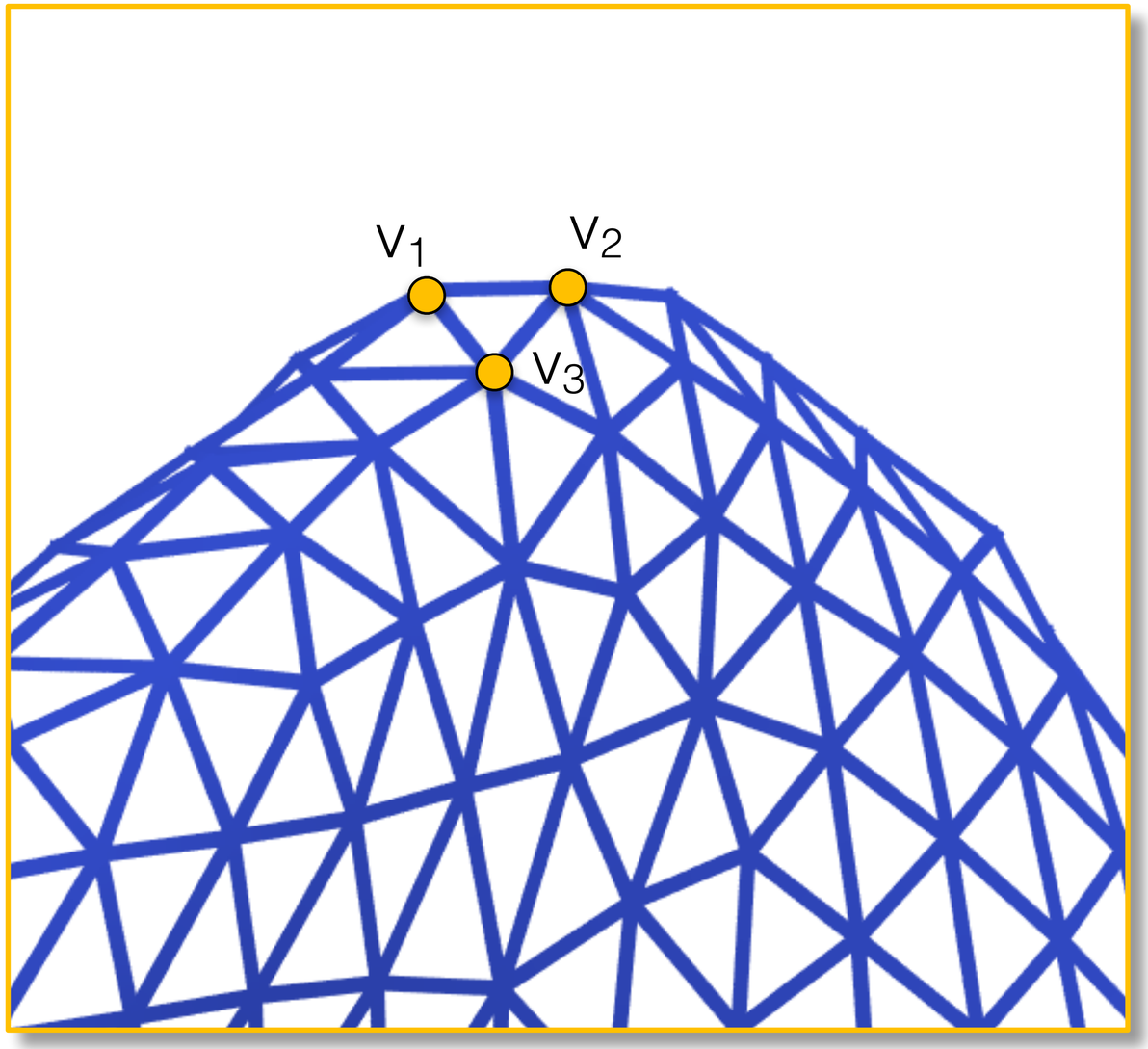


Store vertex positions as
n×3 matrix of *real numbers*

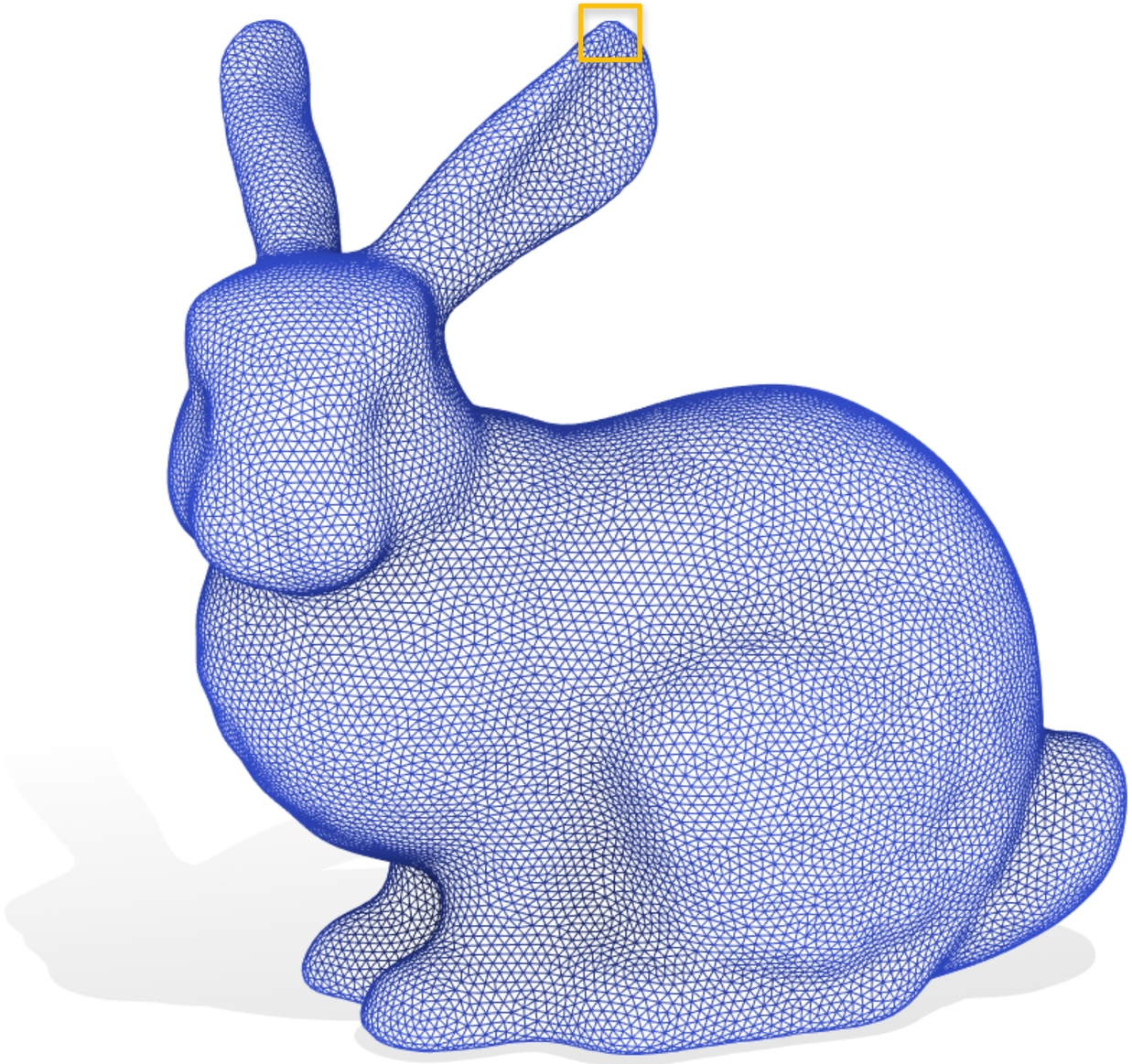
$V = [x_1\ y_1\ z_1;$
$x_2\ y_2\ z_2;$
…
$x_n\ y_n\ z_n]$

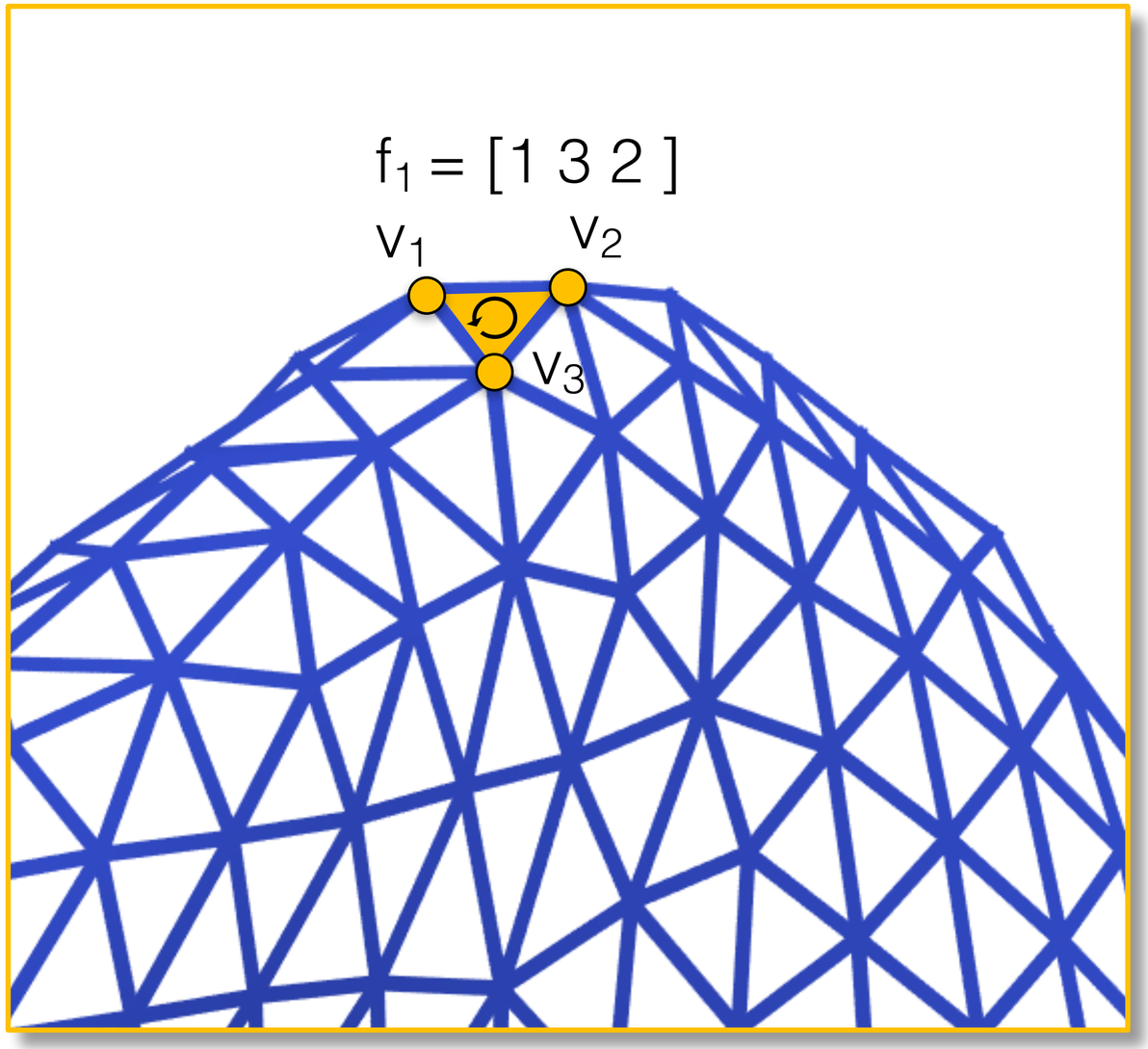
Triangle meshes discretize surfaces…



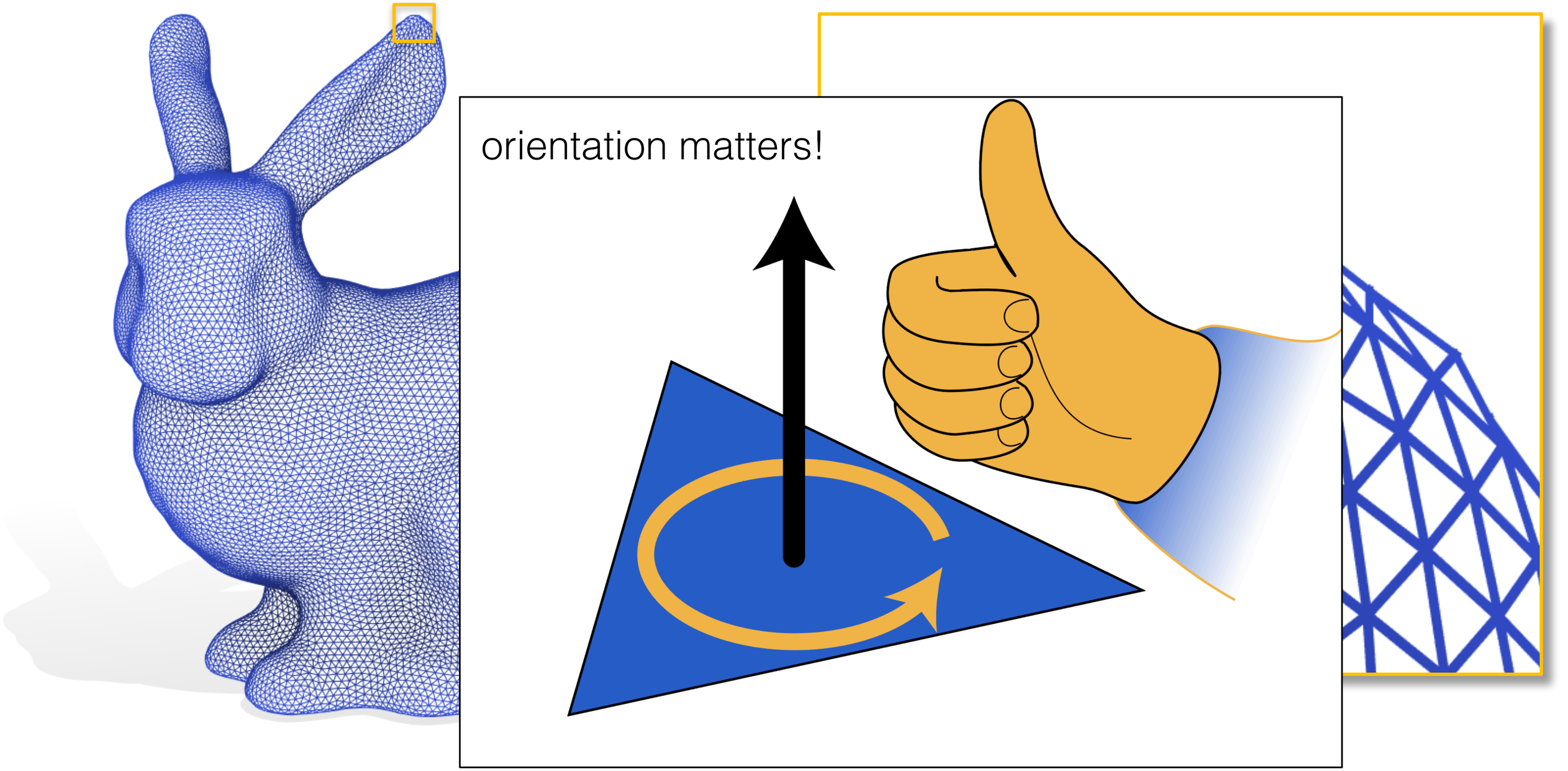


Triangle meshes discretize surfaces…

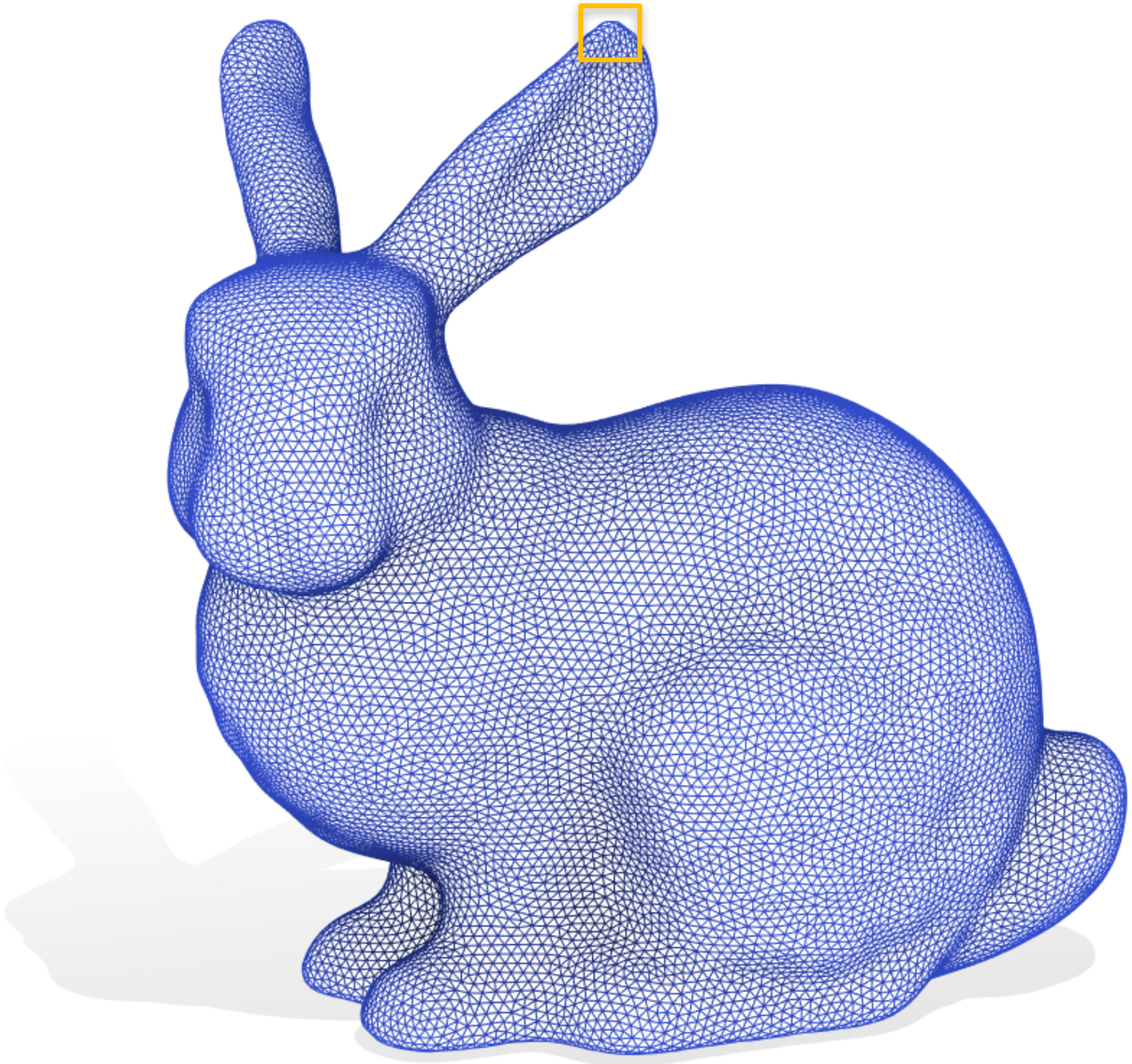


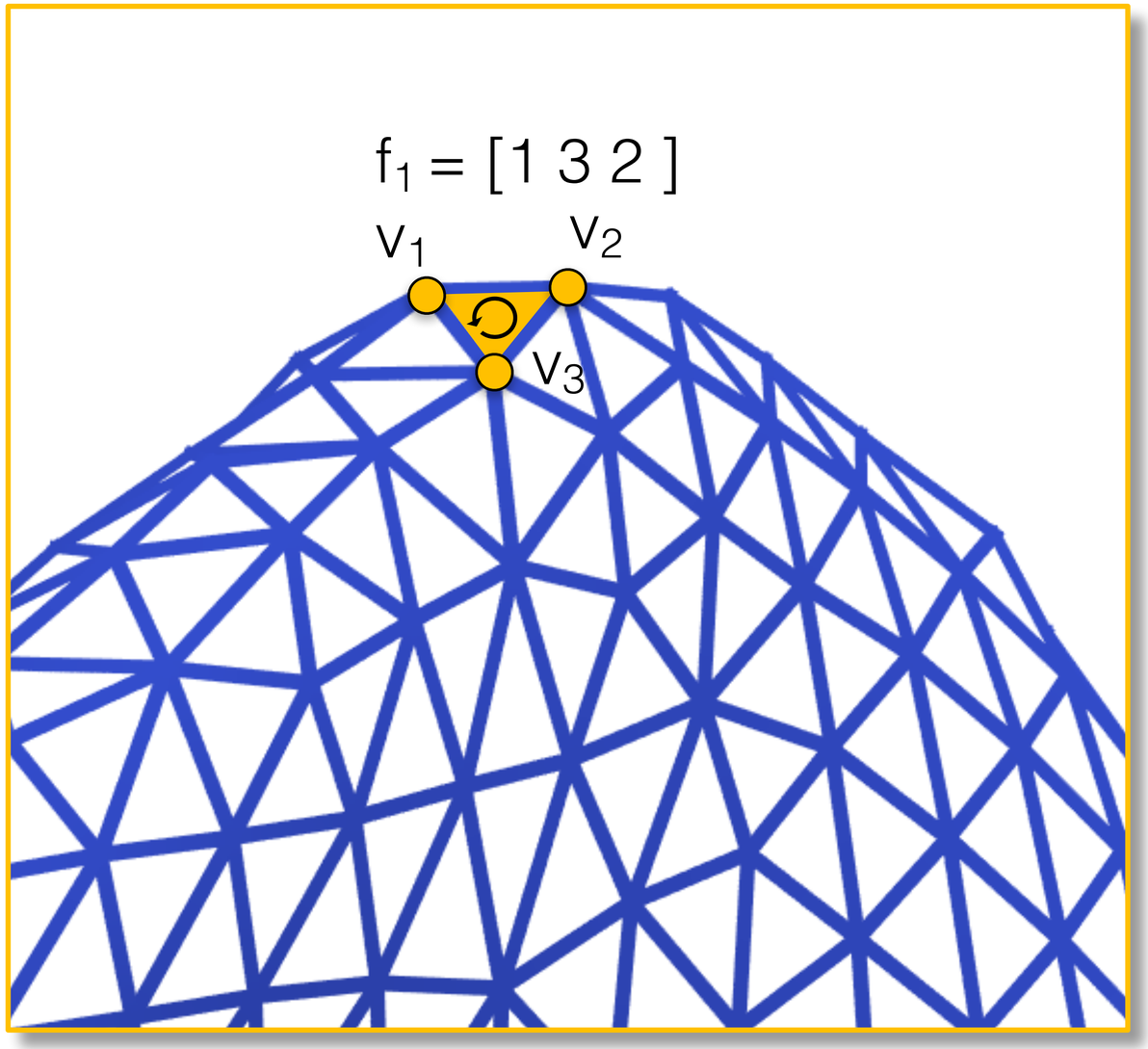


Triangle meshes discretize surfaces…

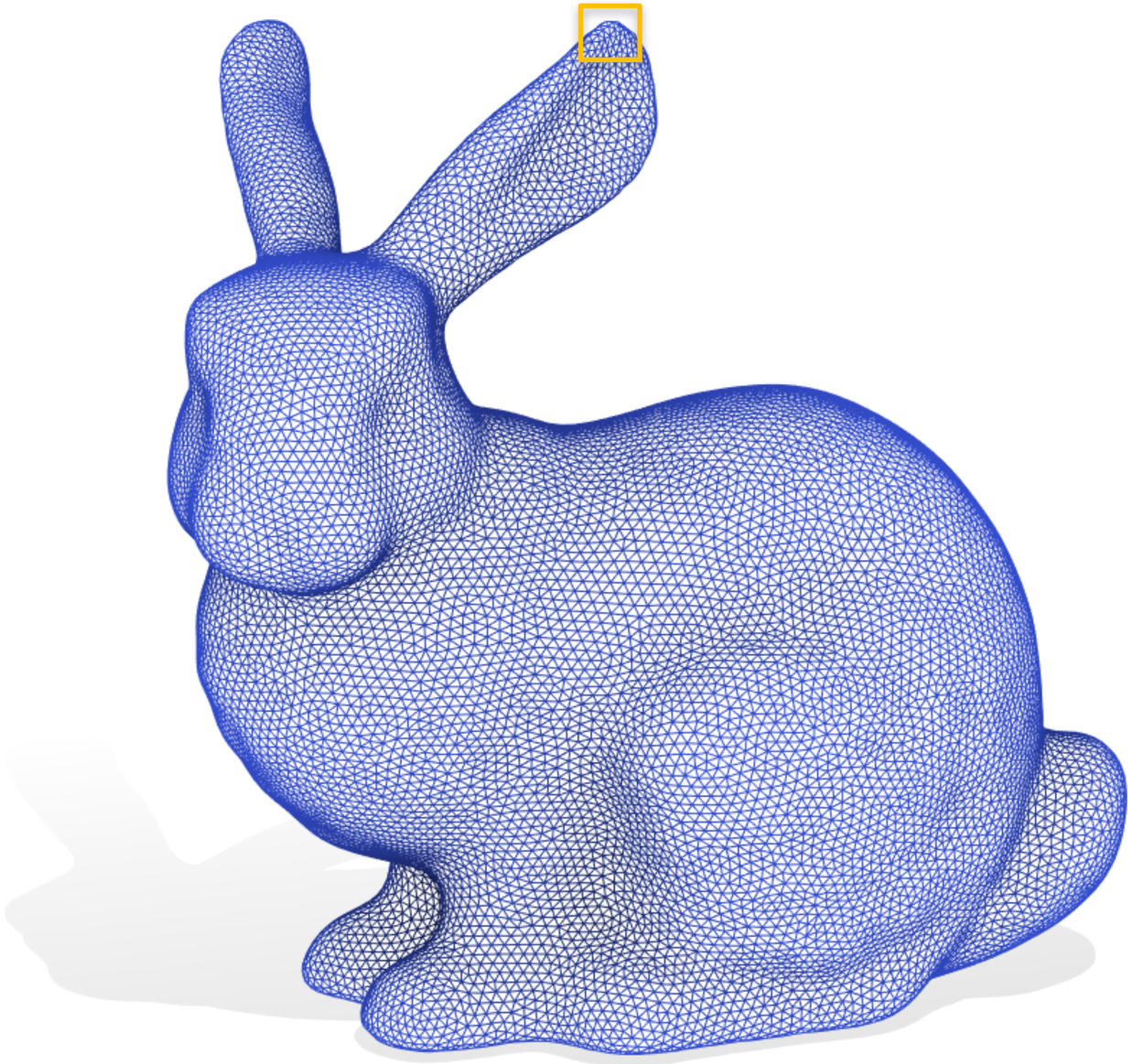


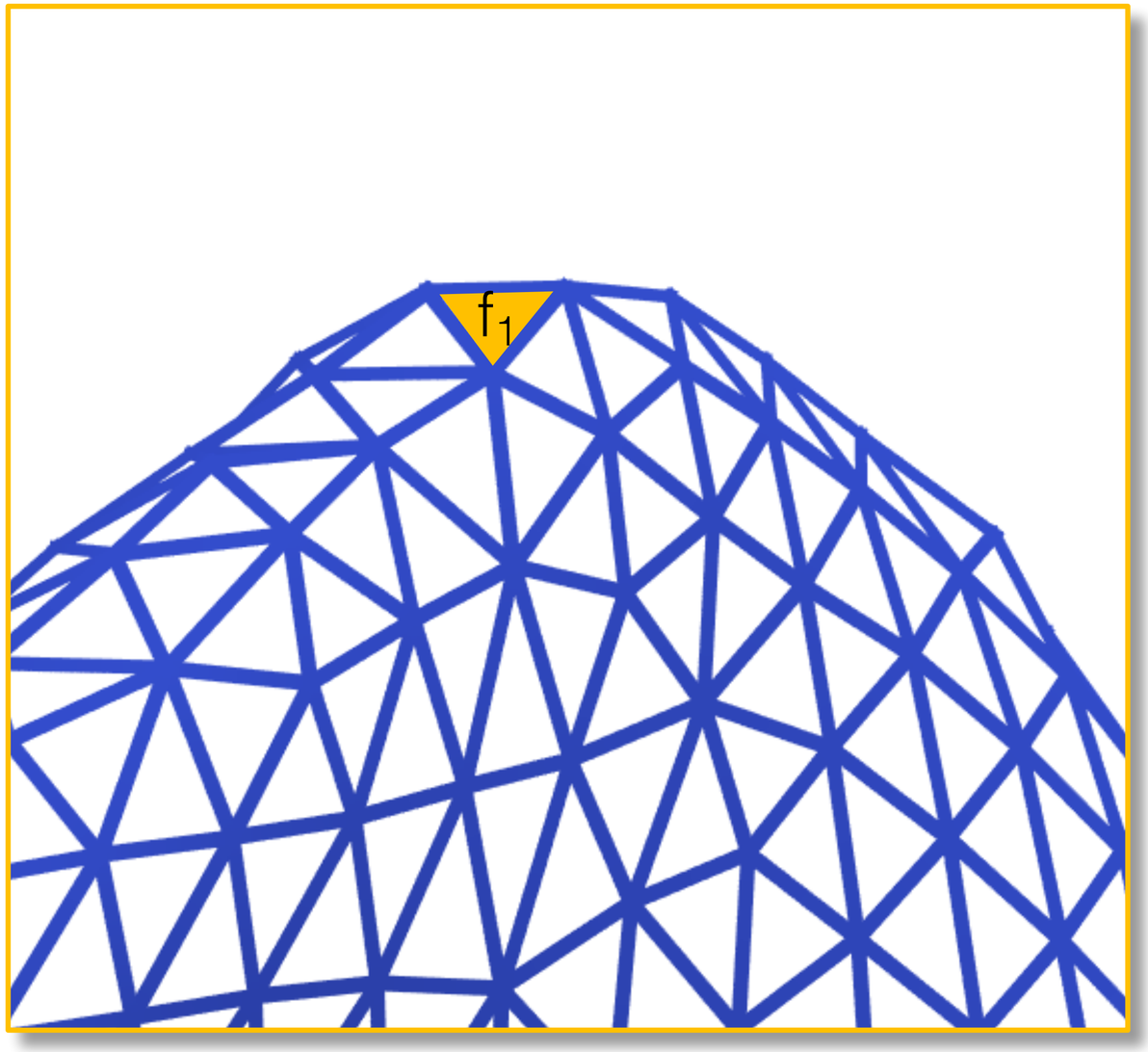
Triangle meshes discretize surfaces…



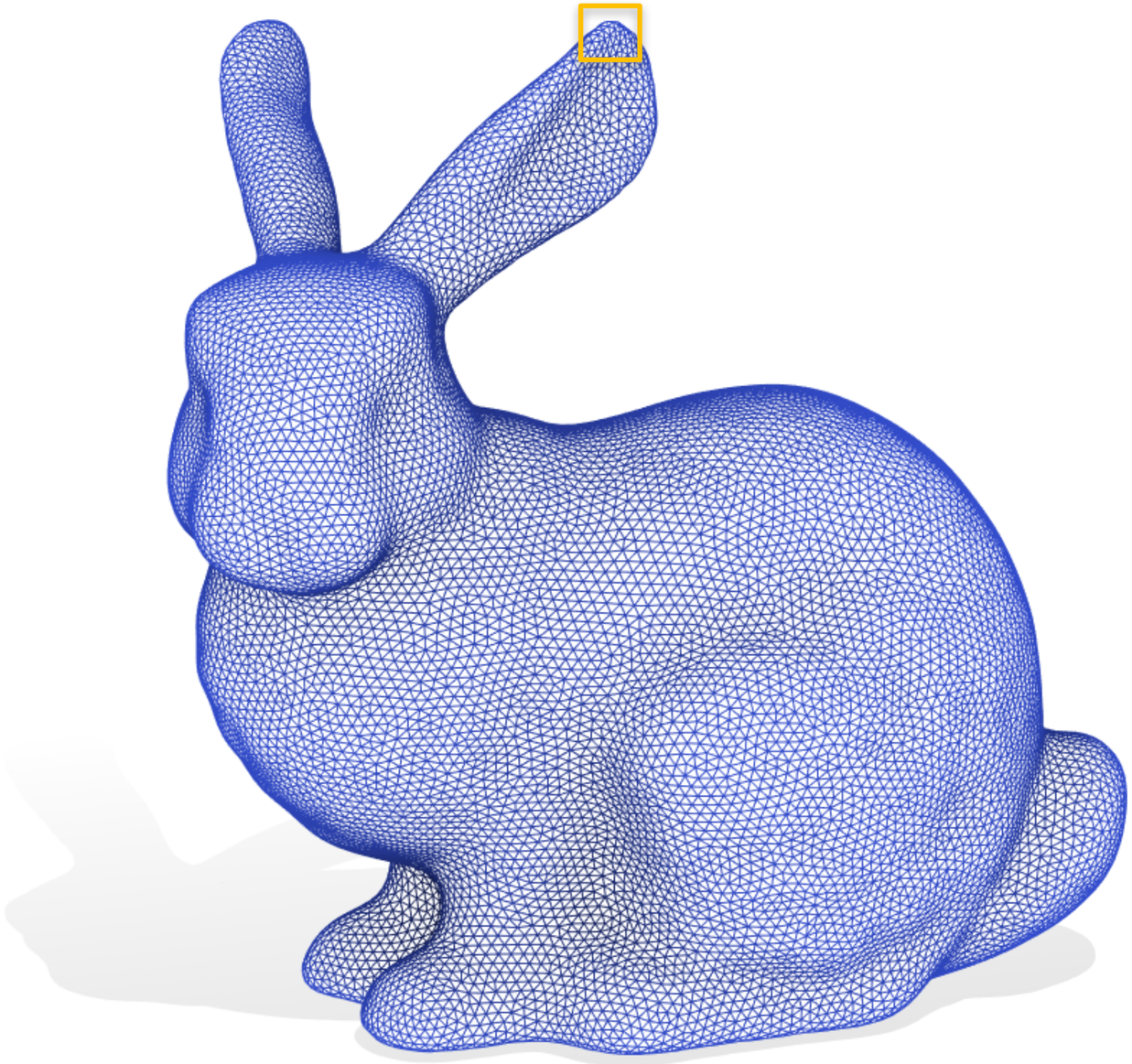


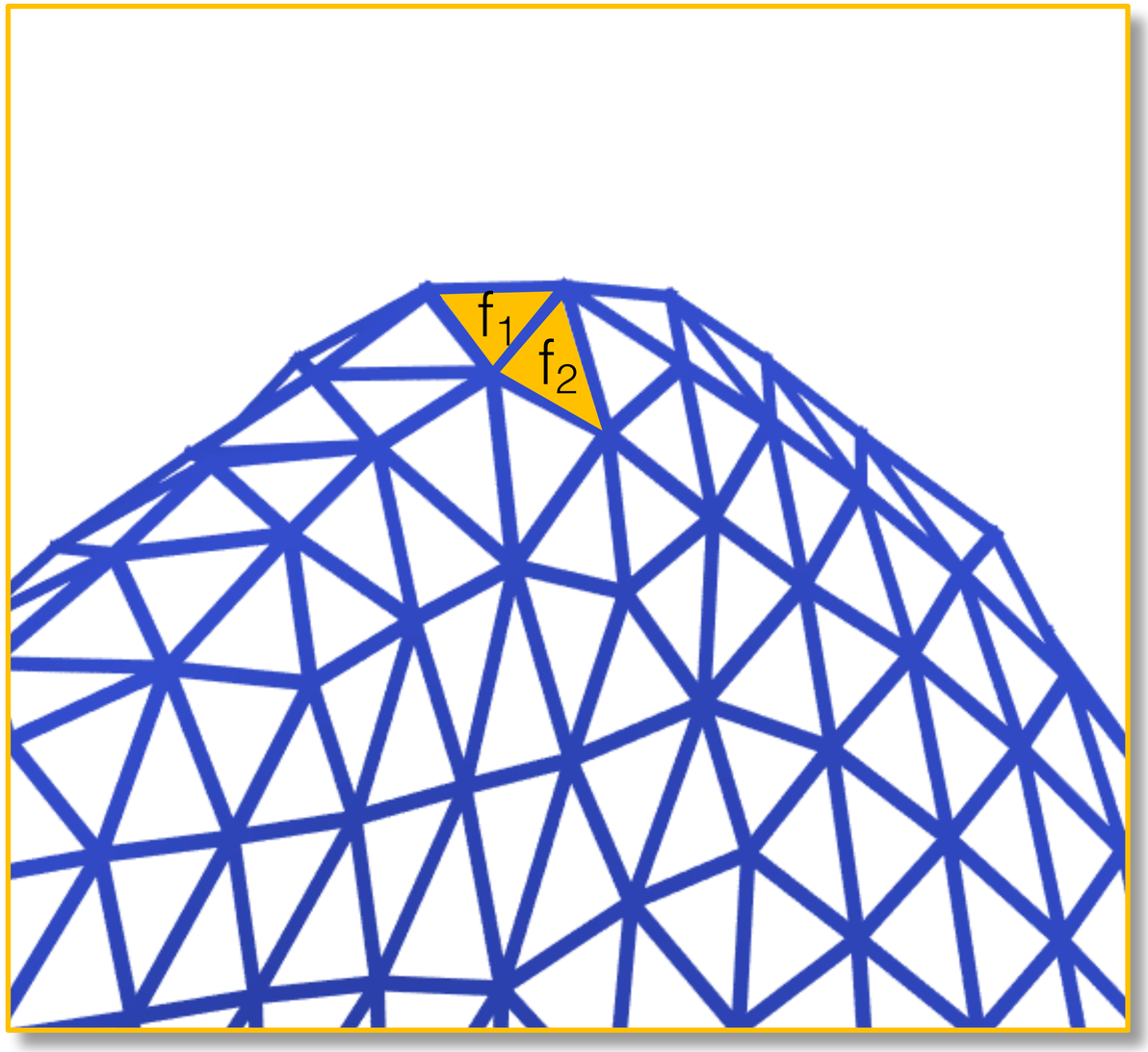
Triangle meshes discretize surfaces…



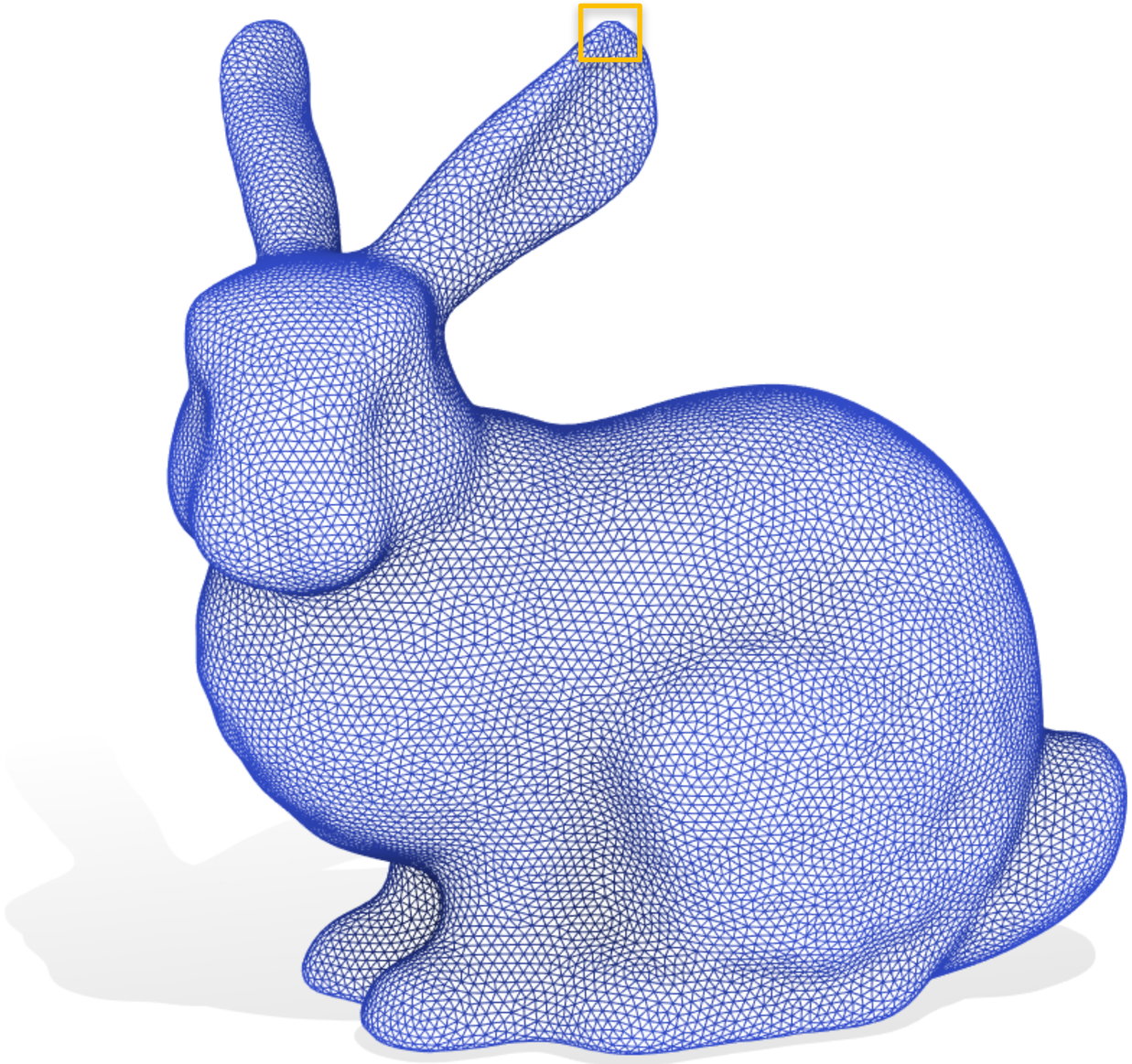


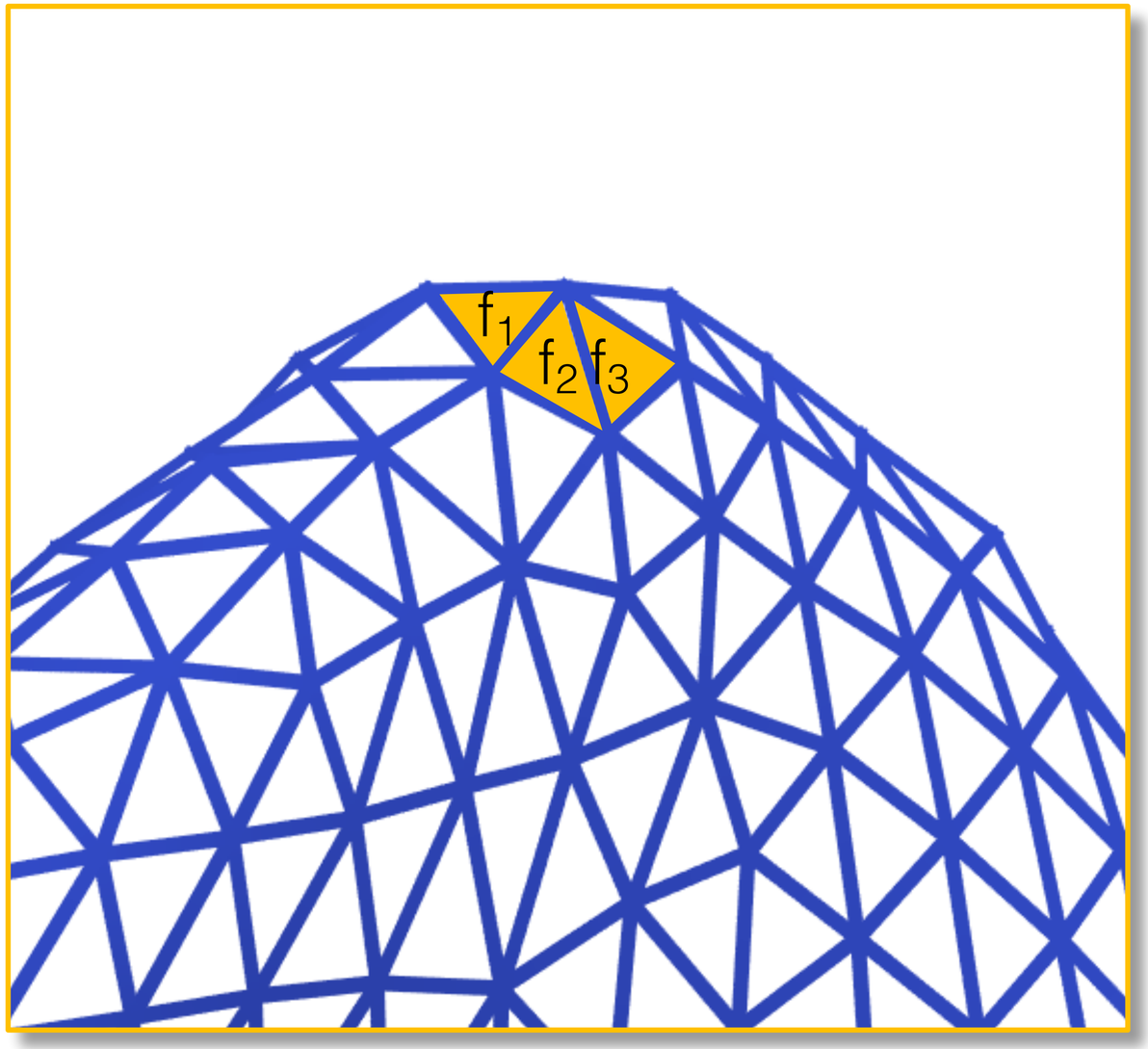
Triangle meshes discretize surfaces…



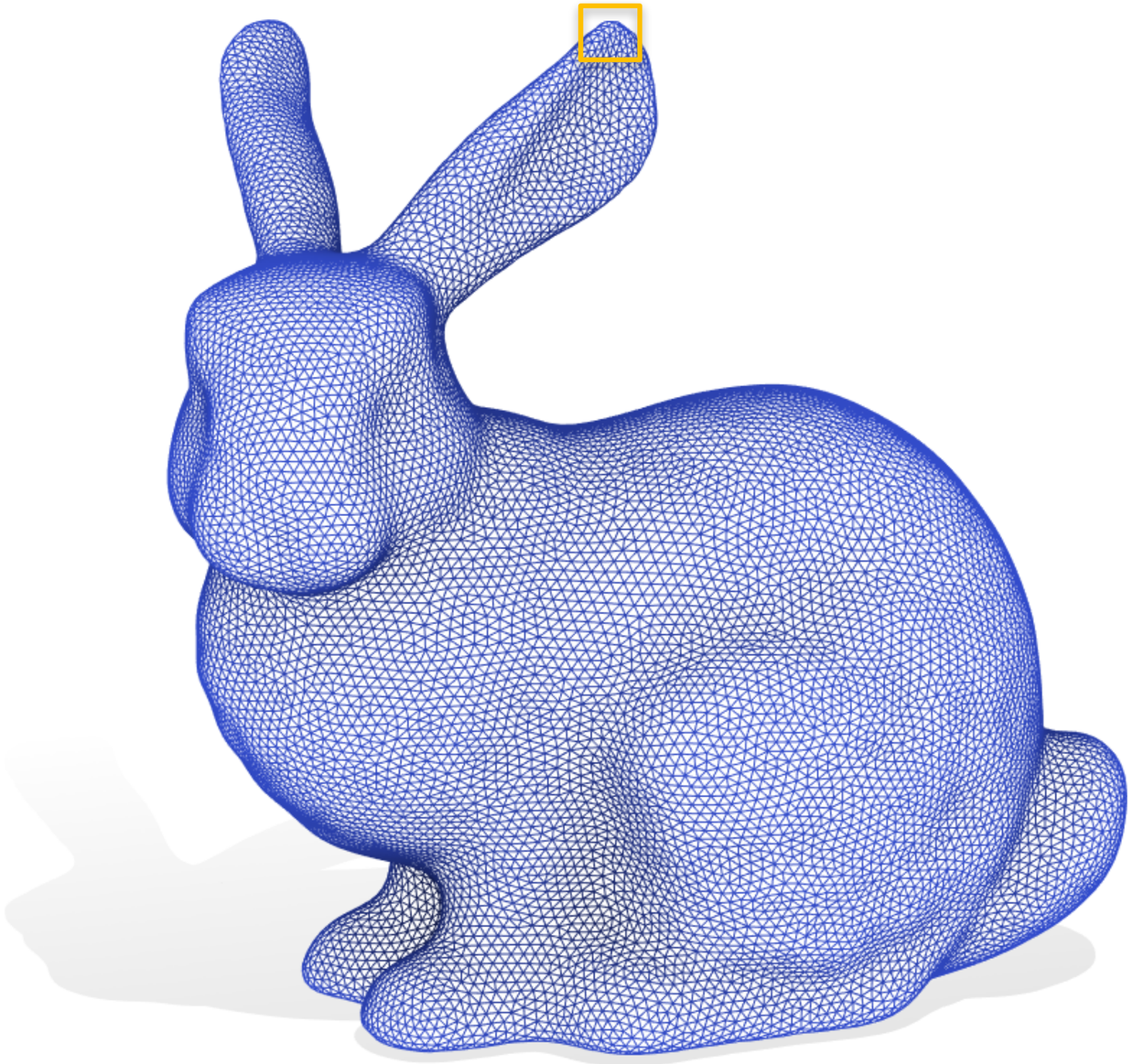


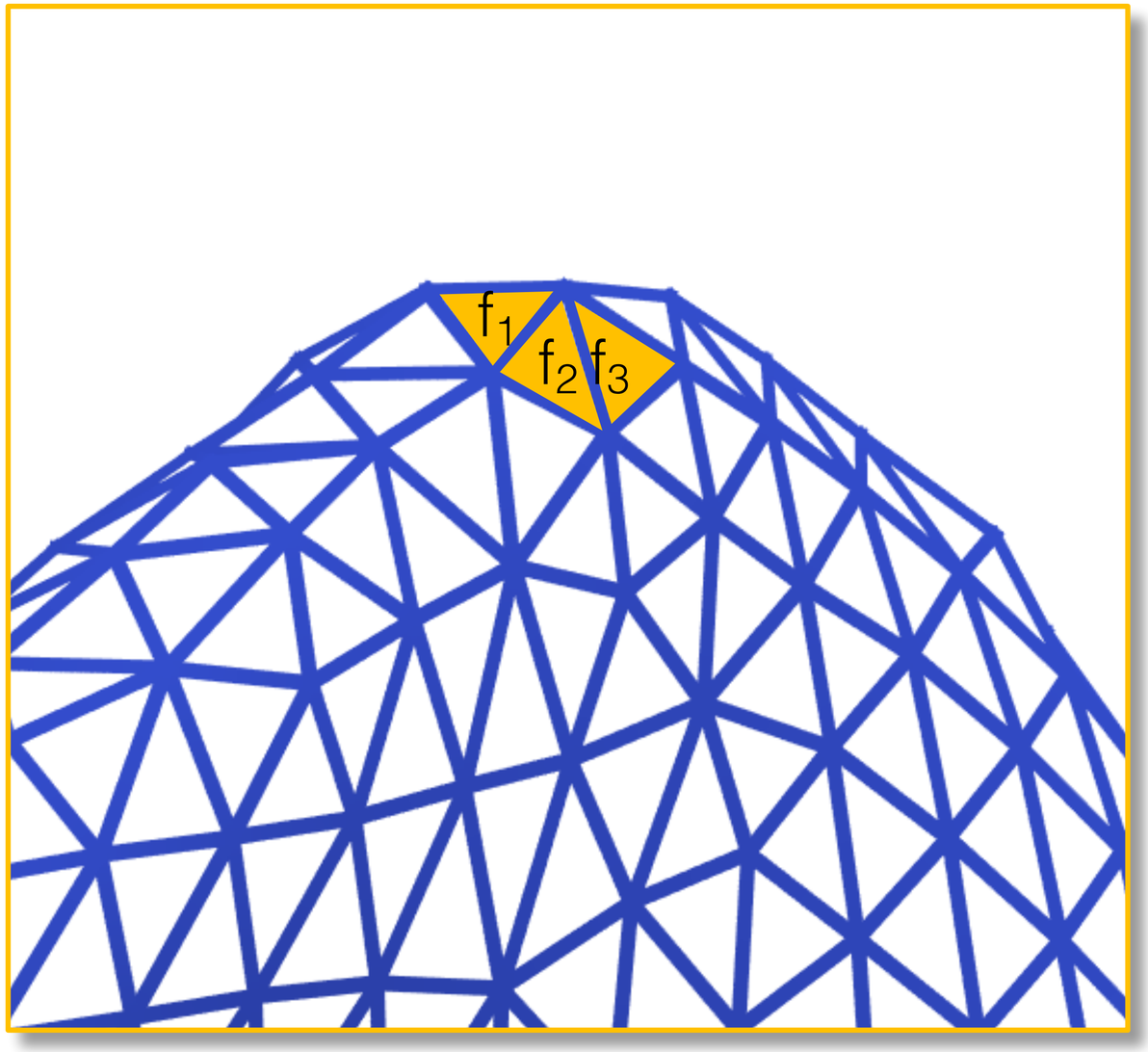
Triangle meshes discretize surfaces…



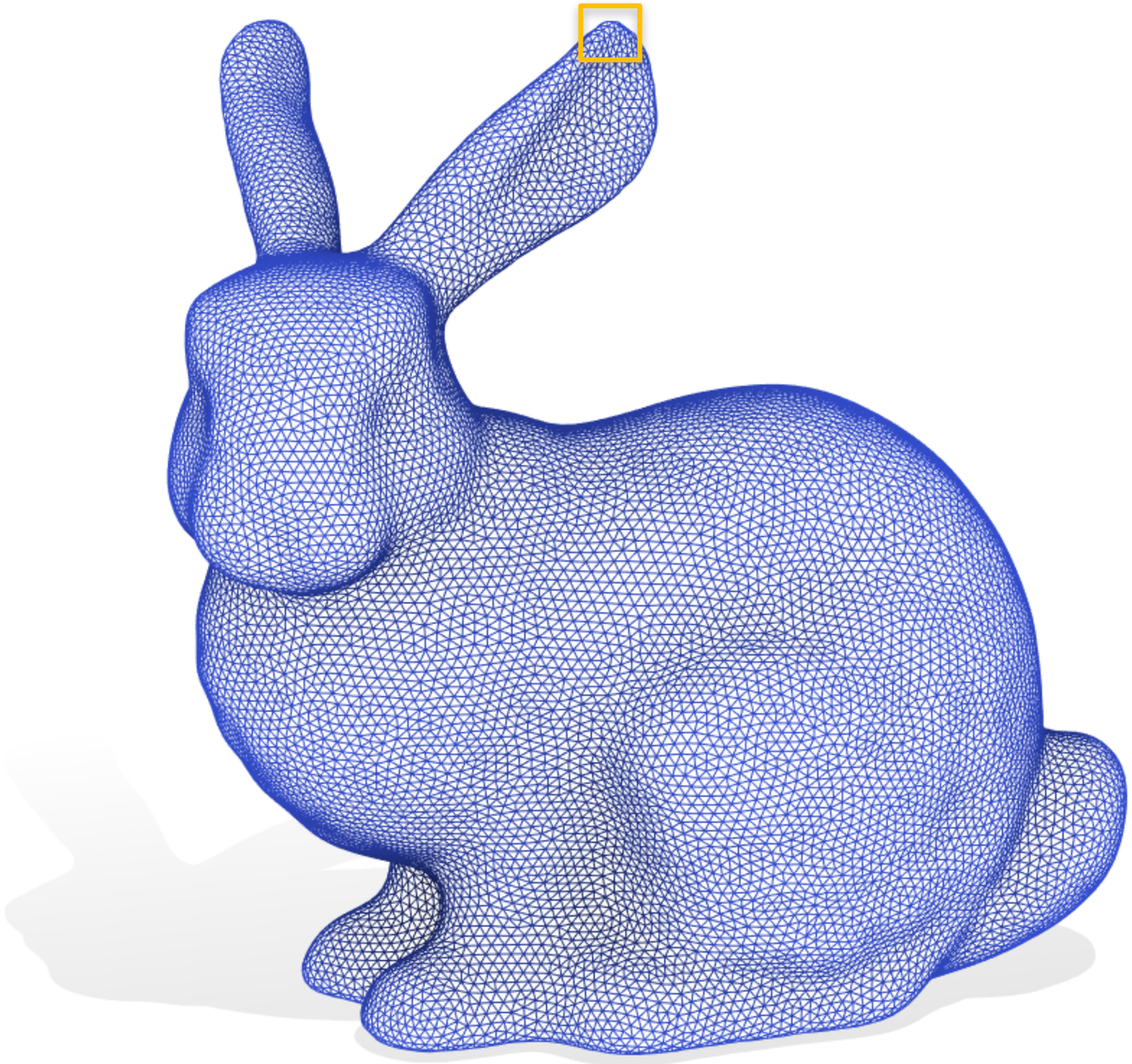


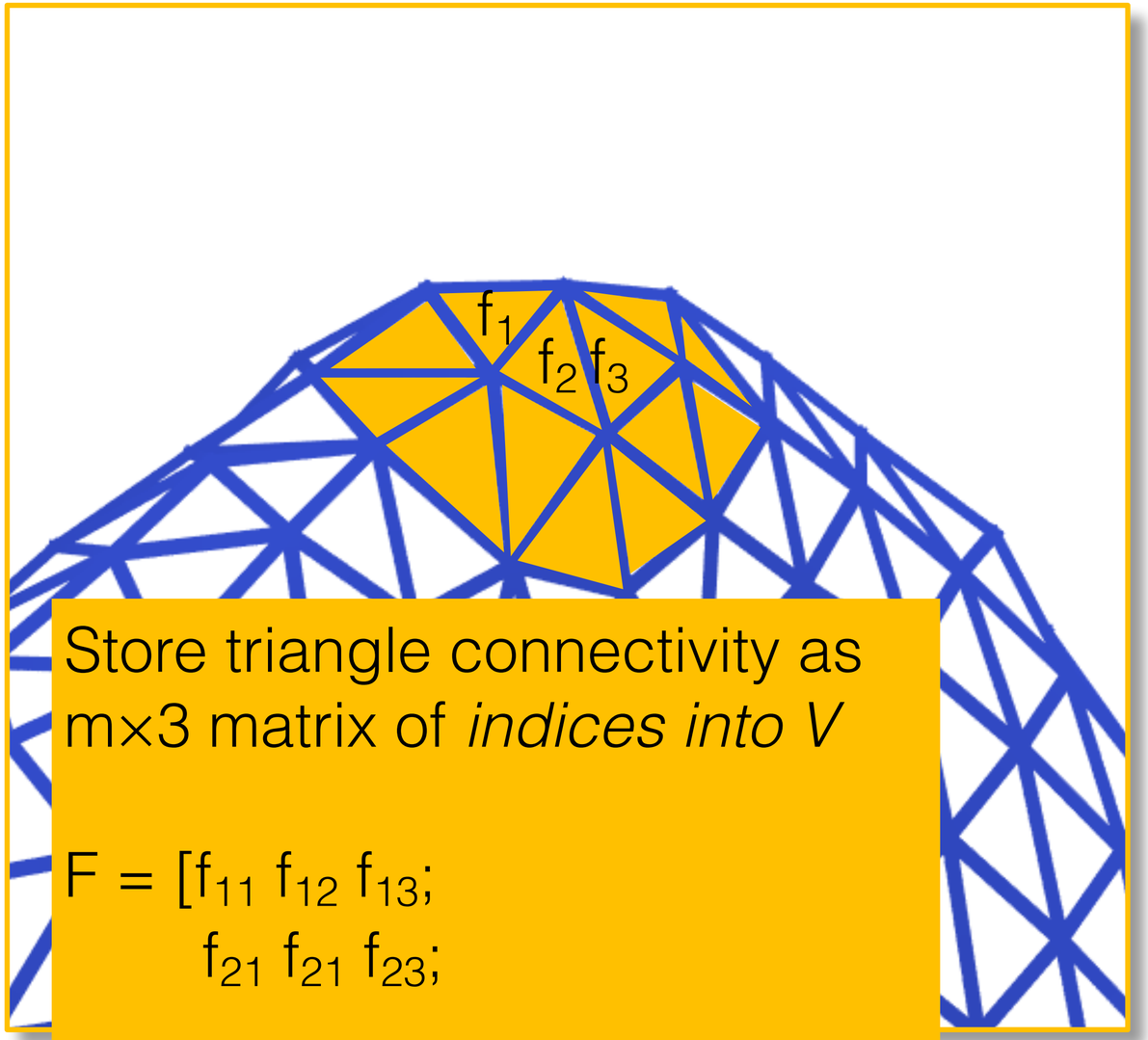
Triangle meshes discretize surfaces…





Triangle meshes discretize surfaces…





Store triangle connectivity as m×3 matrix of *indices into V*

$F = [f_{11}\ f_{12}\ f_{13};$
$f_{21}\ f_{21}\ f_{23};$
…
$f_{n1}\ f_{n2}\ f_{n3}]$

Why RAW matrices?

- Memory efficient and cache friendly
- Indices are simpler to debug than pointers
- Trivially copied and serialized
- **Interchangeable** with other libraries: **numpy**, pyTorch, Tensorflow, Scipy, MATLAB, OpenCV

Getting Started

Binder Demo