Deformable Simplicial Complex

Marek K. Misztal

Informatics and Mathematical Modelling, Technical University of Denmark
mkm@imm.dtu.dk

DSC 2011 Workshop
Kgs. Lyngby, 25th August 2011

Deformable models

- **Explicit** (or **Lagrangian**) methods:

Deformable models

- **Explicit** (or **Lagrangian**) methods:
  - use the parametrization of the interface,

Deformable models

- **Explicit** (or **Lagrangian**) methods:
  - use the parametrization of the interface,
  - perform the advection by solving an ODE:

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p}),$$

Deformable models

- **Explicit** (or **Lagrangian**) methods:
  - use the parametrization of the interface,
  - perform the advection by solving an ODE:

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p}),$$

  - require reparametrization, surgical cuts and collision detection in order to handle changes in the topology of the interface.

Deformable models

- **Explicit** (or **Lagrangian**) methods:
  - use the parametrization of the interface,
  - perform the advection by solving an ODE:

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p}),$$

  - require reparametrization, surgical cuts and collision detection in order to handle changes in the topology of the interface.
- **Implicit** (or **Eulerian**) methods:

Deformable models

- **Explicit** (or **Lagrangian**) methods:
  - use the parametrization of the interface,
  - perform the advection by solving an ODE:

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p}),$$

  - require reparametrization, surgical cuts and collision detection in order to handle changes in the topology of the interface.
- **Implicit** (or **Eulerian**) methods:
  - represent the n-dimensional interface as the 0-level set of a $(n+1)$-dimensional function $\varphi(x_1, x_2, \ldots, x_{n+1})$,

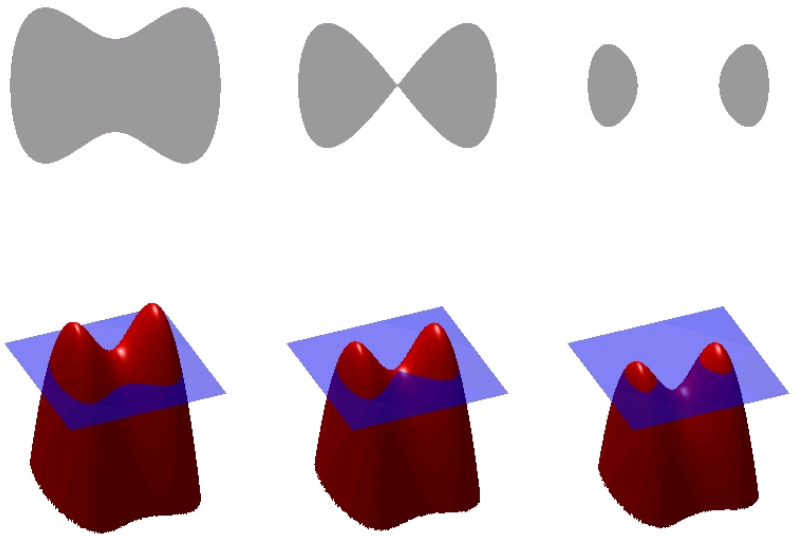
Deformable models

- **Explicit** (or **Lagrangian**) methods:
  - use the parametrization of the interface,
  - perform the advection by solving an ODE:

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p}),$$

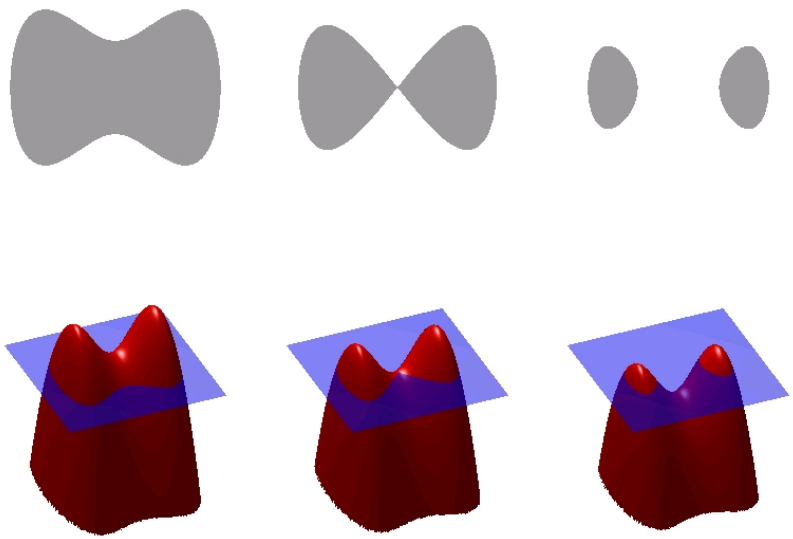
  - require reparametrization, surgical cuts and collision detection in order to handle changes in the topology of the interface.
- **Implicit** (or **Eulerian**) methods:
  - represent the n-dimensional interface as the 0-level set of a $(n+1)$-dimensional function $\varphi(x_1, x_2, \ldots, x_{n+1})$,
  - deformation is produced through evolution of the function φ, rather than the interface itself.

Level Set Method



The function is defined on the nodes of a rectangular grid.

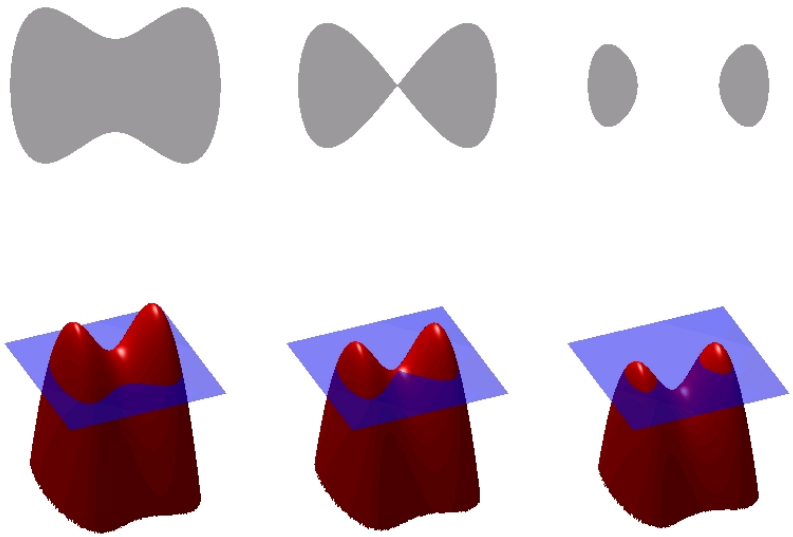
Level Set Method



Evolution of the interface due to the velocity field **v** is described by the following PDE (*level set equation*):

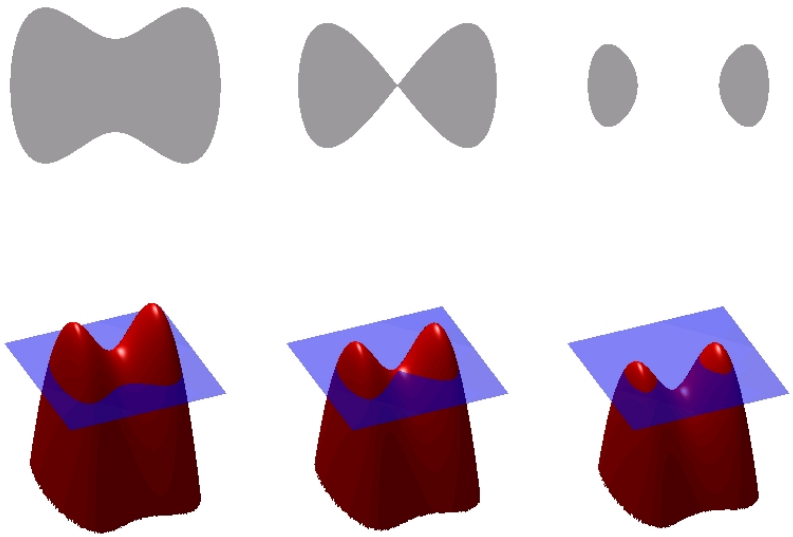
$$\frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi = 0.$$

Level Set Method



Signed distance function is a popular choice for φ.

Level Set Method



This approach provides trivial and robust topological adaptivity.

Drawbacks of the Level Set Method

- Bound to a certain scale (not scale adaptive).

Drawbacks of the Level Set Method

- Bound to a certain scale (not scale adaptive).
- Significant numerical diffusion (which results in troubles with preserving sharp details).

Drawbacks of the Level Set Method

- Bound to a certain scale (not scale adaptive).
- Significant numerical diffusion (which results in troubles with preserving sharp details).
- Lack of the explicit representation of the interface.

Drawbacks of the Level Set Method

- Bound to a certain scale (not scale adaptive).
- Significant numerical diffusion (which results in troubles with preserving sharp details).
- Lack of the explicit representation of the interface.
- Difficulty in representing multiple phases.

Drawbacks of the Level Set Method

- Bound to a certain scale (not scale adaptive).
- Significant numerical diffusion (which results in troubles with preserving sharp details).
- Lack of the explicit representation of the interface.
- Difficulty in representing multiple phases.
- Cannot be treated as a black box.

Drawbacks of the Level Set Method

- Bound to a certain scale (not scale adaptive).
- Significant numerical diffusion (which results in troubles with preserving sharp details).
- Lack of the explicit representation of the interface.
- Difficulty in representing multiple phases.
- Cannot be treated as a black box.
- Extra dimension needed.

Deformable simplicial complex

- The interface is represented **explicitly** as a piecewise linear curve (in 2D) or surface (in 3D).

Deformable simplicial complex

- The interface is represented **explicitly** as a piecewise linear curve (in 2D) or surface (in 3D).
- However, the embedding space is also a subject to a discretization (triangulation in 2D or tetrahedralization in 3D) fulfilling two conditions:

Deformable simplicial complex

- The interface is represented **explicitly** as a piecewise linear curve (in 2D) or surface (in 3D).
- However, the embedding space is also a subject to a discretization (triangulation in 2D or tetrahedralization in 3D) fulfilling two conditions:
  - *simplicial complex criterion* – the intersection of two simplices can be either empty or be their common face;

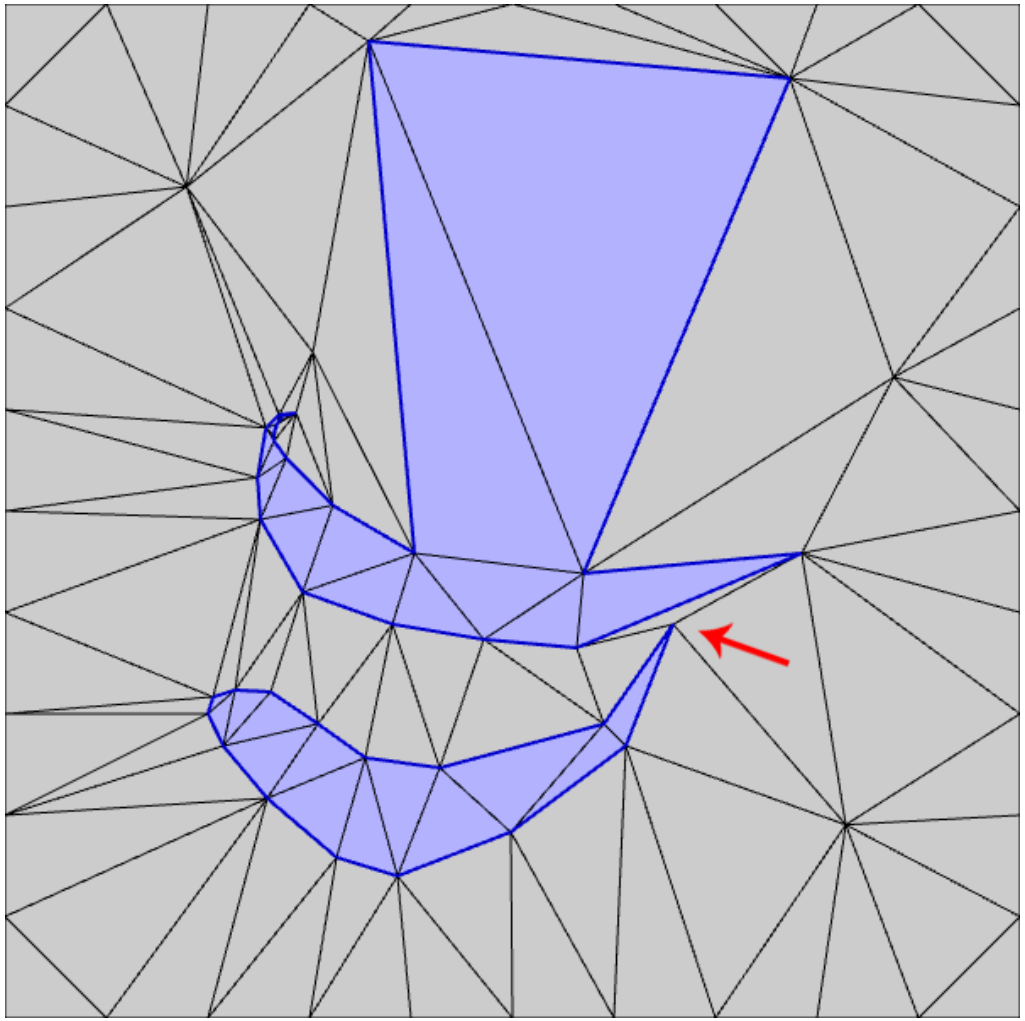
Deformable simplicial complex

- The interface is represented **explicitly** as a piecewise linear curve (in 2D) or surface (in 3D).
- However, the embedding space is also a subject to a discretization (triangulation in 2D or tetrahedralization in 3D) fulfilling two conditions:
  - *simplicial complex criterion* – the intersection of two simplices can be either empty or be their common face;
  - it conforms to the interface – the interface can be seen as defined **implicitly** as a set of boundary edges (faces) between the triangles (tetrahedra) marked as *outside* and *inside*.

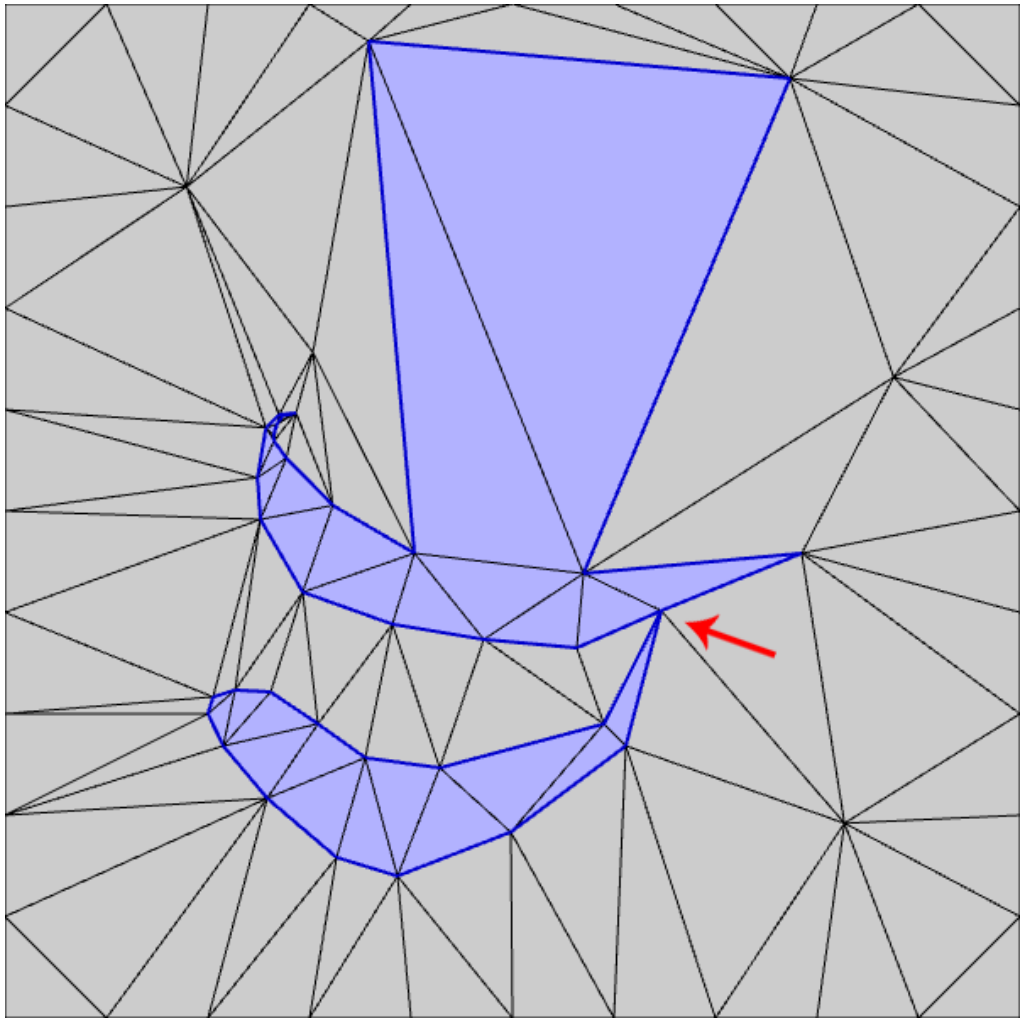
Deformable simplicial complex

- The interface is represented **explicitly** as a piecewise linear curve (in 2D) or surface (in 3D).
- However, the embedding space is also a subject to a discretization (triangulation in 2D or tetrahedralization in 3D) fulfilling two conditions:
  - *simplicial complex criterion* – the intersection of two simplices can be either empty or be their common face;
  - it conforms to the interface – the interface can be seen as defined **implicitly** as a set of boundary edges (faces) between the triangles (tetrahedra) marked as *outside* and *inside*.
- The interface deformation is performed via displacements of the interface vertices, while keeping the simplicial complex criterion of the underlying triangulation (tetrahedralization) all the time.

2D DSC



2D DSC



Algorithm Overview

1. In each iteration we compute new positions for the vertices of the interface, and attempt to displace them to the new positions, one ofter another.

Algorithm Overview

1. In each iteration we compute new positions for the vertices of the interface, and attempt to displace them to the new positions, one ofter another.
2. If the displacement does not invert any triangles (tetrahedra) in the 1-ring of the vertex, it is performed.

Algorithm Overview

1. In each iteration we compute new positions for the vertices of the interface, and attempt to displace them to the new positions, one ofter another.
2. If the displacement does not invert any triangles (tetrahedra) in the 1-ring of the vertex, it is performed.
3. Otherwise it is moved as far as possible (without inverting any triangles/tetrahedra) along the straight line connecting the old and the new positions.

Algorithm Overview

1. In each iteration we compute new positions for the vertices of the interface, and attempt to displace them to the new positions, one ofter another.
2. If the displacement does not invert any triangles (tetrahedra) in the 1-ring of the vertex, it is performed.
3. Otherwise it is moved as far as possible (without inverting any triangles/tetrahedra) along the straight line connecting the old and the new positions.
4. The displacement step is followed by a *mesh improvement step* aiming at improving the quality of the mesh and removing degenerate triangles (tetrahedra) created during displacement step.

Algorithm Overview

1. In each iteration we compute new positions for the vertices of the interface, and attempt to displace them to the new positions, one ofter another.
2. If the displacement does not invert any triangles (tetrahedra) in the 1-ring of the vertex, it is performed.
3. Otherwise it is moved as far as possible (without inverting any triangles/tetrahedra) along the straight line connecting the old and the new positions.
4. The displacement step is followed by a *mesh improvement step* aiming at improving the quality of the mesh and removing degenerate triangles (tetrahedra) created during displacement step.
5. Steps 3 and 4 are performed until all vertices are moved to the new positions.

Advantages of the Representation

- Little numerical diffusion.

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.
  - does not change gratuitously between time steps;

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.
  - does not change gratuitously between time steps;
- Adaptive resolution.

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.
  - does not change gratuitously between time steps;
- Adaptive resolution.
- Robust topological adaptivity.

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.
  - does not change gratuitously between time steps;
- Adaptive resolution.
- Robust topological adaptivity.
- Supports multiple phases.

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.
  - does not change gratuitously between time steps;
- Adaptive resolution.
- Robust topological adaptivity.
- Supports multiple phases.
- Simple, intrinsic collision detection mechanism.

Advantages of the Representation

- Little numerical diffusion.
- Explicit interface representation.
  - does not change gratuitously between time steps;
- Adaptive resolution.
- Robust topological adaptivity.
- Supports multiple phases.
- Simple, intrinsic collision detection mechanism.
- Allows for topology control.

Mesh improvement in 2D

In the 2D case, we attempt to keep the Delaunay property of the mesh. Mesh repair step involves the following operations:

Mesh improvement in 2D

In the 2D case, we attempt to keep the Delaunay property of the mesh. Mesh repair step involves the following operations:

- **Laplacian smoothing**: moving non-interface vertex towards the arycenter of its neighbors;

Mesh improvement in 2D

In the 2D case, we attempt to keep the Delaunay property of the mesh. Mesh repair step involves the following operations:

- **Laplacian smoothing**: moving non-interface vertex towards the arycenter of its neighbors;
- **edge flips**: for non-interface edges, not fulfilling the Delaunay property;

Mesh improvement in 2D

In the 2D case, we attempt to keep the Delaunay property of the mesh. Mesh repair step involves the following operations:

- **Laplacian smoothing**: moving non-interface vertex towards the arycenter of its neighbors;
- **edge flips**: for non-interface edges, not fulfilling the Delaunay property;
- **detail control**: edge collapse (for non-interface edges) and *needle* splits;

Mesh improvement in 2D

In the 2D case, we attempt to keep the Delaunay property of the mesh. Mesh repair step involves the following operations:

- **Laplacian smoothing**: moving non-interface vertex towards the arycenter of its neighbors;
- **edge flips**: for non-interface edges, not fulfilling the Delaunay property;
- **detail control**: edge collapse (for non-interface edges) and *needle* splits;
- **degenerate triangles removal**: *cap* flips, allowing the interface topology to change.

3D mesh

- The outline of the 3D DSC method resembled that of 2D DSC method, however some of the tools useful in the 2D case exhibit mediocre performance in 3D.

3D mesh

- The outline of the 3D DSC method resembled that of 2D DSC method, however some of the tools useful in the 2D case exhibit mediocre performance in 3D.
- Delaunay meshes, useful in the 2D case, tend to contain *slivers* (flat, nearly degenerate tetrahedra) unless special rules regarding to vertex placement apply.

3D mesh

- The outline of the 3D DSC method resembled that of 2D DSC method, however some of the tools useful in the 2D case exhibit mediocre performance in 3D.
- Delaunay meshes, useful in the 2D case, tend to contain *slivers* (flat, nearly degenerate tetrahedra) unless special rules regarding to vertex placement apply.
- Laplacian smoothing may produce inverted tetrahedra in 3D despite reasonable performance in 2D.

Quality measure

Instead of using the Delaunay quality measure of a tetrahedron (minimum solid angle) in mesh quality improvement operations, we aimed for a quality measure which would penalize both flat tetrahedra, characterized by near-zero volume despite the fact that the distances between their vertices (edge lengths) can be large.

We have chosen to improve the volume-length ratio of a tetrahedron:

$$Q(\sigma) = 6\sqrt{2}\frac{V(\sigma)}{l_{rms}^3},$$

where $V(t)$ is the oriented volume of a tetrahedron t, and l_{rms} is the average (root-mean-squared) of the lengths of its edges:

$$l_{\mathrm{rms}} = \sqrt{\frac{l_{12}^2 + l_{13}^2 + l_{14}^2 + l_{23}^2 + l_{24}^2 + l_{34}^2}{6}}.$$

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.
- It contains the following types of operations:

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.
- It contains the following types of operations:
  - smoothing;

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.
- It contains the following types of operations:
  - smoothing;
  - reconnection;

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.
- It contains the following types of operations:
  - smoothing;
  - reconnection;
  - detail control through edge splits and edge collapses;

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.
- It contains the following types of operations:
  - smoothing;
  - reconnection;
  - detail control through edge splits and edge collapses;
  - degenerate tetrahedron removal, through relabelling and more aggressive methods;

Mesh improvement step

- Mesh improvement step aims at improving mesh quality defined as the volume-length ratio.
- It contains the following types of operations:
  - smoothing;
  - reconnection;
  - detail control through edge splits and edge collapses;
  - degenerate tetrahedron removal, through relabelling and more aggressive methods;
- In order to optimize the performance, we only perform one pass of improvement for the whole mesh per iteration. In other vertex displacement passes, we only perform improvement for the set of tetrahedra adjacent to the vertices, which have not arrived at their final positions yet.

Smoothing

- Laplacian smoothing, useful in 2D meshes, does not work quite as well in the 3D case – it tends to produce inverted tetrahedra.

Smoothing

- Laplacian smoothing, useful in 2D meshes, does not work quite as well in the 3D case – it tends to produce inverted tetrahedra.
- We are using **smart Laplacian smoothing** (where the displacement is only performed if it improves the quality locally) performs much better, however it sometimes fails to improve the mesh quality significantly.

Smoothing

- Laplacian smoothing, useful in 2D meshes, does not work quite as well in the 3D case – it tends to produce inverted tetrahedra.
- We are using **smart Laplacian smoothing** (where the displacement is only performed if it improves the quality locally) performs much better, however it sometimes fails to improve the mesh quality significantly.
- If that is the case and the minimum quality in the star of a vertex v is lower than 0.1, we perform **optimization-based smoothing** (by L. Freitag et al.), which tries to maximize the minimum quality in the star of v. This is clearly a non-smooth optimization problem.

Topological operations

- The topological operations (or *flips*, or *swaps*) are generalizations of the edge flip in the 2D case.

Topological operations

- The topological operations (or *flips*, or *swaps*) are generalizations of the edge flip in the 2D case.
- They aim at reconnecting the mesh without changing vertex placement.

Topological operations

- The topological operations (or *flips*, or *swaps*) are generalizations of the edge flip in the 2D case.
- They aim at reconnecting the mesh without changing vertex placement.
- There are several ways of generalizing edge flip to 3D meshes, including:

Topological operations

- The topological operations (or *flips*, or *swaps*) are generalizations of the edge flip in the 2D case.
- They aim at reconnecting the mesh without changing vertex placement.
- There are several ways of generalizing edge flip to 3D meshes, including:
  - edge removal,

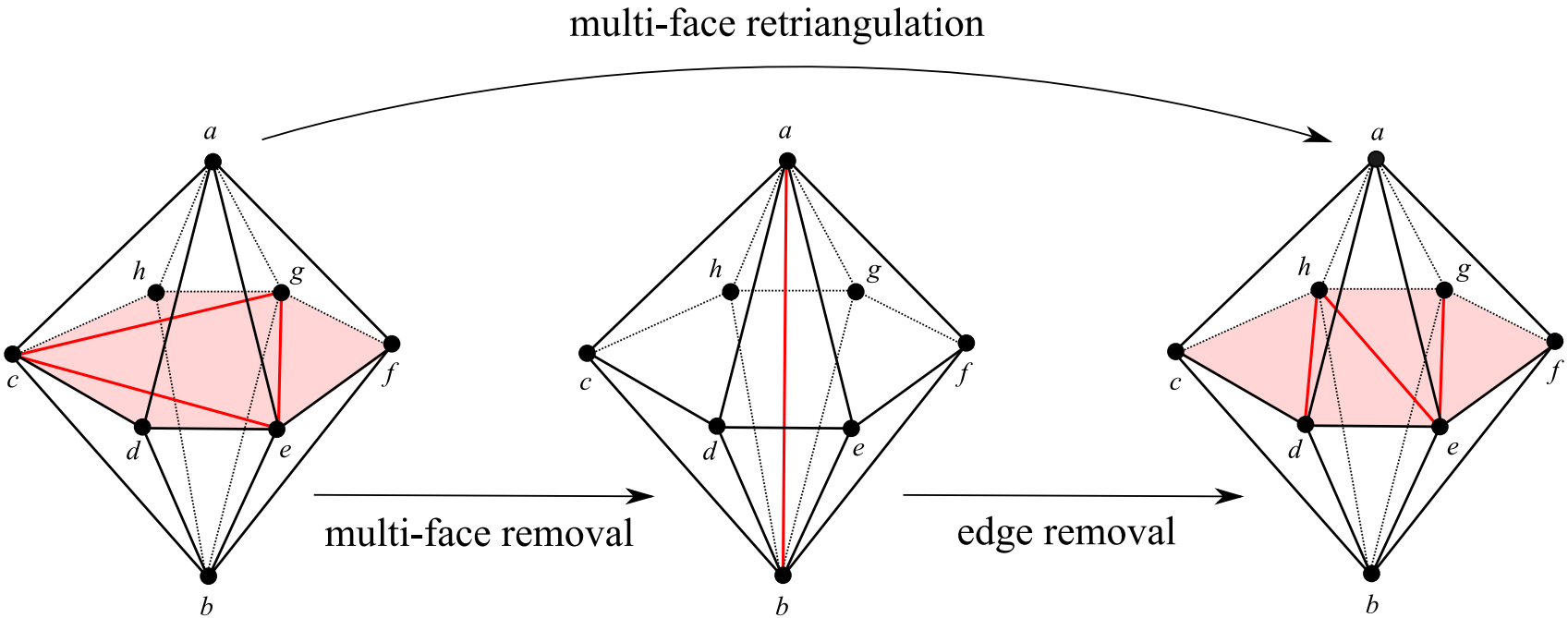
Topological operations

- The topological operations (or *flips*, or *swaps*) are generalizations of the edge flip in the 2D case.
- They aim at reconnecting the mesh without changing vertex placement.
- There are several ways of generalizing edge flip to 3D meshes, including:
  - edge removal,
  - multi-face removal,

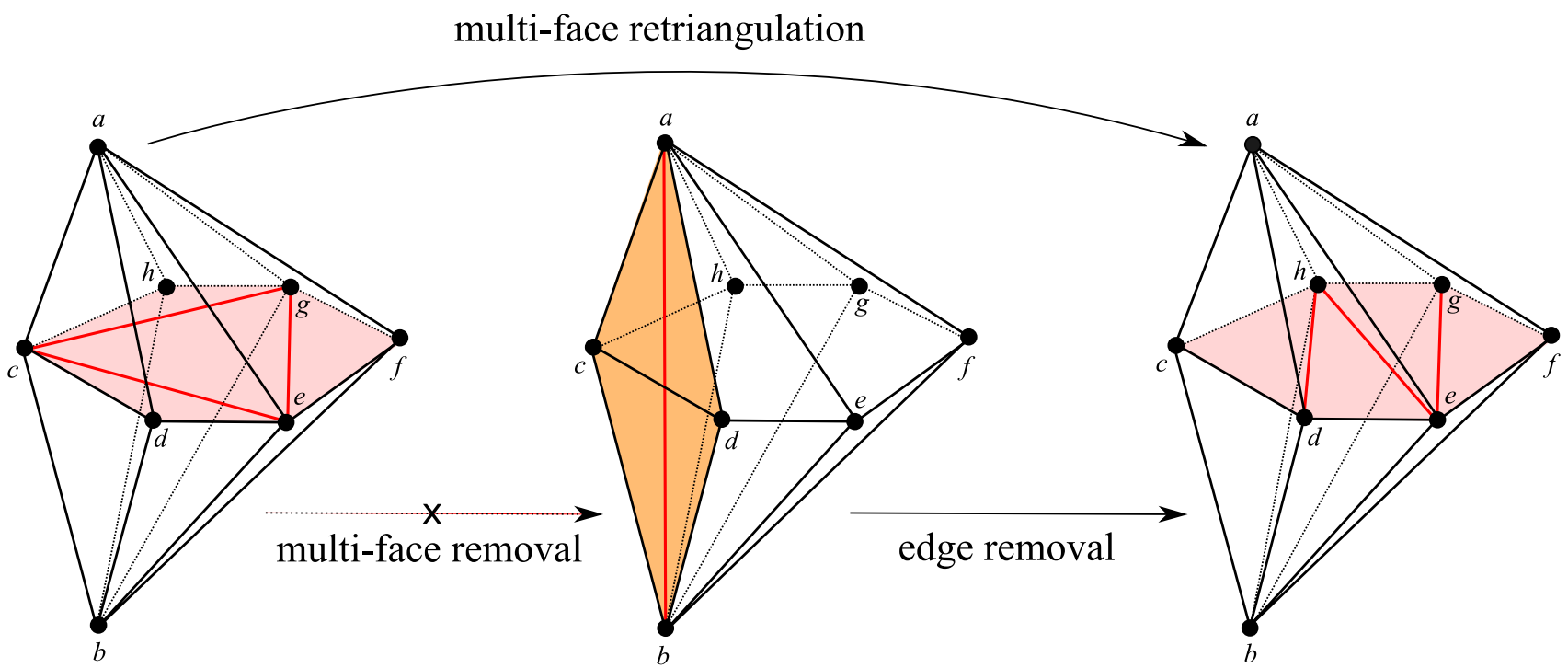
Topological operations

- The topological operations (or *flips*, or *swaps*) are generalizations of the edge flip in the 2D case.
- They aim at reconnecting the mesh without changing vertex placement.
- There are several ways of generalizing edge flip to 3D meshes, including:
  - edge removal,
  - multi-face removal,
  - multi-face retriangulation.

Topological operations (reconnection)



Multi-face retriangulation



Topological operations

- Edge removal requires to select the final triangulation of the link of the removed edge. The selection is made using *Klincsek's algorithm*: a dynamic programming method which maximizes the minimum quality over all tetrahedra produced through this triangulation.

Topological operations

- Edge removal requires to select the final triangulation of the link of the removed edge. The selection is made using *Klincsek's algorithm*: a dynamic programming method which maximizes the minimum quality over all tetrahedra produced through this triangulation.
- Analogously MFRT is also using *Klincsek's algoritm* in order to determine the final triangulation of the polygon "sandwiched" between two apices.

Topological operations

- Edge removal requires to select the final triangulation of the link of the removed edge. The selection is made using *Klincsek's algorithm*: a dynamic programming method which maximizes the minimum quality over all tetrahedra produced through this triangulation.
- Analogously MFRT is also using *Klincsek's algoritm* in order to determine the final triangulation of the polygon "sandwiched" between two apices.
- Multi-face removal requires finding the initial set of tetrahedra to be removed. This is also done using dynamic programming.

Topological pass

- **Input:** a set of tetrahedra.

Topological pass

- **Input:** a set of tetrahedra.
- For each tetrahedron in the input set, if it has not been removed yet and its quality is less than 0.2:

Topological pass

- **Input:** a set of tetrahedra.
- For each tetrahedron in the input set, if it has not been removed yet and its quality is less than 0.2:
  1. Attempt to remove each edge using edge removal. Edge removal is only performed if it increases the minimum quality over the set of tetrahedra affected by the operation.

Topological pass

- **Input:** a set of tetrahedra.
- For each tetrahedron in the input set, if it has not been removed yet and its quality is less than 0.2:
  1. Attempt to remove each edge using edge removal. Edge removal is only performed if it increases the minimum quality over the set of tetrahedra affected by the operation.
  2. Attempt to remove each face using multi-face retriangulation and multi-face removal. Either of this operation is only performed if it increases the minimum quality over the set of tetrahedra affected by the operation.

Detail control

The purpose of detail control is to ensure that the embedding mesh has enough Steiner vertices, so that it is flexible enough to accomodate changes to the interface and, on the other hand, to make sure that the number of Steiner vertices does not increase too much, in order to keep the performance efficient.

We utilize two operations:

Detail control

The purpose of detail control is to ensure that the embedding mesh has enough Steiner vertices, so that it is flexible enough to accomodate changes to the interface and, on the other hand, to make sure that the number of Steiner vertices does not increase too much, in order to keep the performance efficient.

We utilize two operations:

- **edge split**: for long edges connecting two interface vertices, or an interface vertex with a boundary vertex;

Detail control

The purpose of detail control is to ensure that the embedding mesh has enough Steiner vertices, so that it is flexible enough to accomodate changes to the interface and, on the other hand, to make sure that the number of Steiner vertices does not increase too much, in order to keep the performance efficient.

We utilize two operations:

- **edge split**: for long edges connecting two interface vertices, or an interface vertex with a boundary vertex;
- **edge collapse**: for short edges, not connected to either the interface or the boundary of the embedding mesh, as long as it increases the minimum quality locally or does not decrease it below a certain threshold value.

Interface topology changes

Whenever two parts of the interface are about to collide, nearly degenerate tetrahedra begin to appear in the mesh. We are producing interface topology changes by removing those tetrahedra. We do it in two ways;

Interface topology changes

Whenever two parts of the interface are about to collide, nearly degenerate tetrahedra begin to appear in the mesh. We are producing interface topology changes by removing those tetrahedra. We do it in two ways;

- *tetrahedron re-labelling*: switching the label of a tetrahedron from *outside* to *inside* (or the other way round) if it is “squeezed” between two parts of the interface.

Interface topology changes

Whenever two parts of the interface are about to collide, nearly degenerate tetrahedra begin to appear in the mesh. We are producing interface topology changes by removing those tetrahedra. We do it in two ways;

- *tetrahedron re-labelling*: switching the label of a tetrahedron from *outside* to *inside* (or the other way round) if it is “squeezed” between two parts of the interface.
- *degeneracy removal*: removing nearly flat tetrahedra by “flattening” them and replacing with 4, 3 or 2 faces. Similarily, we remove degenerate (nearly colinear) faces and degenerate (very short) edges.

Surface mesh improvement

- It turns out, in the 3D case, velocity field computation often depends on the quality of the surface mesh. Hence, we need to keep the mesh quality high.

Surface mesh improvement

- It turns out, in the 3D case, velocity field computation often depends on the quality of the surface mesh. Hence, we need to keep the mesh quality high.
- In order to do so, we perform the following operations on the interface:

Surface mesh improvement

- It turns out, in the 3D case, velocity field computation often depends on the quality of the surface mesh. Hence, we need to keep the mesh quality high.
- In order to do so, we perform the following operations on the interface:
  - *geometry preserving smoothing*: null-space smoothing, moving each interface, manifold vertex only in the null space of its local quadric metric tensor.

Surface mesh improvement

- It turns out, in the 3D case, velocity field computation often depends on the quality of the surface mesh. Hence, we need to keep the mesh quality high.
- In order to do so, we perform the following operations on the interface:
  - *geometry preserving smoothing*: null-space smoothing, moving each interface, manifold vertex only in the null space of its local quadric metric tensor.
  - *edge flips*: for non-feature edges not fulfilling the Delaunay criterion (this requires reconnection of the embedding tetrahedral mesh through *edge removal*).

Surface mesh improvement

- It turns out, in the 3D case, velocity field computation often depends on the quality of the surface mesh. Hence, we need to keep the mesh quality high.
- In order to do so, we perform the following operations on the interface:
  - *geometry preserving smoothing*: null-space smoothing, moving each interface, manifold vertex only in the null space of its local quadric metric tensor.
  - *edge flips*: for non-feature edges not fulfilling the Delaunay criterion (this requires reconnection of the embedding tetrahedral mesh through *edge removal*).
  - *edge collapse*: for edges shorter than a given threshold,

Surface mesh improvement

- It turns out, in the 3D case, velocity field computation often depends on the quality of the surface mesh. Hence, we need to keep the mesh quality high.
- In order to do so, we perform the following operations on the interface:
  - *geometry preserving smoothing*: null-space smoothing, moving each interface, manifold vertex only in the null space of its local quadric metric tensor.
  - *edge flips*: for non-feature edges not fulfilling the Delaunay criterion (this requires reconnection of the embedding tetrahedral mesh through *edge removal*).
  - *edge collapse*: for edges shorter than a given threshold,
  - *edge splits*: for edges longer than a given threshold.

Implementation

- The 2D DSC implementation is built on top of the *half-edge* data structure available in the GEL library.

Implementation

- The 2D DSC implementation is built on top of the *half-edge* data structure available in the GEL library.
- The 3D DSC implementation relies on our own C++ implementation of the Incidence Simplicial data structure for 3D simplicial complexes by de Floriani et al.

Implementation

- The 2D DSC implementation is built on top of the *half-edge* data structure available in the GEL library.
- The 3D DSC implementation relies on our own C++ implementation of the Incidence Simplicial data structure for 3D simplicial complexes by de Floriani et al.
- In its current form, the 3D DSC algorithm requires a triangle mesh as an input. This triangle mesh is then normalized and tetrahedralized on the inside and on the outside using TetGen (the outside mesh is bounded by a sparsly subdivided $(-1;-1;-1) \times (1;1;1)$ box).

Simple geometric flows

- **Rotation:** produced by multiplying vertex positions by a rotation matrix $\mathbf{M}(\mathbf{e}, \theta)$, where $\mathbf{e}$ is the rotation axis and θ is a small angle.

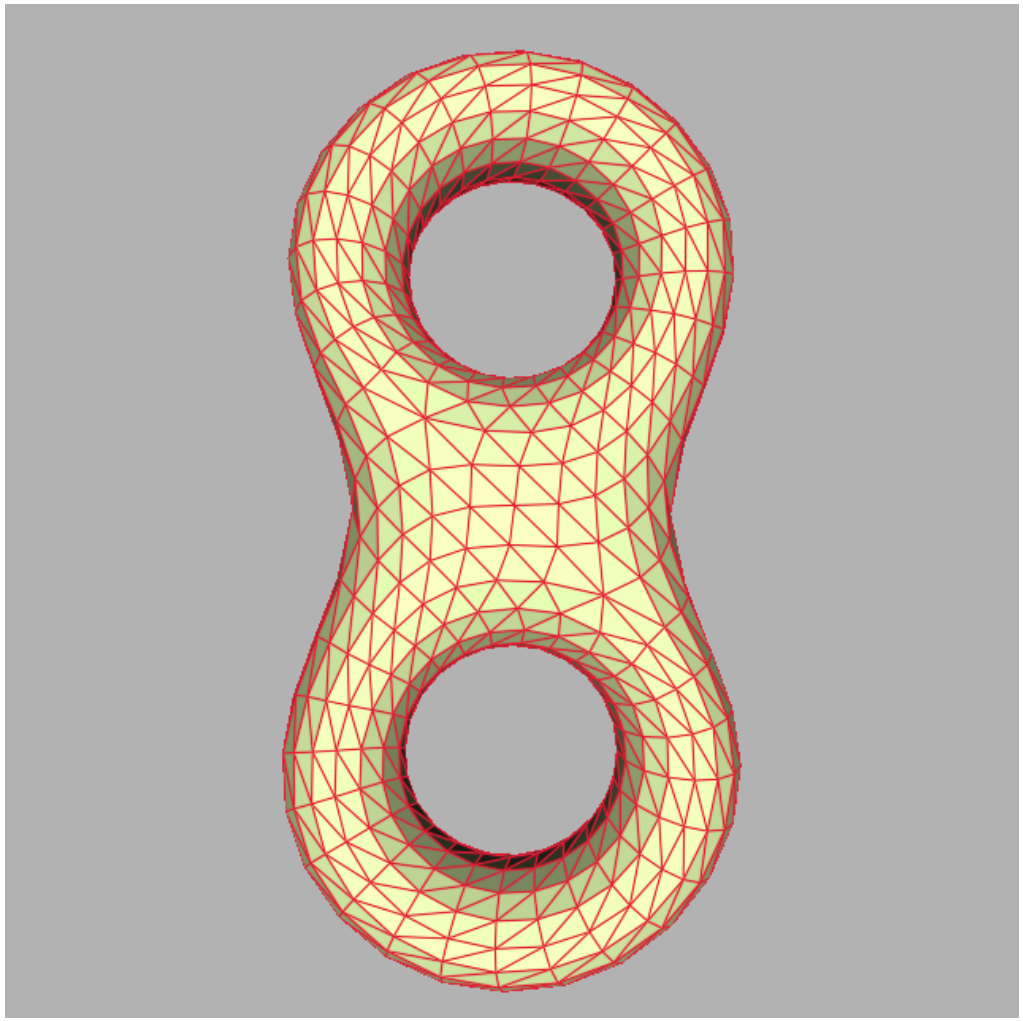
Simple geometric flows

- **Rotation:** produced by multiplying vertex positions by a rotation matrix $\mathbf{M}(\mathbf{e}, \theta)$, where $\mathbf{e}$ is the rotation axis and θ is a small angle.
- **Mean curvature flow:** computed using the *cotangent formula*.

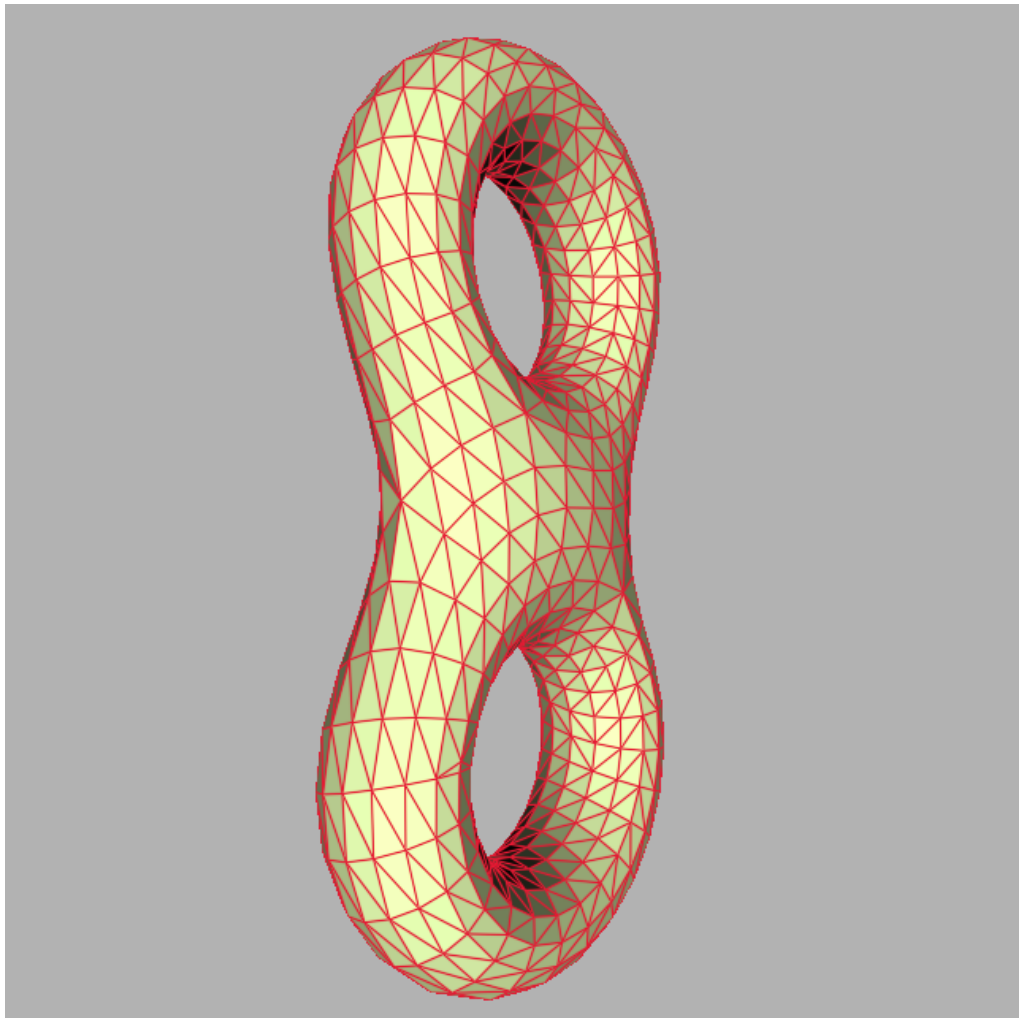
Simple geometric flows

- **Rotation:** produced by multiplying vertex positions by a rotation matrix $\mathbf{M}(\mathbf{e}, \theta)$, where $\mathbf{e}$ is the rotation axis and θ is a small angle.
- **Mean curvature flow:** computed using the *cotangent formula*.
- **Offsetting:** performed through face offsetting, rather than displacing the vertices by a constant distance in the approximate normal direction.

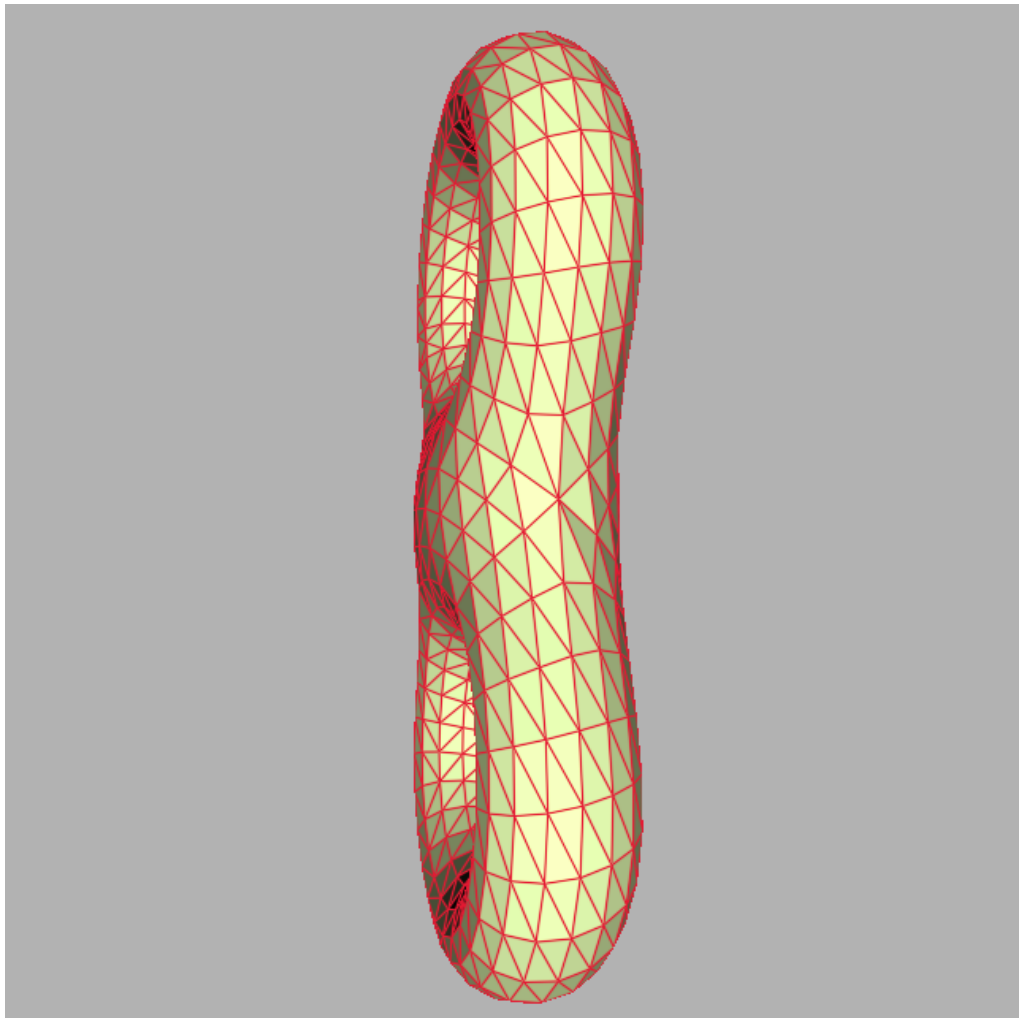
Rotation



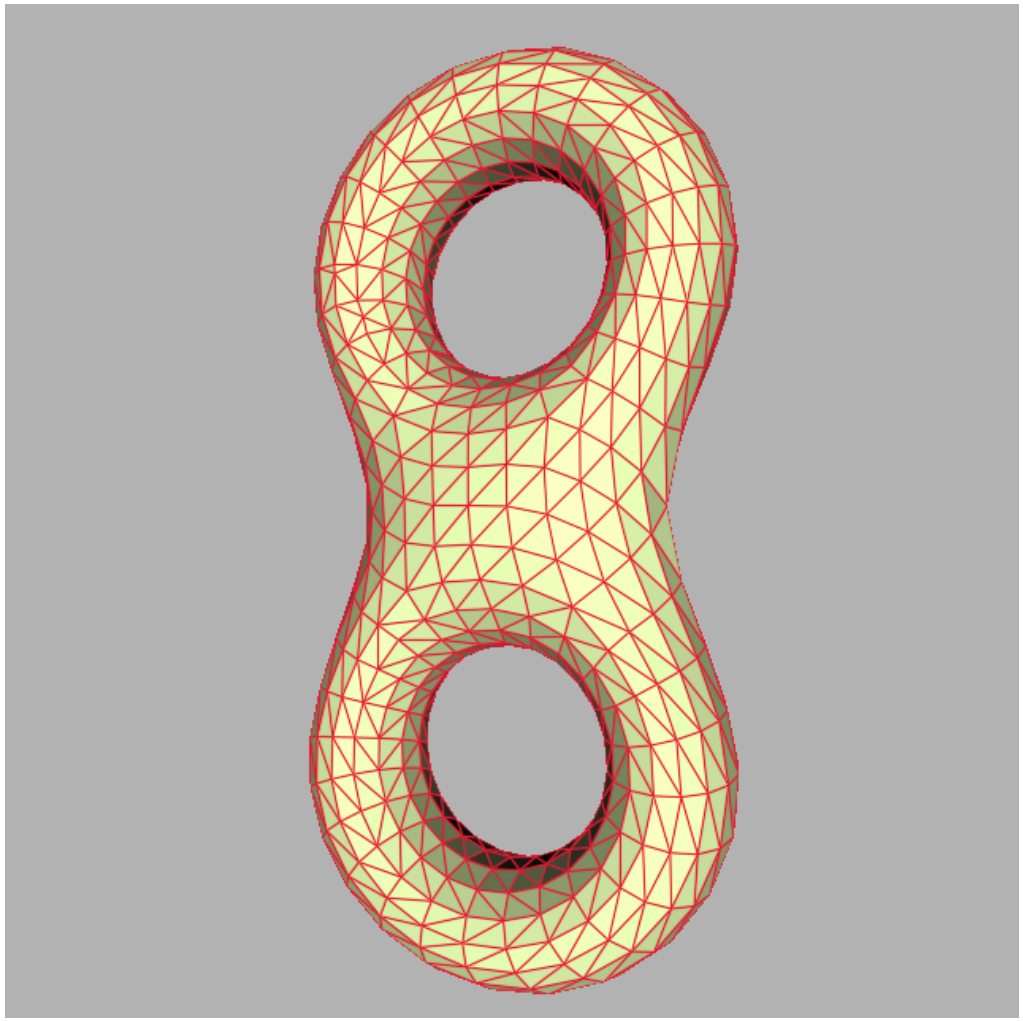
Rotation



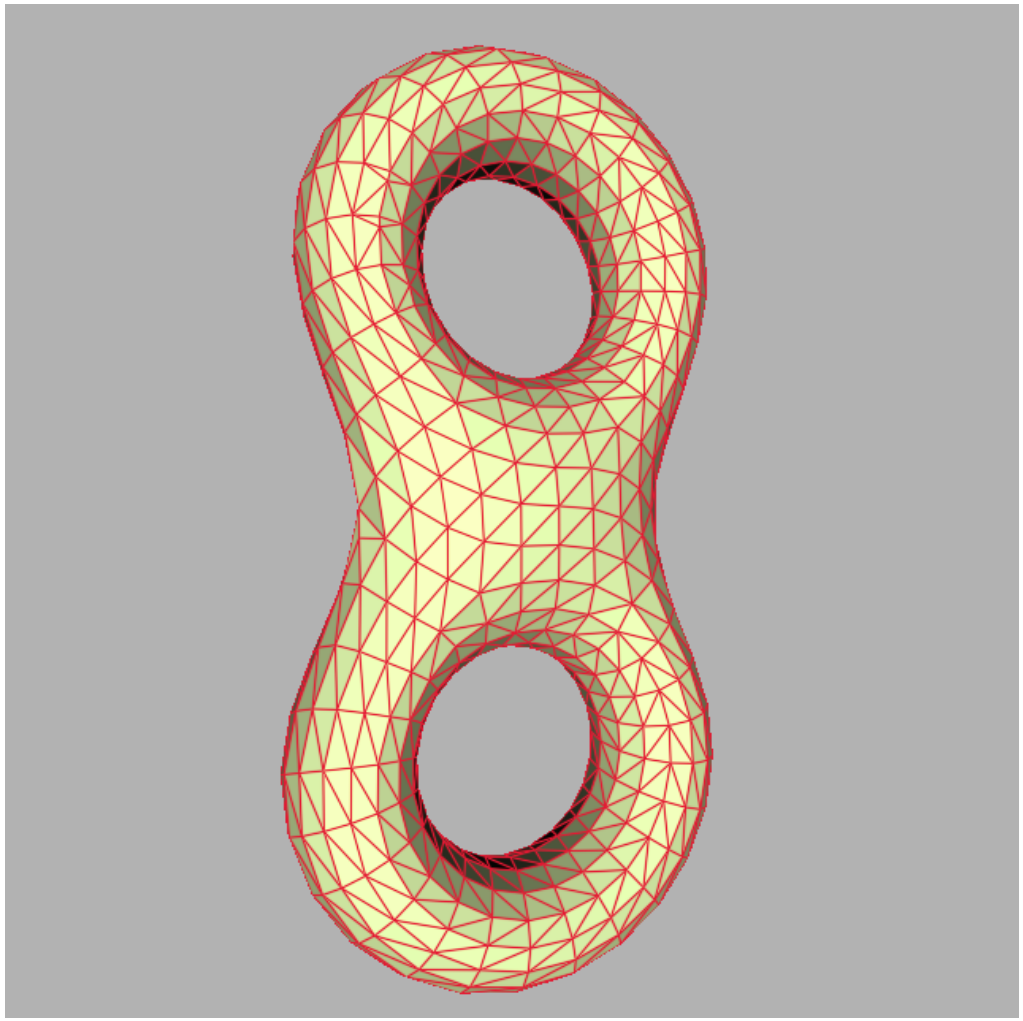
Rotation



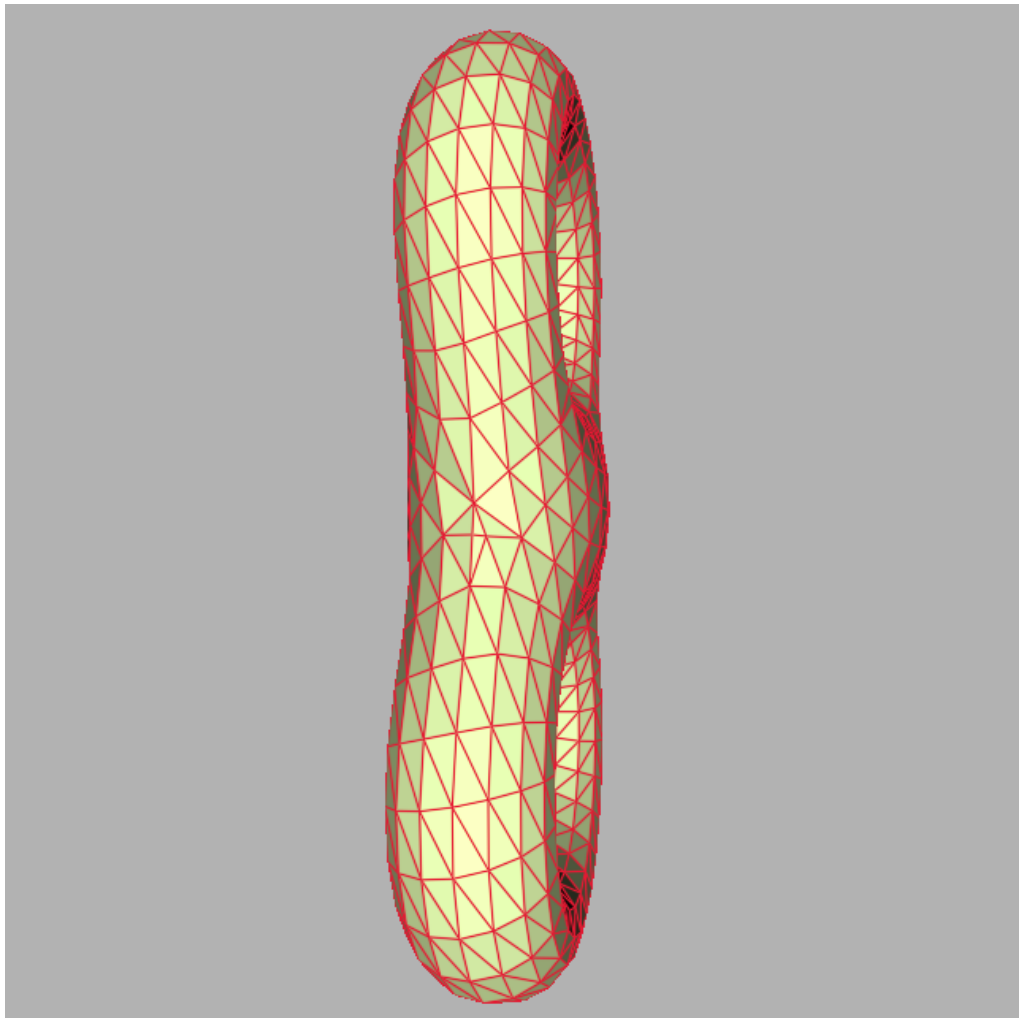
Rotation



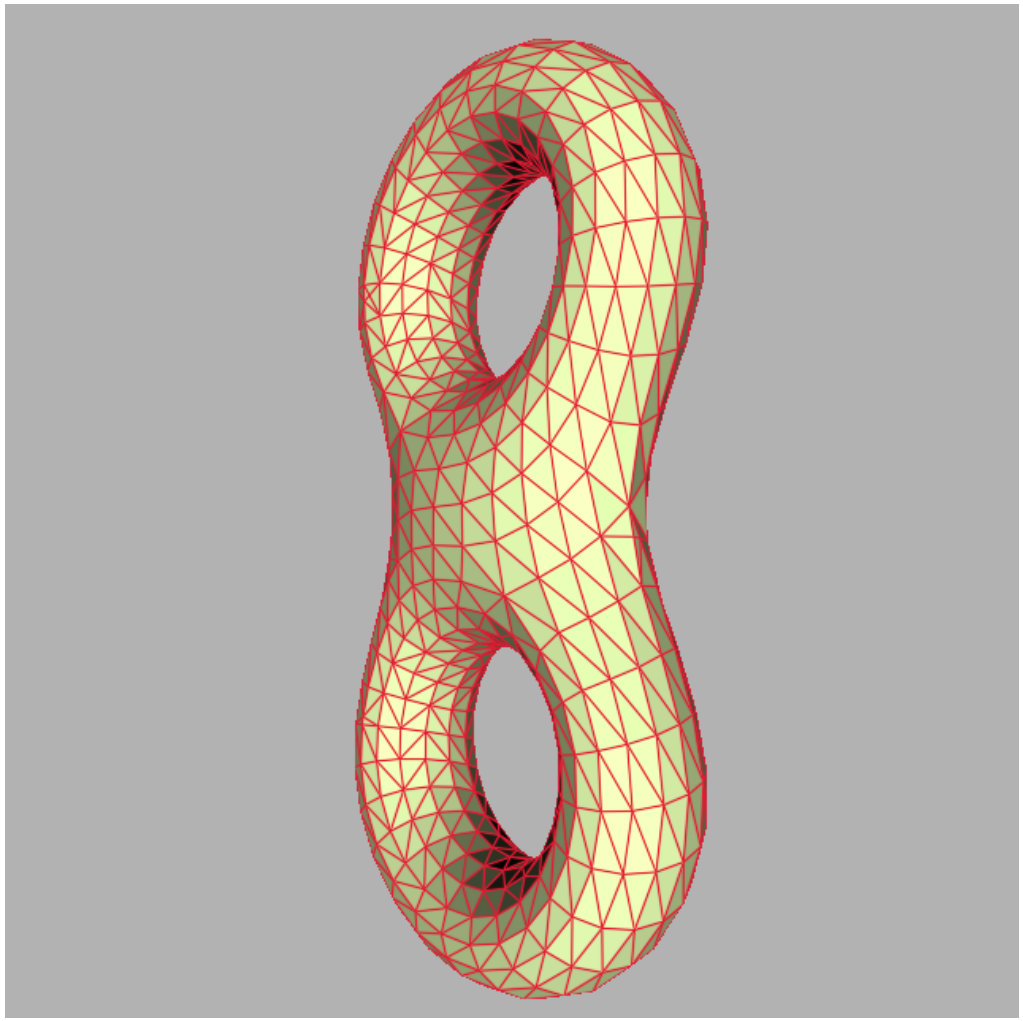
Rotation



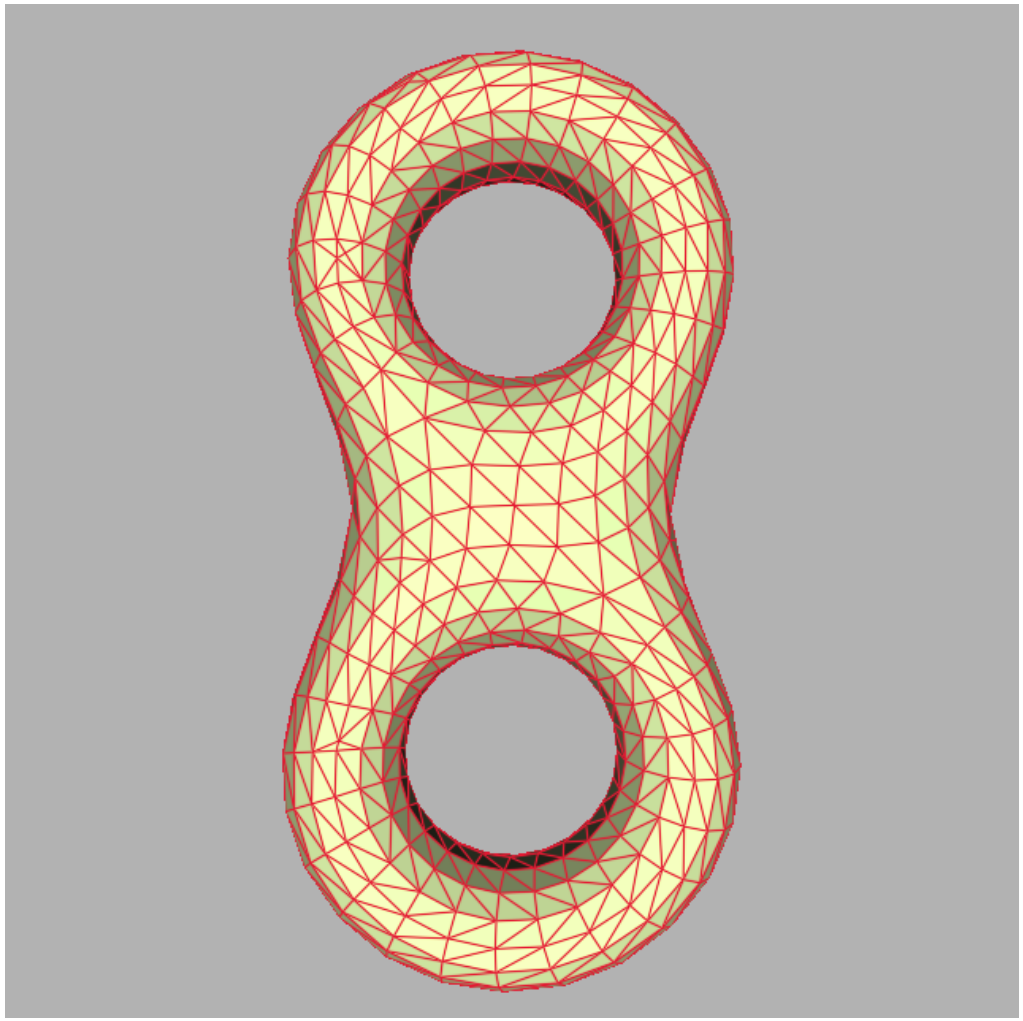
Rotation



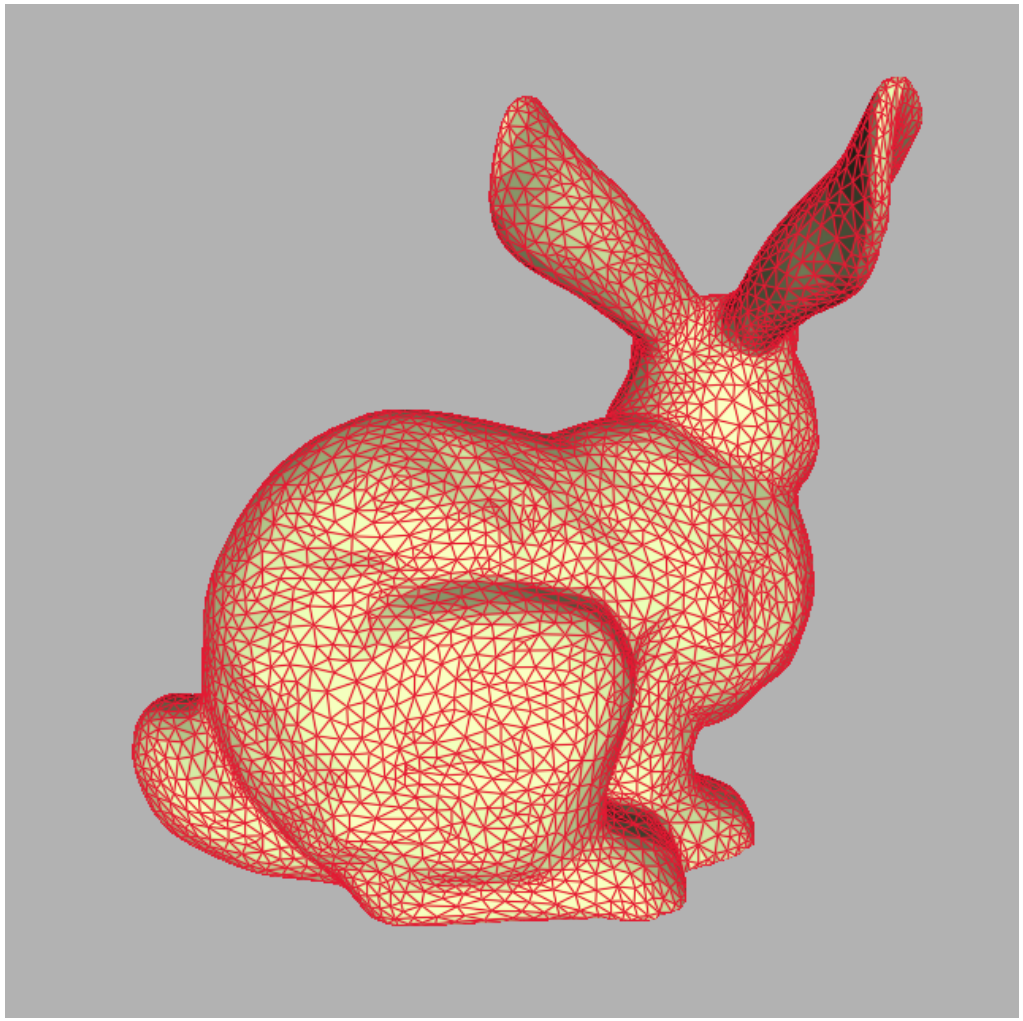
Rotation



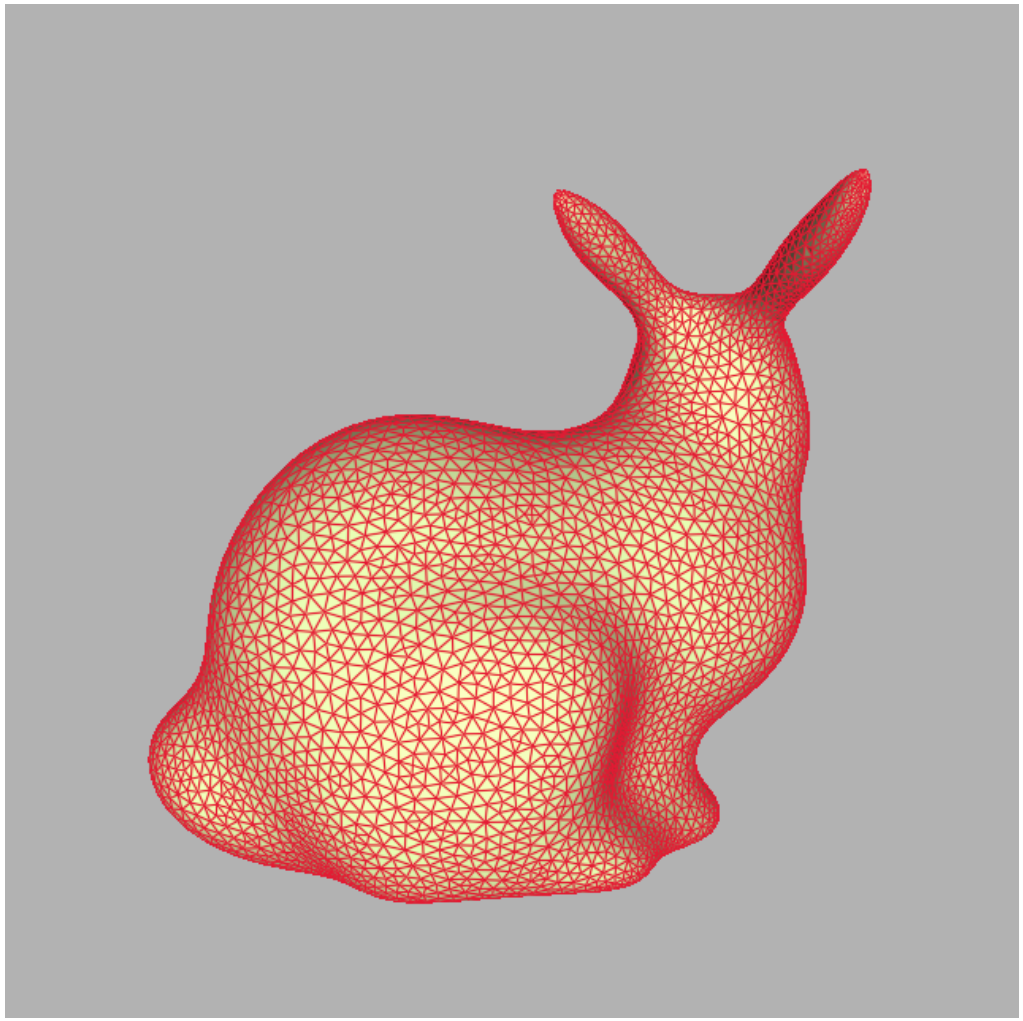
Rotation



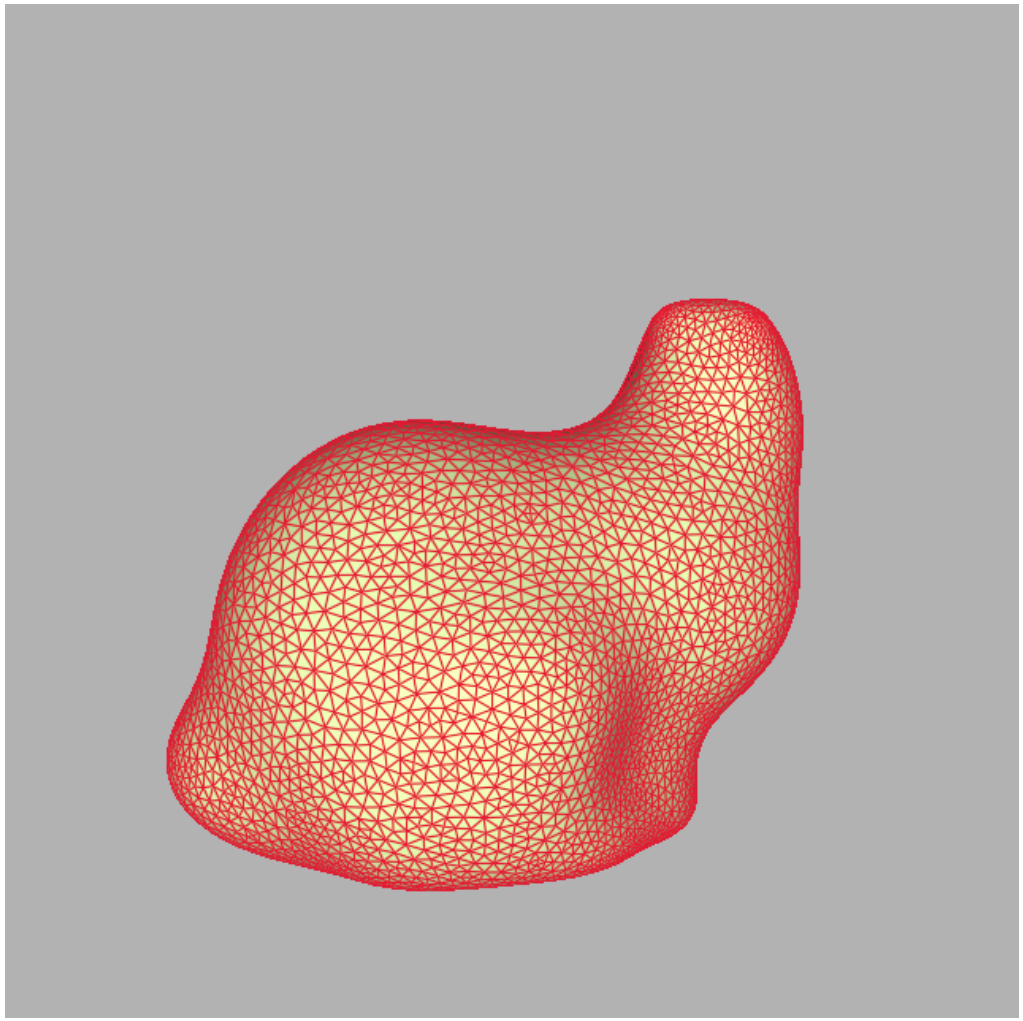
Mean curvature flow



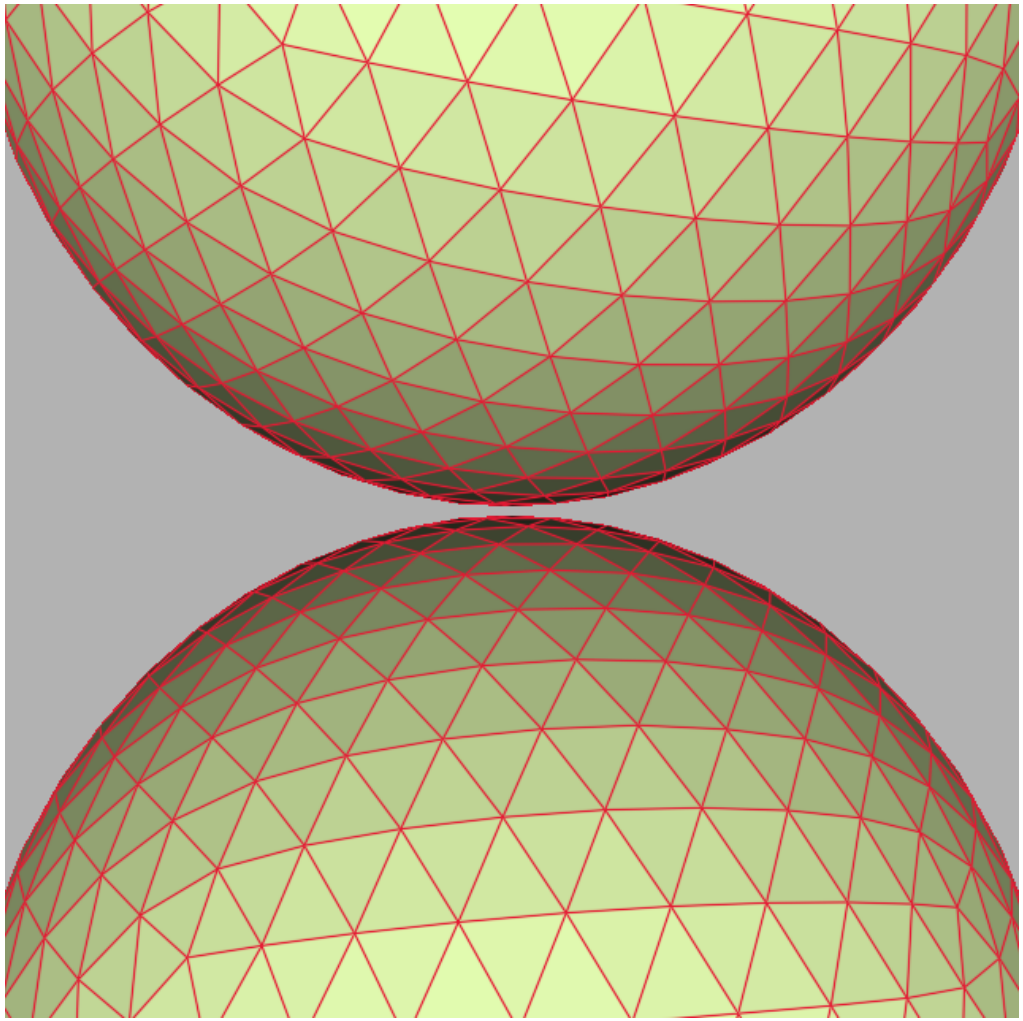
Mean curvature flow



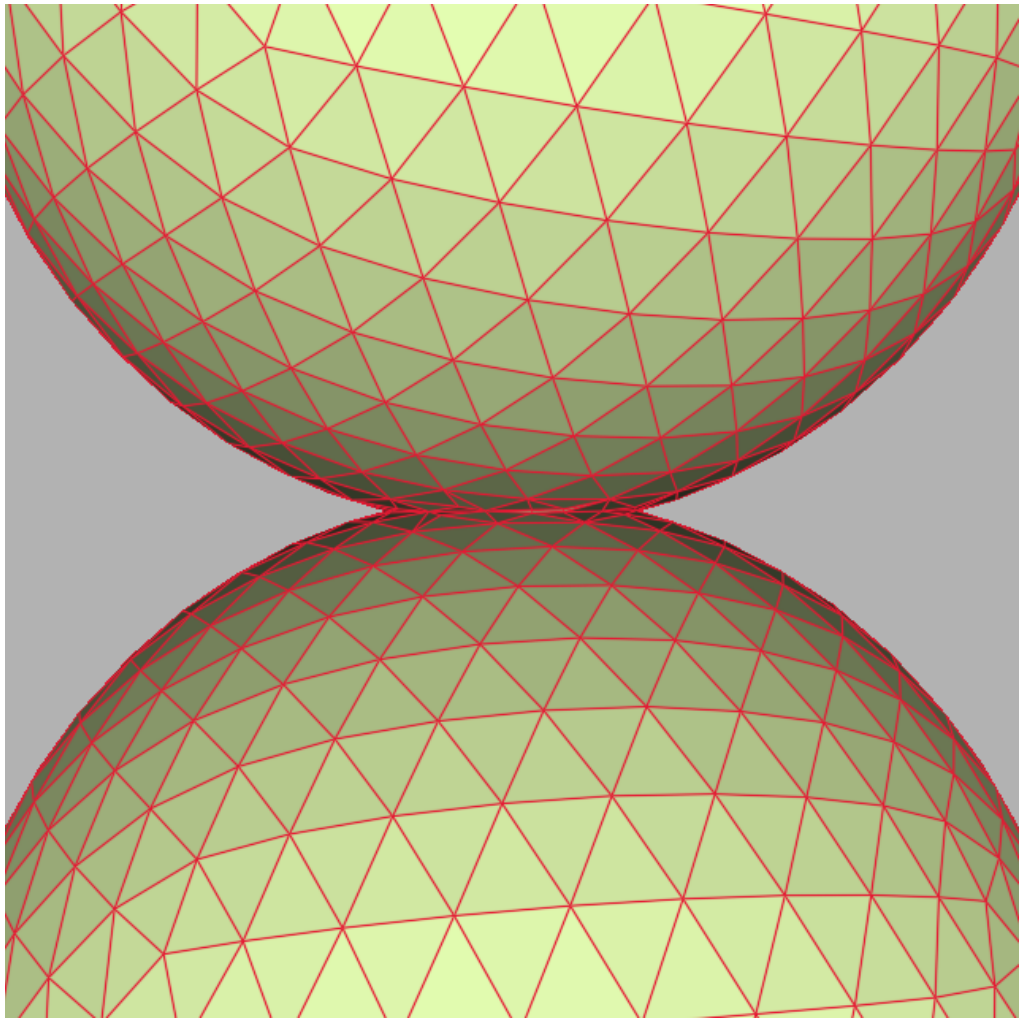
Mean curvature flow



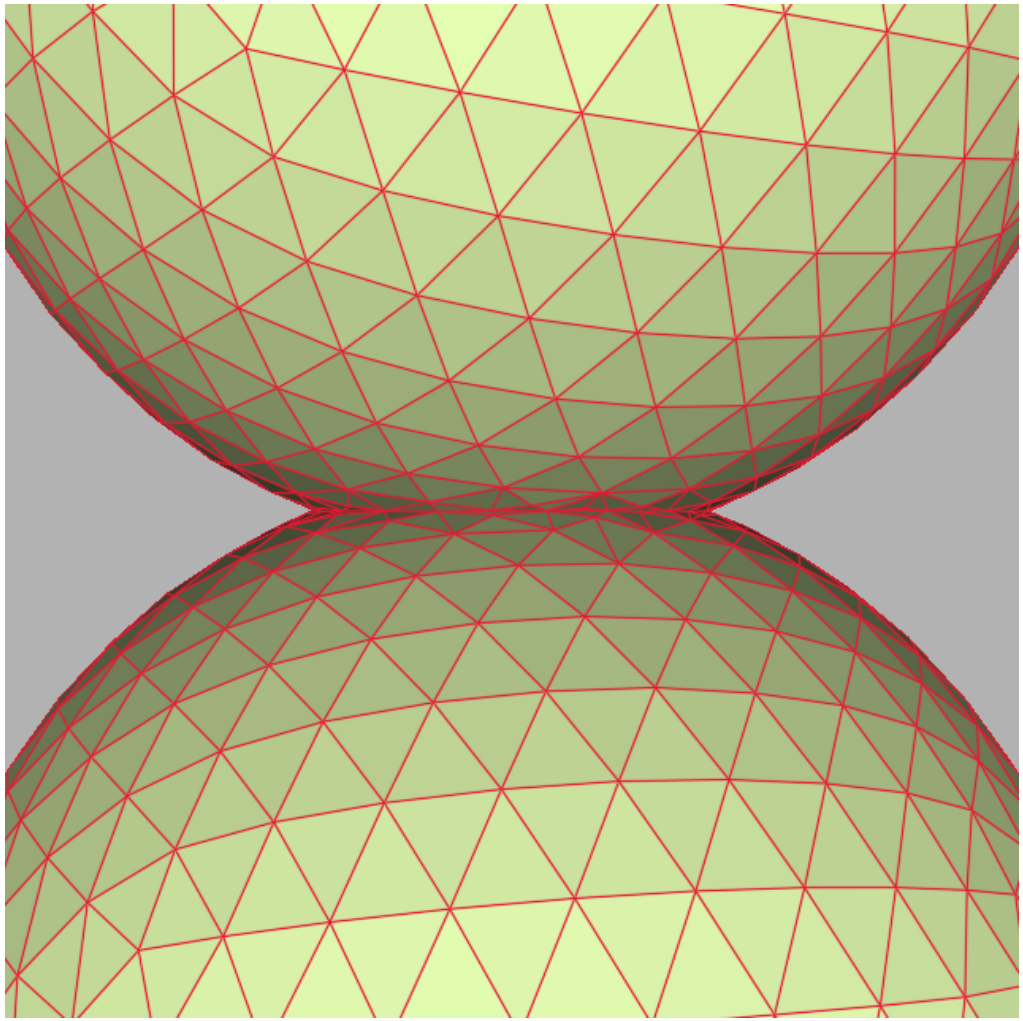
Offsetting



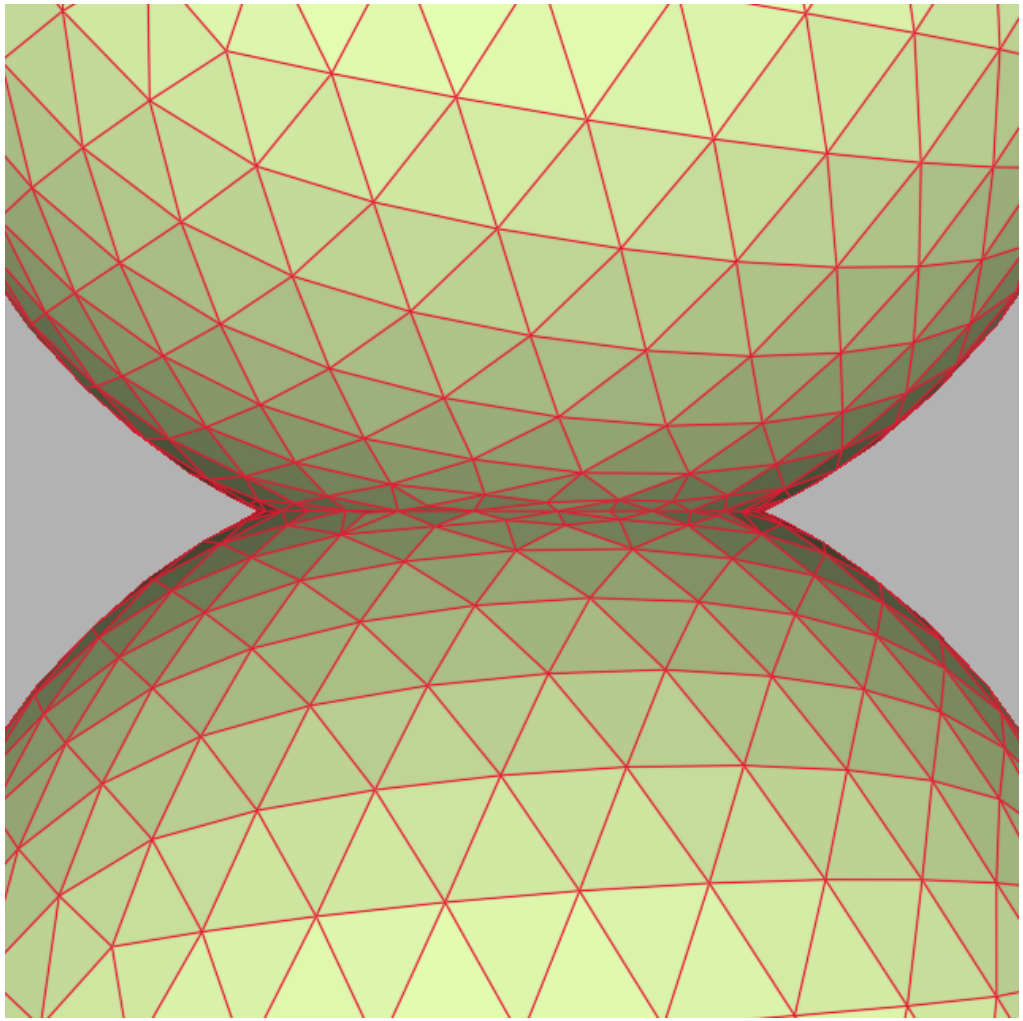
Offsetting



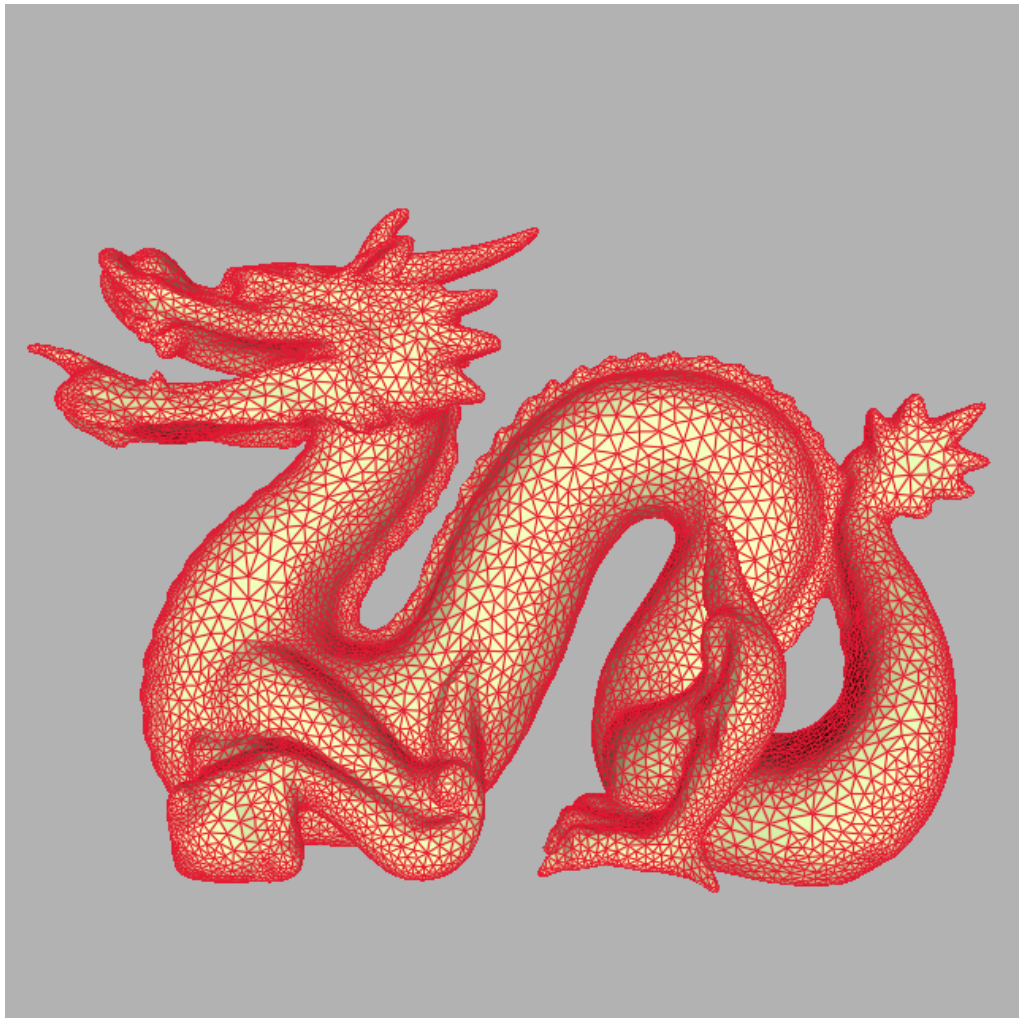
Offsetting



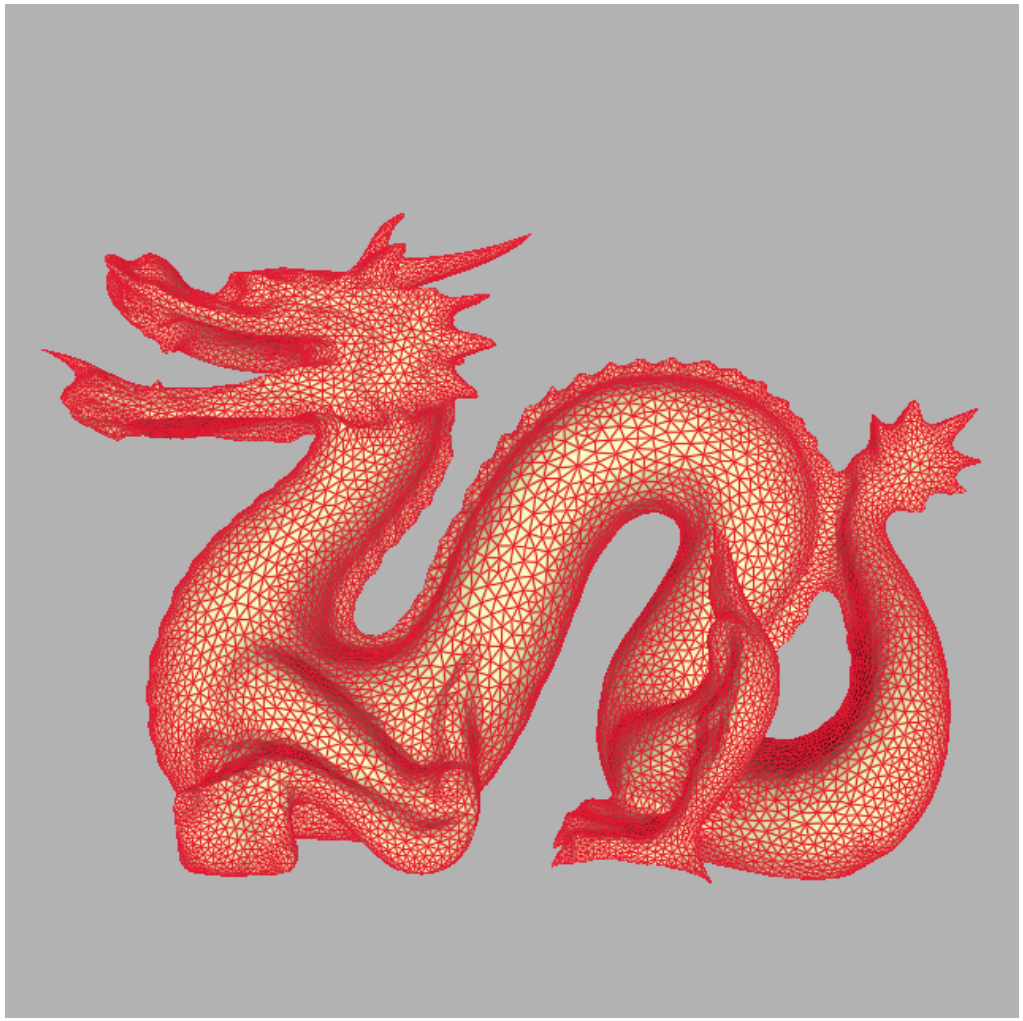
Offsetting



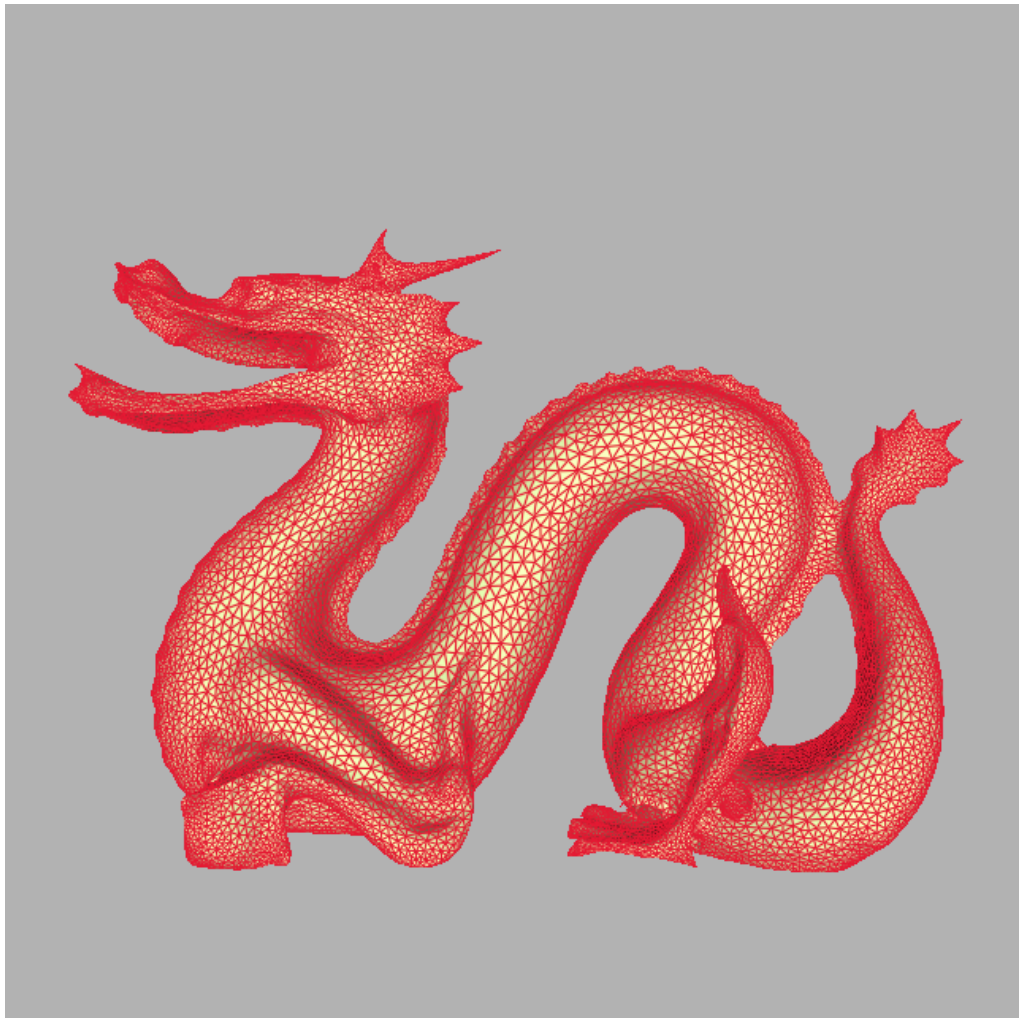
Offsetting



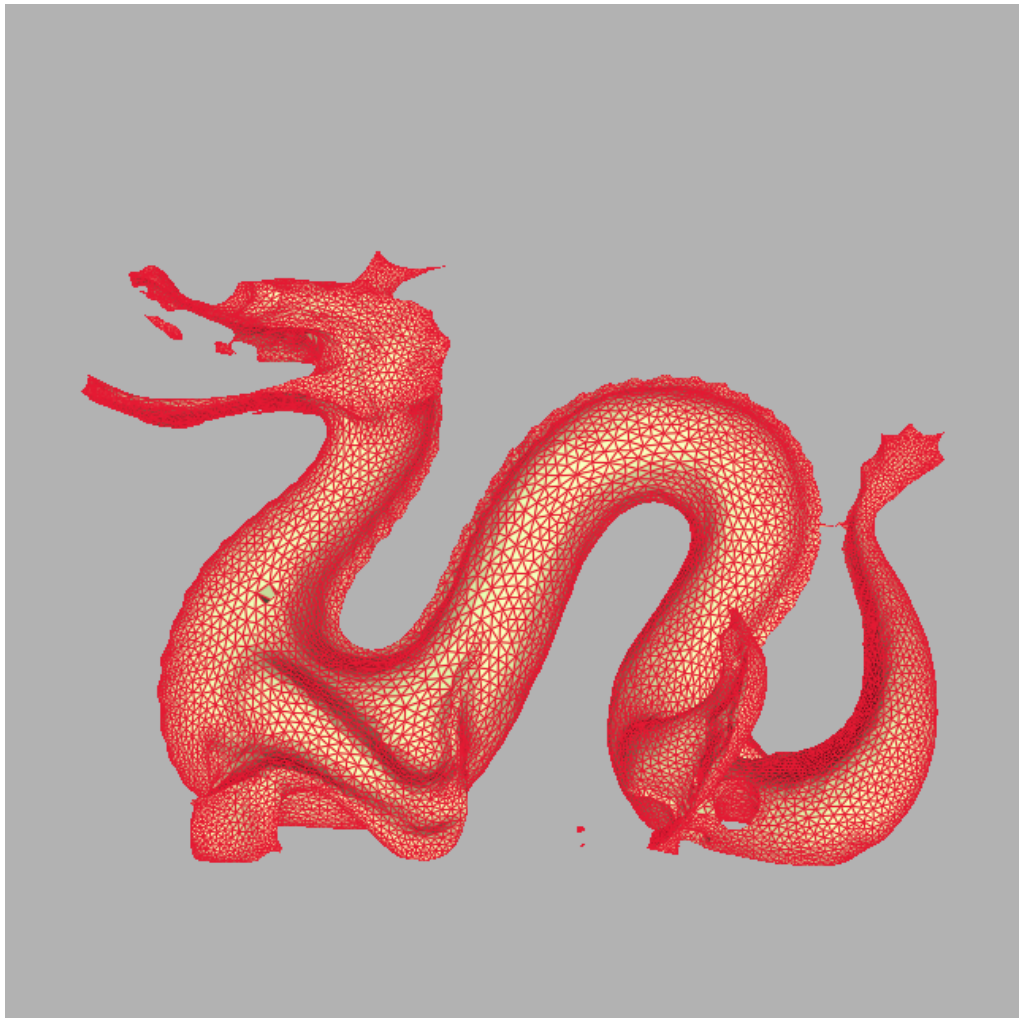
Offsetting



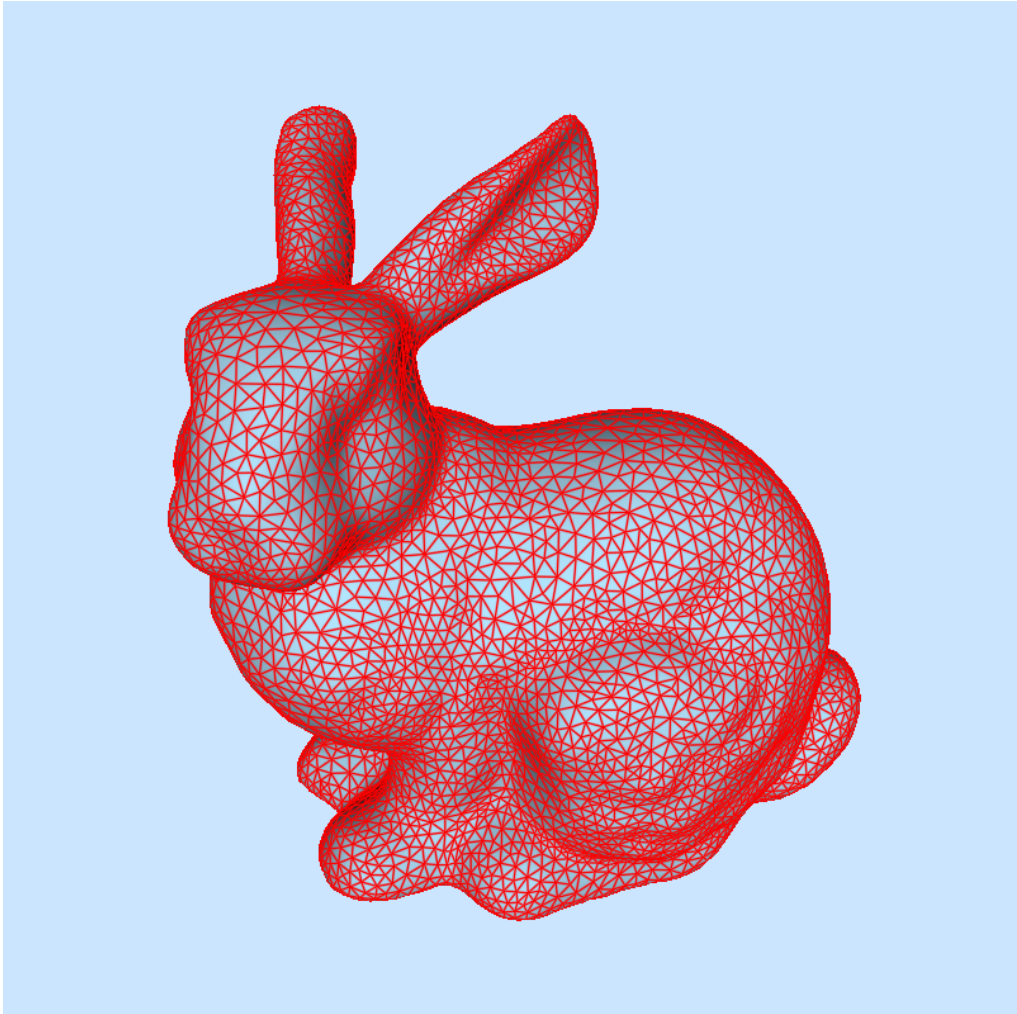
Offsetting



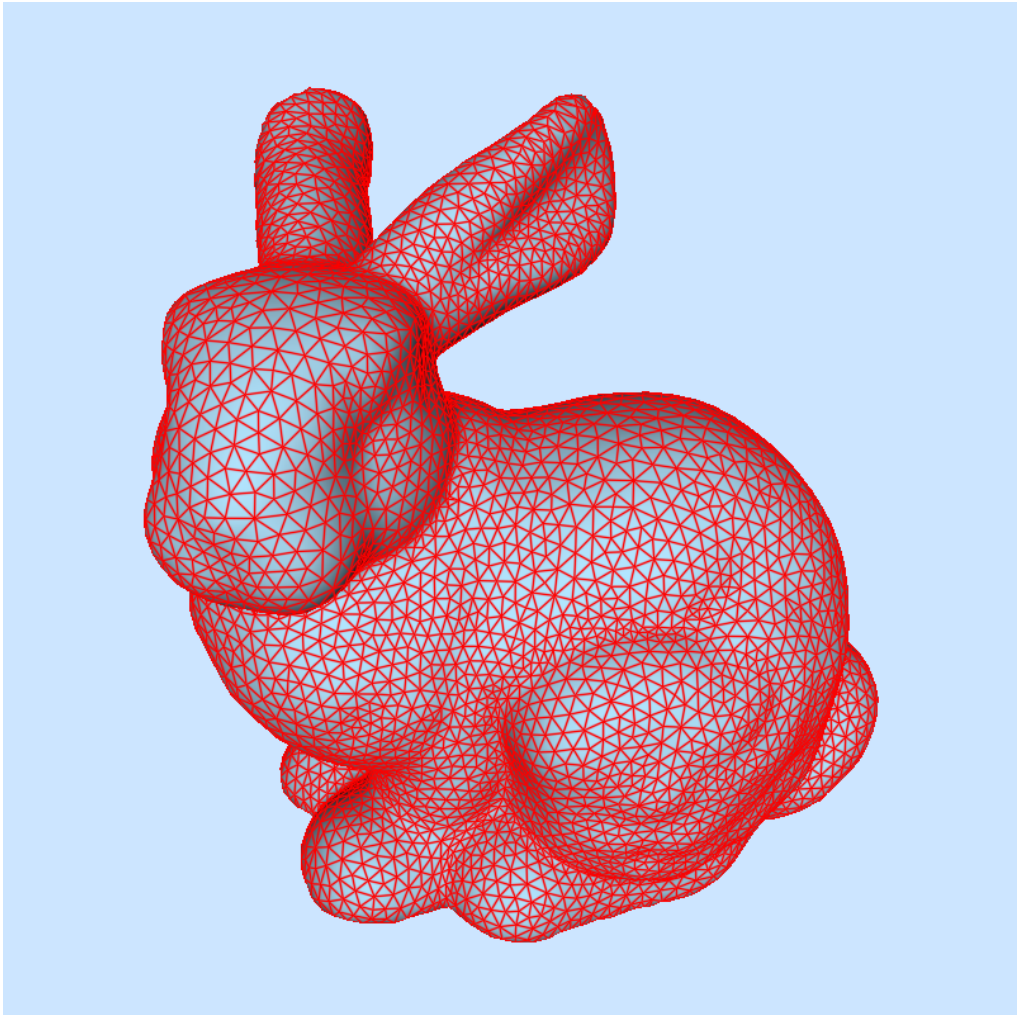
Offsetting



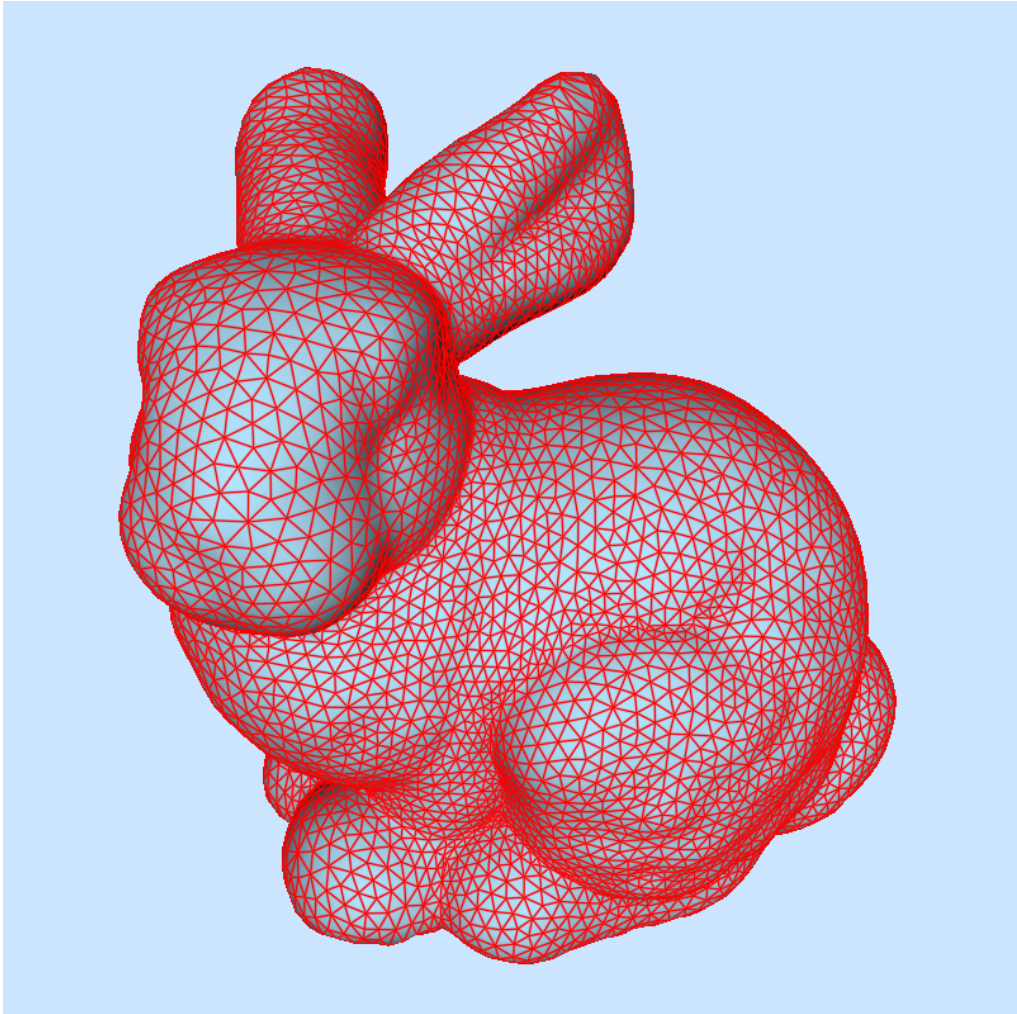
Offsetting



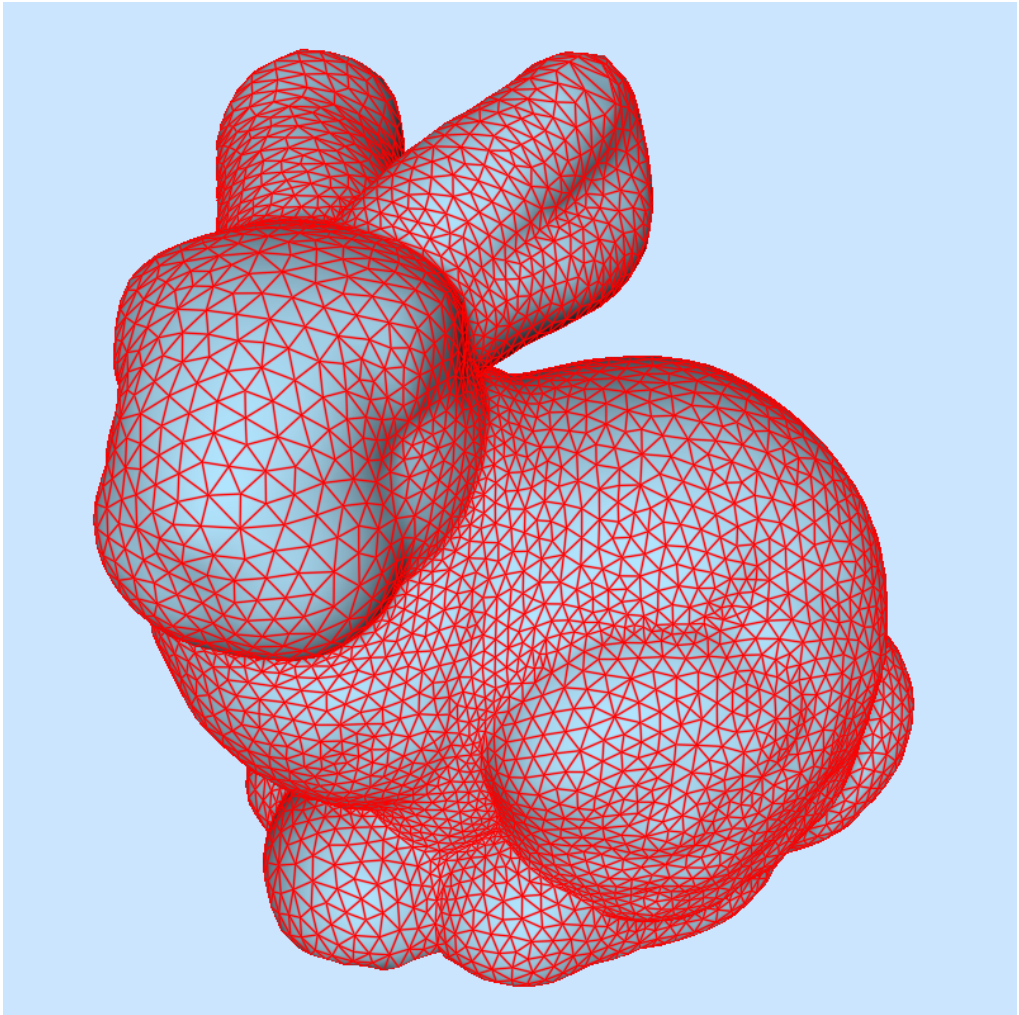
Offsetting



Offsetting



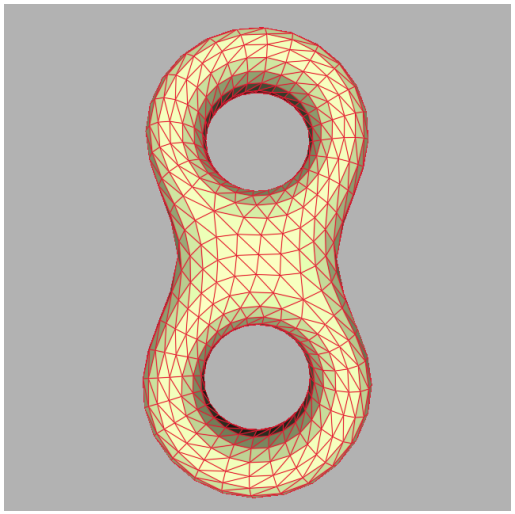
Offsetting

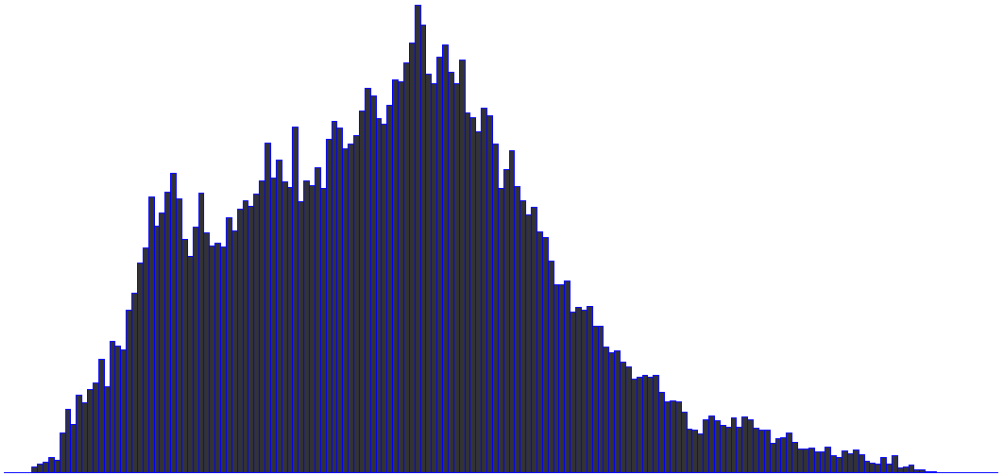


Performance

example	GENUS2	2SPHERES	BUNNY	DRAGON
#surf. vertices	766	1680	6929	27549
#surf. triangles	1536	3348	13934	55098
#total tets	10309	16433	77631	372015
#inside tets	2703	5554	35362	160110
time per iteration	1.2 s	1.4 s	18.2 s	48.0 s

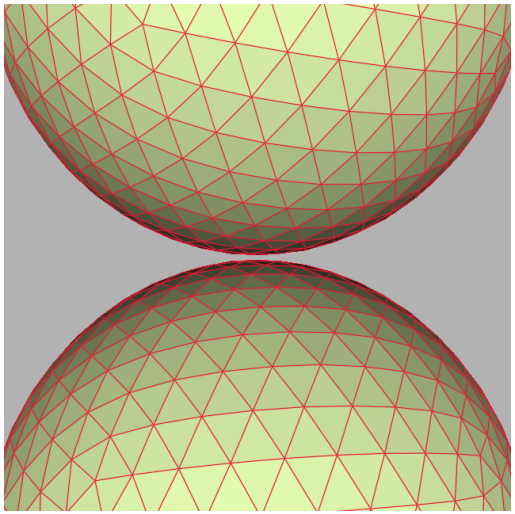
Dihedral angles

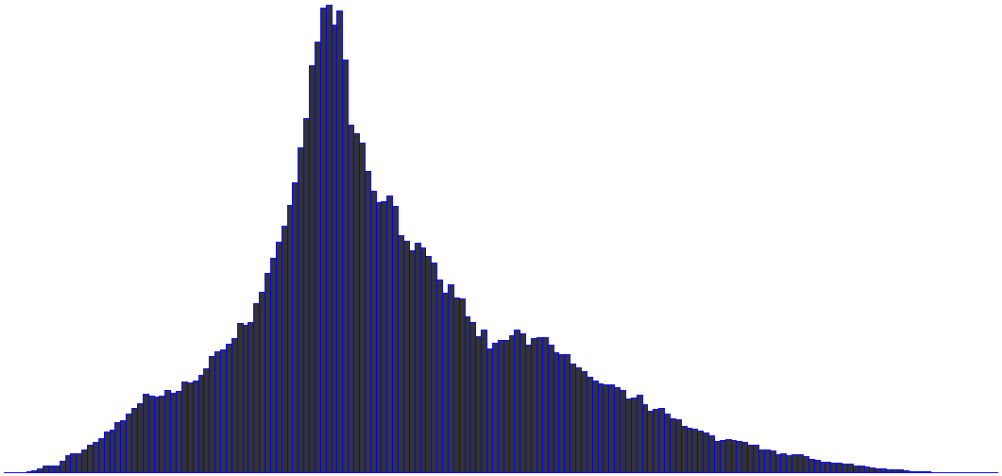




Although low quality tetrahedra occasionally appear in the DSC mesh, dihedral angles from outside 6°–171° range constitute less than 0.1% of all dihedral angles in the tetrahedral mesh throughout all iterations.

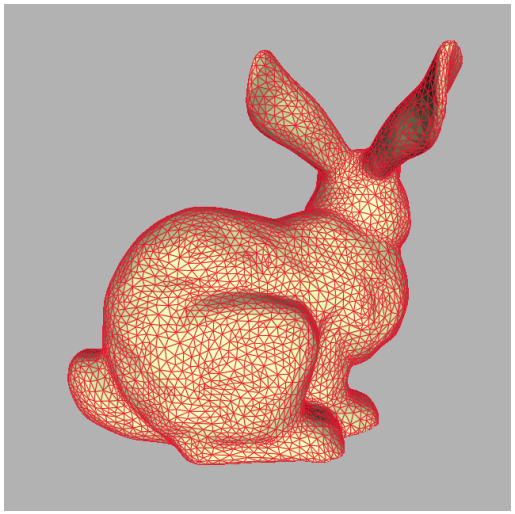
Dihedral angles

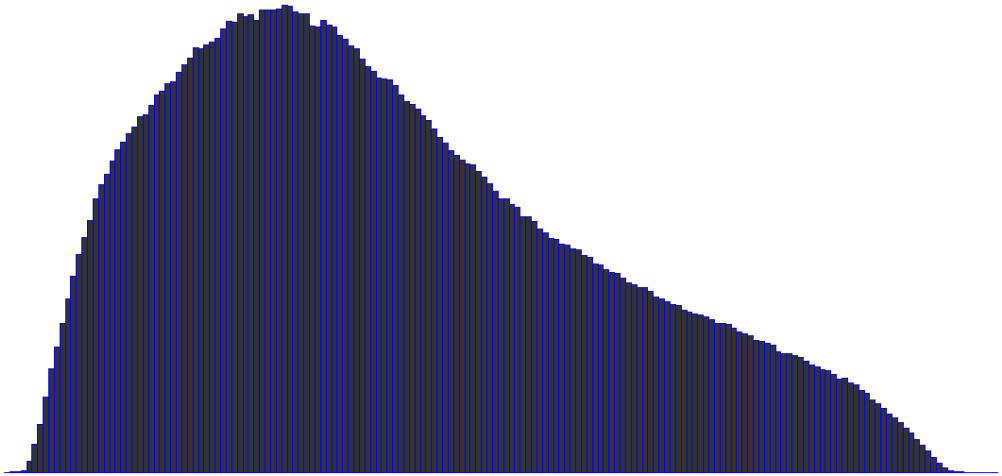




Although low quality tetrahedra occasionally appear in the DSC mesh, dihedral angles from outside 6°–171° range constitute less than 0.1% of all dihedral angles in the tetrahedral mesh throughout all iterations.

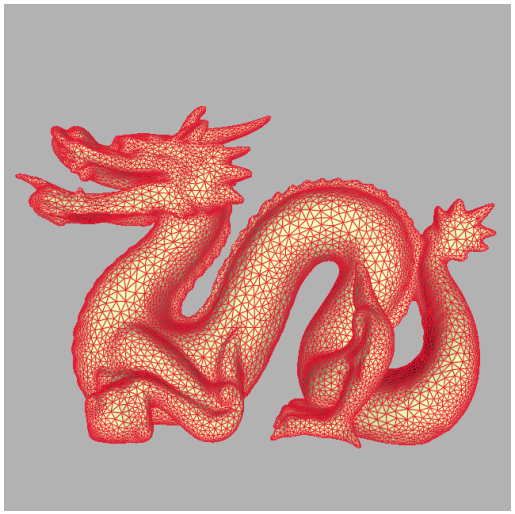
Dihedral angles

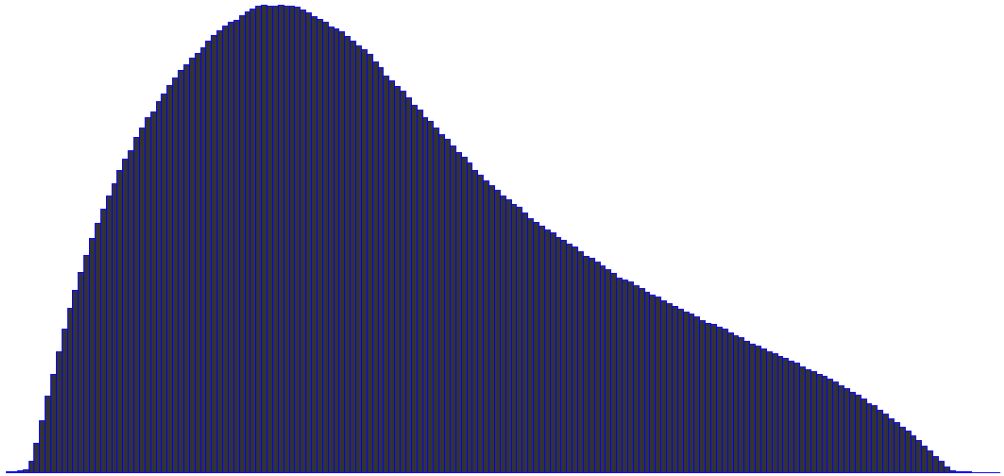




Although low quality tetrahedra occasionally appear in the DSC mesh, dihedral angles from outside 6°–171° range constitute less than 0.1% of all dihedral angles in the tetrahedral mesh throughout all iterations.

Dihedral angles





Although low quality tetrahedra occasionally appear in the DSC mesh, dihedral angles from outside 6°–171° range constitute less than 0.1% of all dihedral angles in the tetrahedral mesh throughout all iterations.

Conclusions

- The main drawbacks of the DSC method are:

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;
- However, it also has significant advantages:

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;
- However, it also has significant advantages:
  - numerical diffusion due to the interface tracking method alone is residual and controllable;

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;
- However, it also has significant advantages:
  - numerical diffusion due to the interface tracking method alone is residual and controllable;
  - it provides explicit representation of the interface, which does not change gratuitously between steps;

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;
- However, it also has significant advantages:
  - numerical diffusion due to the interface tracking method alone is residual and controllable;
  - it provides explicit representation of the interface, which does not change gratuitously between steps;
  - it naturally supports multiple phases;

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;
- However, it also has significant advantages:
  - numerical diffusion due to the interface tracking method alone is residual and controllable;
  - it provides explicit representation of the interface, which does not change gratuitously between steps;
  - it naturally supports multiple phases;
  - intrinsic collision detection mechanism offers a possibility for topology control;

Conclusions

- The main drawbacks of the DSC method are:
  - its complexity in terms of mesh operations used;
  - performance, inferior to that of the level set method;
- However, it also has significant advantages:
  - numerical diffusion due to the interface tracking method alone is residual and controllable;
  - it provides explicit representation of the interface, which does not change gratuitously between steps;
  - it naturally supports multiple phases;
  - intrinsic collision detection mechanism offers a possibility for topology control;
  - if offers a unified computational and interface tracking framework for FEM simulations;