Topological Concepts in Physics Simulations

CSCI 535 - Jacob Hofer

Big Question

How do we simulate physical objects that can undergo topological changes?

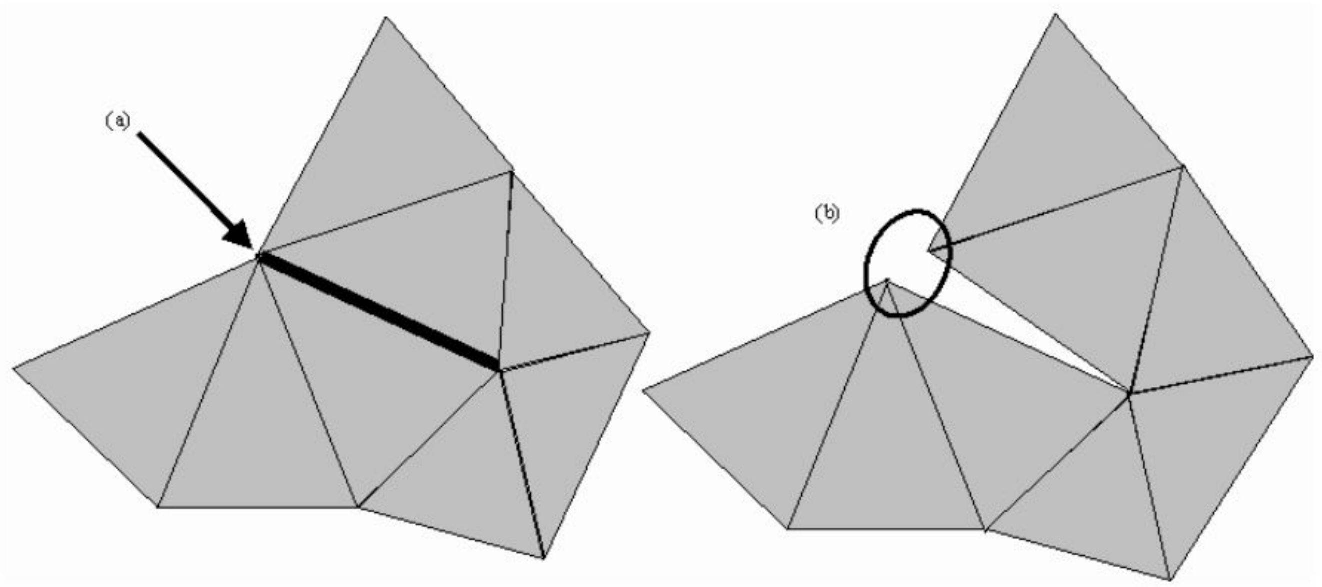
Handling Cuts

Allowing for meshes to be cut and torn apart allow us to simulate various physical interactions, such as:

- Surgery
- Car Crashes
- Object Tearing

Handling Cuts

Given a triangulation, how can we handle cutting at a vertex?

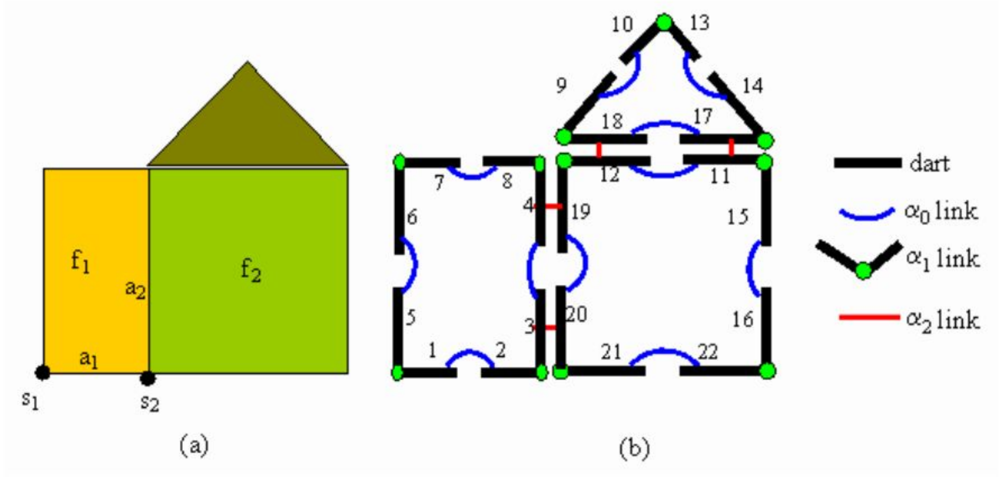


Handling Cuts

Introduce the Following:

- Darts
- Generalized Maps
- Orbits
- i-cells

These models work for any object structure, but when considering physics, we often look at simplicial complexes, or just triangulations.



Handling Cuts

Our framework provides a good representation of the structure, but no physical data is included yet.

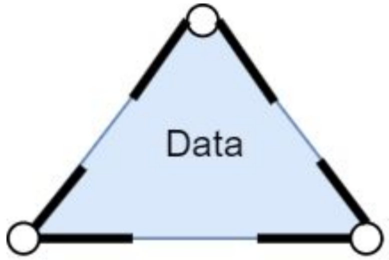
Multiple options for embedding:

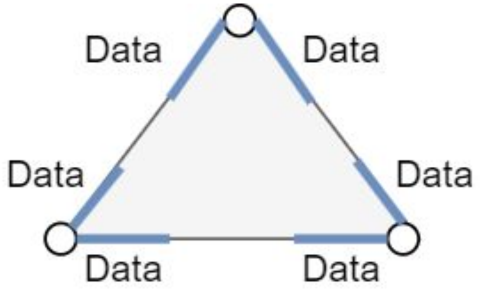
- Attach to i-cells
- Attach to all darts of an orbit
- Attach to a single dart of an orbit

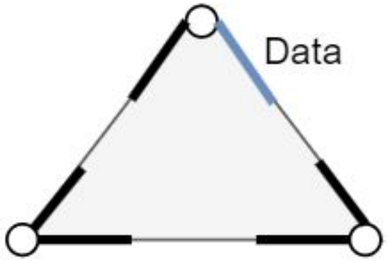
Typical approach is attaching to a single dart.

Examples of embedding data include:

- Mass associated to a vertex orbit
- Spring stiffness associated to an edge orbit



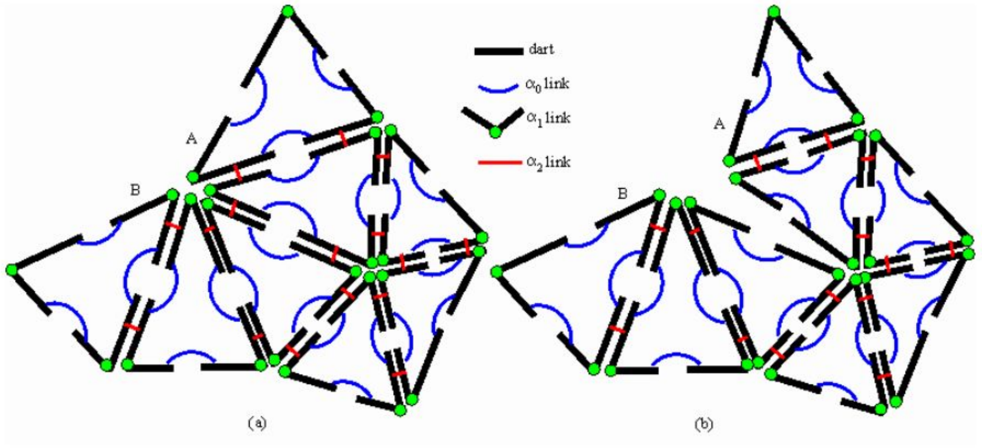




Handling Cuts

To cut our mesh, introduce face dissociation:

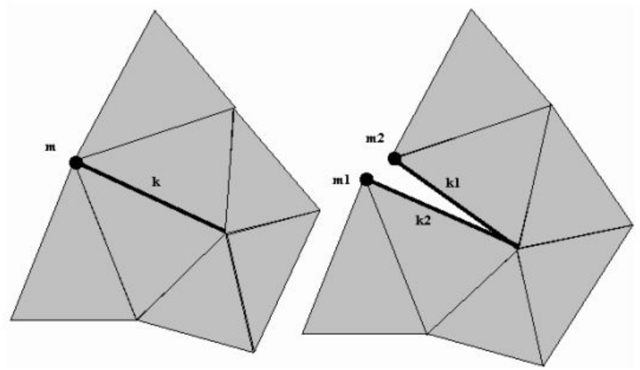
- Obtain a dart for each vertex and edge of face
- Obtain a dart for each vertex and edge of adjacent faces using α_2 links
- Eliminate the α_2 links
- Check for vertex dissociation by checking vertex orbits
- Check for edge dissociation by checking edge orbits



Handling Cuts

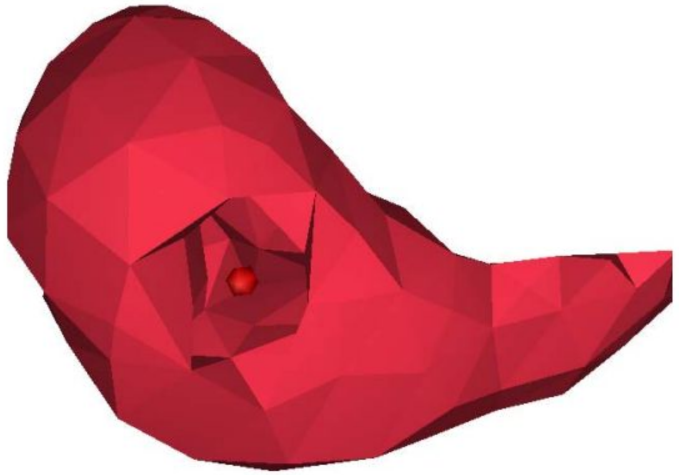
Split embeddings in the presence of vertex and edge dissociation. For example

- Splitting mass between the two new vertex orbits
- Splitting spring stiffness between the two new edge orbits



Removing volumes is simple:

- Isolate volume via face dissociation
- Remove darts and embeddings of the volume



Handling Deformations

Meshes also need to be able to deform, in both plastic and elastic fashions.

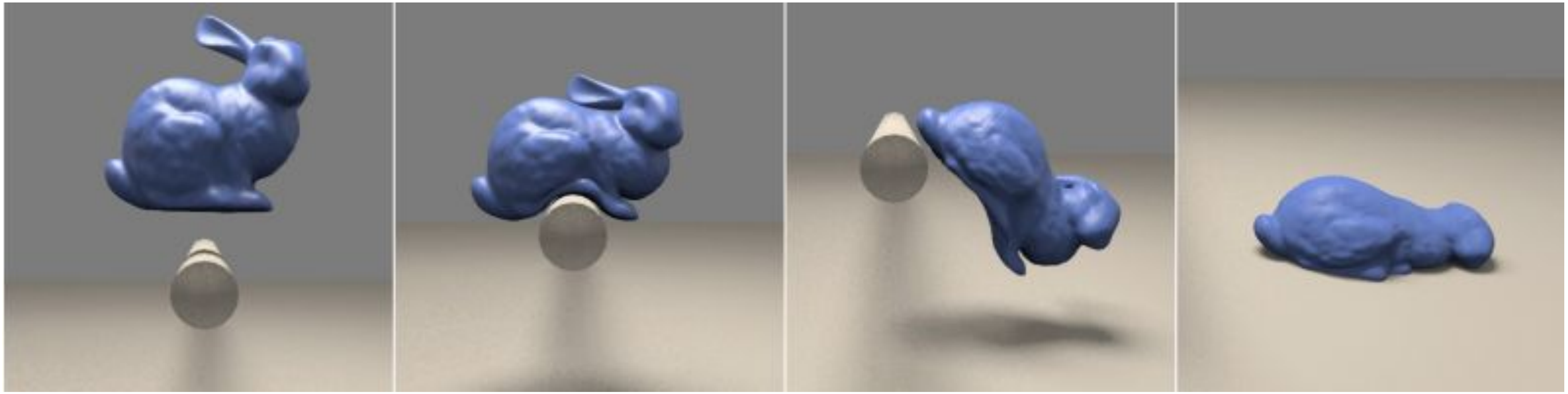
Plastic deformation occurs when a deformed object does not return to its original state.

Elastic deformation occurs when a deformed object returns to its original state.

Often we experience both types of deformation, with levels varying depending on the type of material.

Handling Deformation

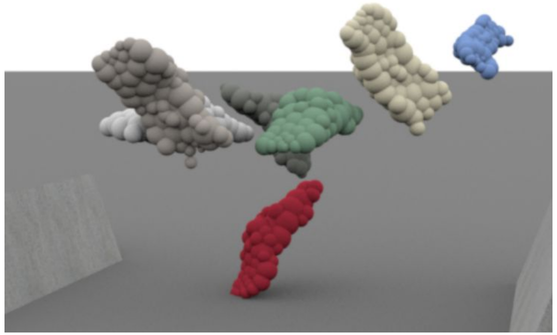
How can we handle deforming a body?



Handling Deformation

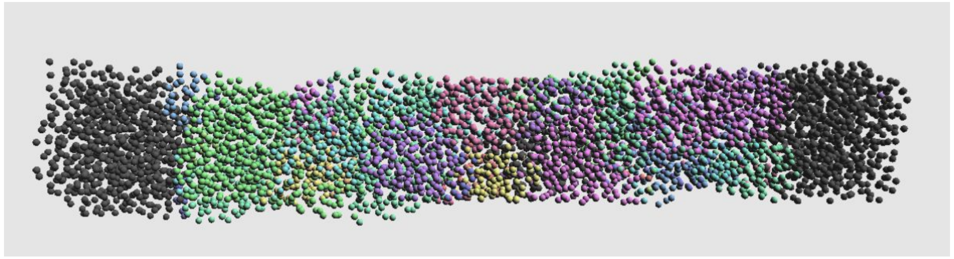
Example-Based Deformations

- Precompute series of mesh deformations
- Represent mesh using overlapping spheres
- Use spheres to compute collisions
- Blend between deformations to move spheres
- Good for games or animation
- Bad for accurate simulations



Point-Based Deformations

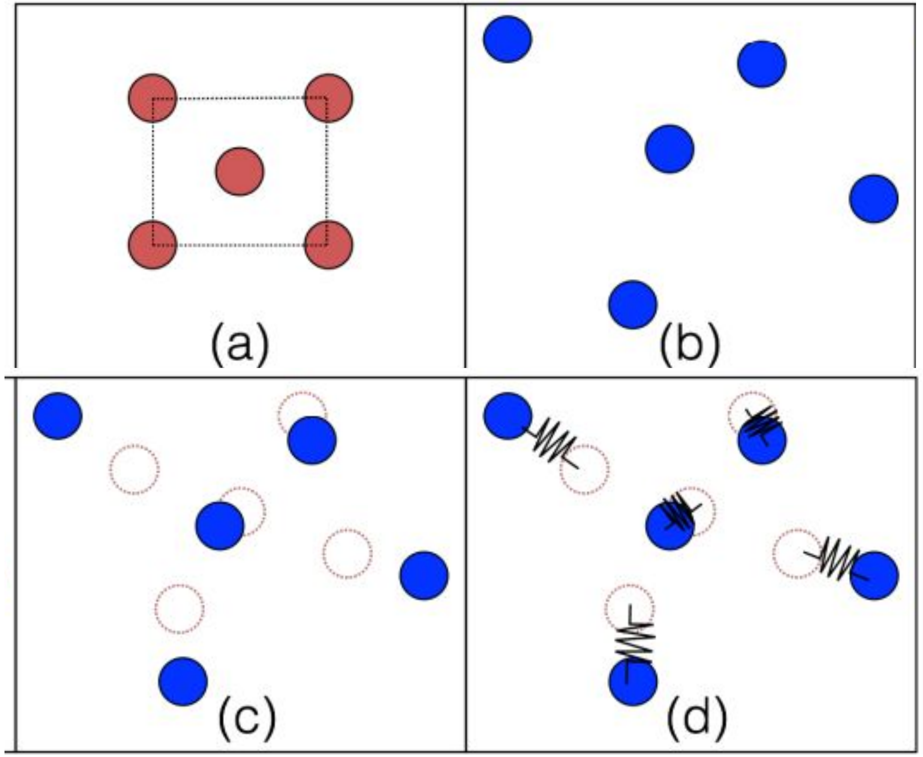
- Represent objects using point clouds
- Original point cloud denotes the rest position
- Physics is applied directly on particles
- Use rest positions to compute plastic and elastic deformations
- Much better for accurate simulations



Handling Deformation

To deform our point cloud, we can employ clustered shape matching:

- Split the object into overlapping clusters that provide rest positions
- Apply rigid physics on the clusters to obtain goal positions
- Use hookean springs to move particles to their goal positions



Handling Deformation

Formally, shape matching seeks a solution to $\boldsymbol{R}, \bar{\boldsymbol{x}}$-$\bar{\boldsymbol{r}}$, minimizing:

$$\sum_i m_i \|\mathbf{R}(\mathbf{r}_i - \bar{\mathbf{r}}) - (\mathbf{x}_i - \bar{\mathbf{x}})\|^2$$

Where m_i, $\boldsymbol{r}_i$, and $\boldsymbol{x}_i$ are the mass, rest position, and world position of particle i.

We then use these values to compute the goal positions for each particle:

$$\mathbf{g}_i = \mathbf{R}(\mathbf{r}_i - \bar{\mathbf{r}}) + \bar{\mathbf{x}}$$

When considering clusters, the goal for a particle is based on the weighted sum over all its clusters:

$$\mathbf{g}_i = \sum_c w_{ic}\, \mathbf{g}_{ic}$$

Finally, reclustering takes place, only ever adding one at a time.

Handling Deformation

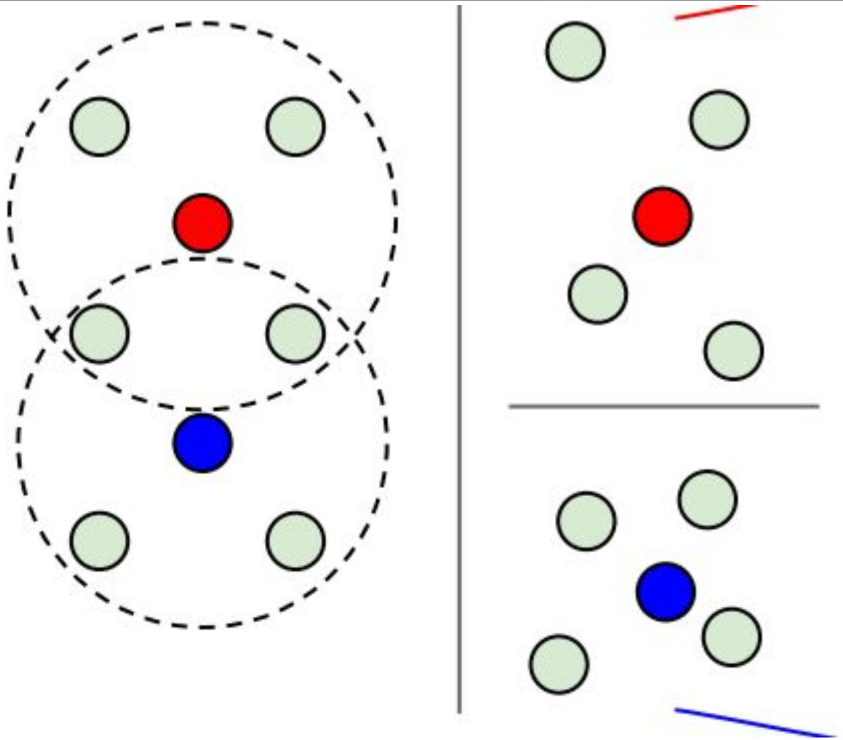
Alternatively, a particle's rest position can be based on its neighbor's world positions.

This causes the rest space to be higher than 3 dimensions, as it contains dimensions for each neighbor.

Example:

The two particles in the center have rest spaces from both the red and blue particles.

Use Moving Least Squares (MLS) to project down into 3 dimensions.



Handling Deformation

Moving least squares seeks to find the deformation gradient $\mathbf{F}_i$ for the ith particle to minimize:

$$\sum_j ||w_{ij}(\mathbf{F}_i \mathbf{u}_{ij} - \mathbf{x}_{ij})||^2$$

where $\boldsymbol{u}_{ij}$ and $\boldsymbol{x}_{ij}$ are the vectors between particles i and j in rest space and world space, and w_{ij} is a weight.

This $\mathbf{F}_i$ is obtained by solving the linear equation:

$$\mathbf{F}_i(\sum_j w_{ij}\mathbf{u}_{ij}\mathbf{u}_{ij}^T) = \sum_j w_{ij}\mathbf{x}_{ij}\mathbf{u}_{ij}^T$$

Finally, $\mathbf{F}_i$ is used to map the particle's position from rest space to world space.

This rest space can then be used as in shape matching.

Handling Deformation

Allows for realistic and stable deformations:

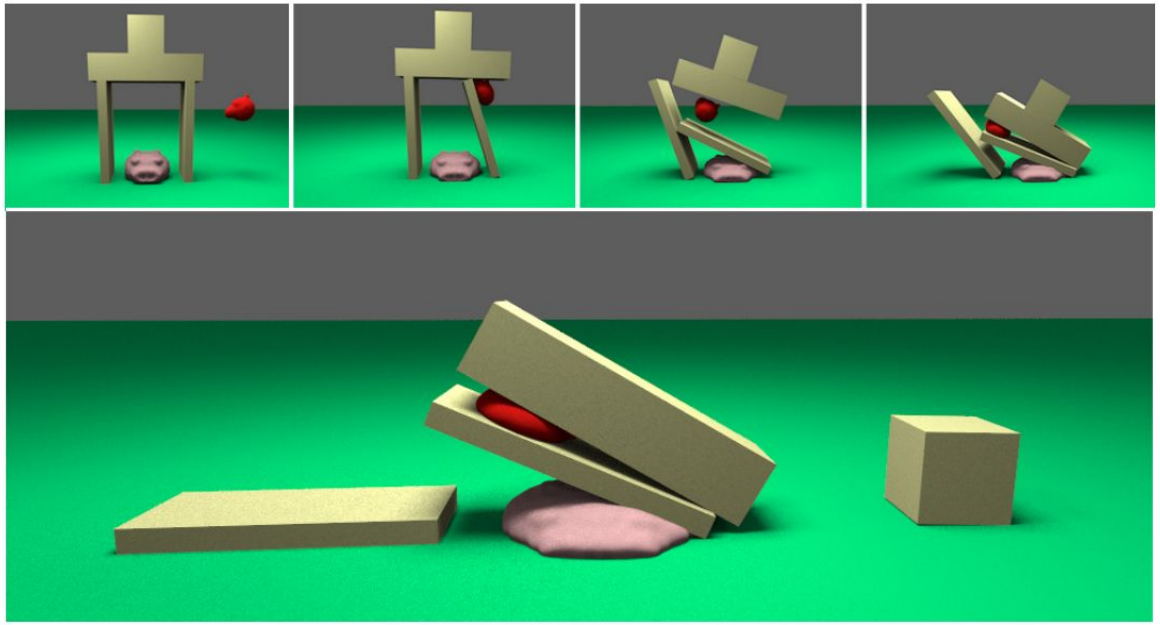


Fig. 10: *An "upset fowl" destroys a pig's house.*

Detecting Collision

We now have a framework for determining how to handle a topological change, but how does one occur?

From collisions of course!

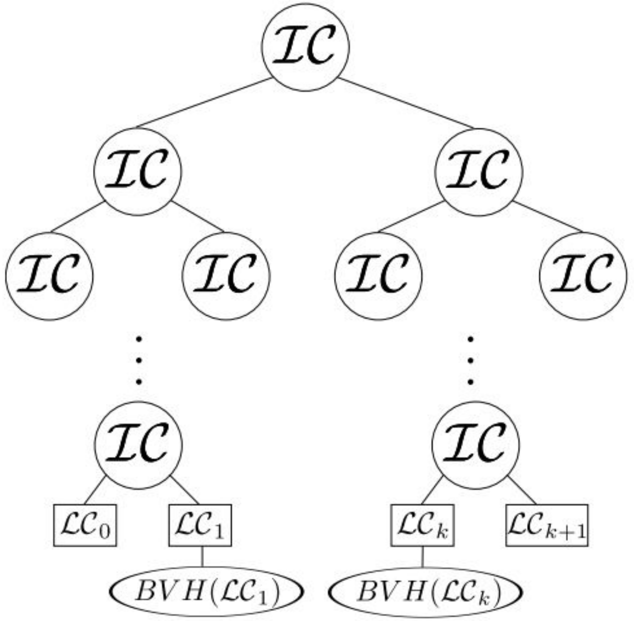
We now introduce a way to quickly detect a collision between objects that remains consistent in the presence of topological changes.

Detecting Collision

Determining if and where two meshes collide can be a difficult task.

An initial thought is to consider every pair of triangles, giving $O(n^2)$.

Instead introduce the idea of bounding volume hierarchies (BVHs).



Detecting Collision

Create a cluster tree in a bottom-up fashion.

Leaf clusters consist of sets of triangles that have no overlap.

Compute leaf clusters with respect to an observation point O, using a characteristic function:

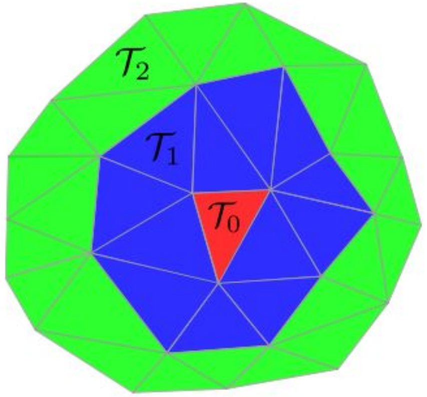
$$\phi(\tau, O, t) = n(t) \cdot (\nu_0(t) - O(t))$$

where τ is a triangle with normal n and vertices v_i.

Triangles do not overlap if the following is true $\forall \tau$:

$$\phi(\tau, O, t) \geq 0$$

Compute leaf clusters from concentric rings of triangles, with O being the centroid of the first triangle.



$$\mathcal{LC} = \mathcal{T}_0 \cup \mathcal{T}_1 \cup \ldots$$

Detecting Collision

Then merge clusters into intermediate clusters.

Compute observation point O_i as an average over all vertices in cluster i.

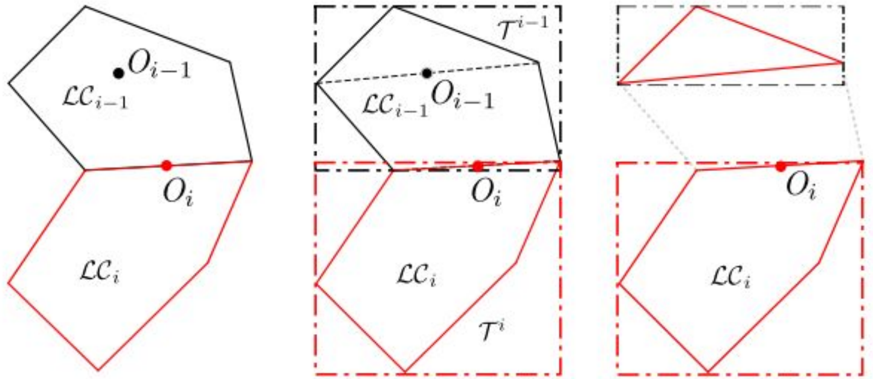
Ensure the triangles of each cluster's final ring satisfy:

$$\phi(\tau, O_i, t) \geq 0,\ \tau \in \mathcal{T}^{i-1} \cup \mathcal{T}^i$$

Check if the bounding volume of $\mathcal{LC}_i$ overlaps the bounding volume of $\mathcal{LC}_{i-1}$- $\mathscr{T}^{i-1}$.

If no overlap, merge.

This creates a binary tree of clusters we can use to generate a BVH.



Obtaining Meshes

Sometimes, such as in the case of simulating vehicles, we have easy access to the mesh for our simulation.

However, sometimes our data does not come from a mesh, but from machines reading values.

So, how do we obtain a mesh?

Obtaining Meshes

One popular algorithm is marching cubes.

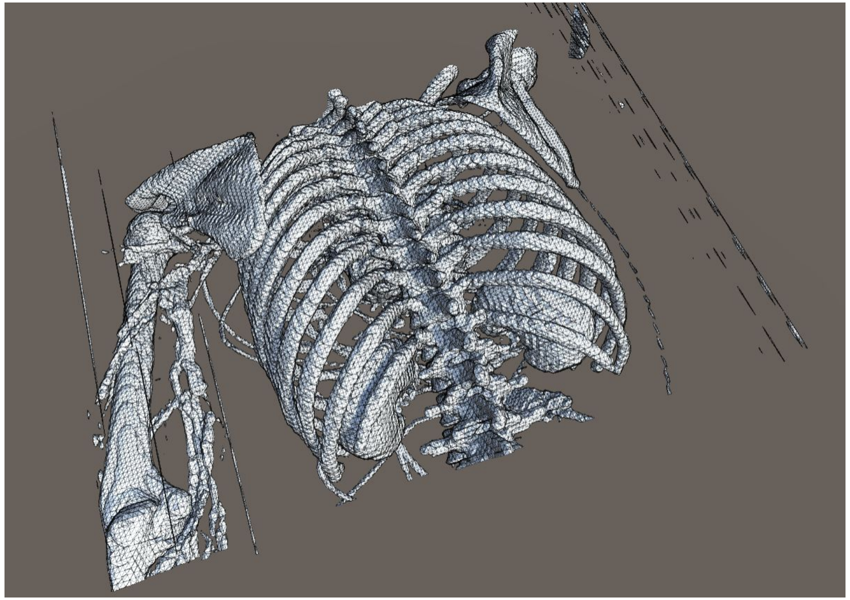
Takes a continuous function defined over 3D space:

$$f : \mathbb{R}^3 \to \mathbb{R}$$

Computes a mesh representing the surface of the level set:

$$f^{-1}(v), v \in \mathbb{R}$$

Especially useful for data from CT Scans, MRI, etc.



Marching Cubes Demo

Available at https://github.com/CodeAX2/CTData/blob/main/CTData.ipynb.

Citations

Falkenstein, Michael, et al. ‘Reclustering for Large Plasticity in Clustered Shape Matching’. Proceedings of the Tenth International Conference on Motion in Games, ACM, 2017, p. 5:1-5:6, https://doi.org/10.1145/3136457.3136473. MIG ’17.

He, Liang, et al. ‘Interactive Continuous Collision Detection for Topology Changing Models Using Dynamic Clustering’. Proceedings of the 19th Symposium on Interactive 3D Graphics and Games, Association for Computing Machinery, 2015, pp. 47–54, https://doi.org/10.1145/2699276.2699286. i3D ’15.

Jones, Ben, et al. ‘Efficient Collision Detection for Example-Based Deformable Bodies’. Proceedings of the Tenth International Conference on Motion in Games, ACM, 2017, p. 4:1-4:5, https://doi.org/10.1145/3136457.3136469. MIG ’17.

Jones, Ben, et al. ‘Deformation Embedding for Point-Based Elastoplastic Simulation’. ACM Trans. Graph., vol. 33, no. 2, Association for Computing Machinery, Apr. 2014, https://doi.org10.1145/2560795.

Meseure, Philippe, et al. Topology-Based Physical Simulation. 11 2010, https://doi.org/10.2312/PE/vriphys/vriphys10/001-010.