Parallel Implementation of Interactive Soft Body Dynamics

Laurin Stecher

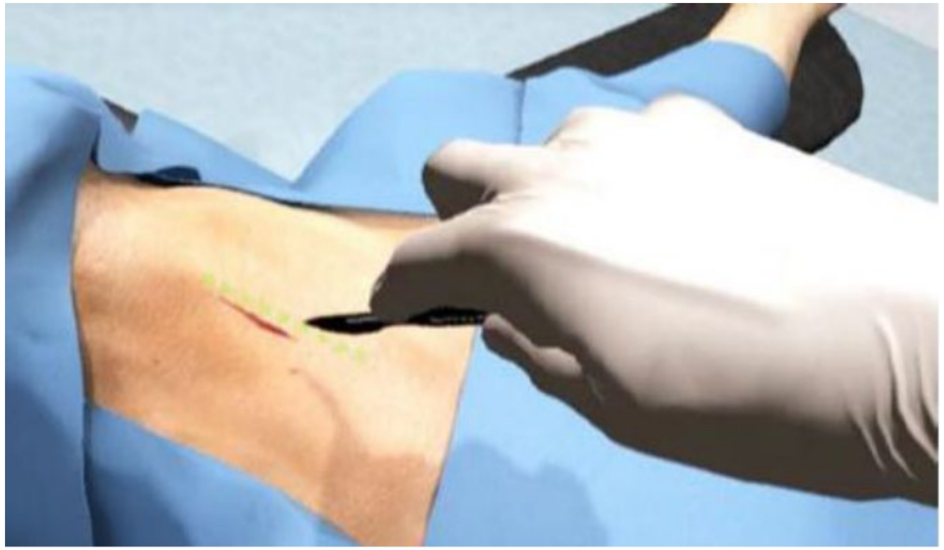
Supervisors: Nikolaus Rauch, Marcel Ritter

Interactive Graphics and Simulation Group

Institute of Computer Science

Motivation

- **Soft body simulations** for surgical training
- Rising demand for **realistic** and **interactive** simulations
- Interactive in **real-time**

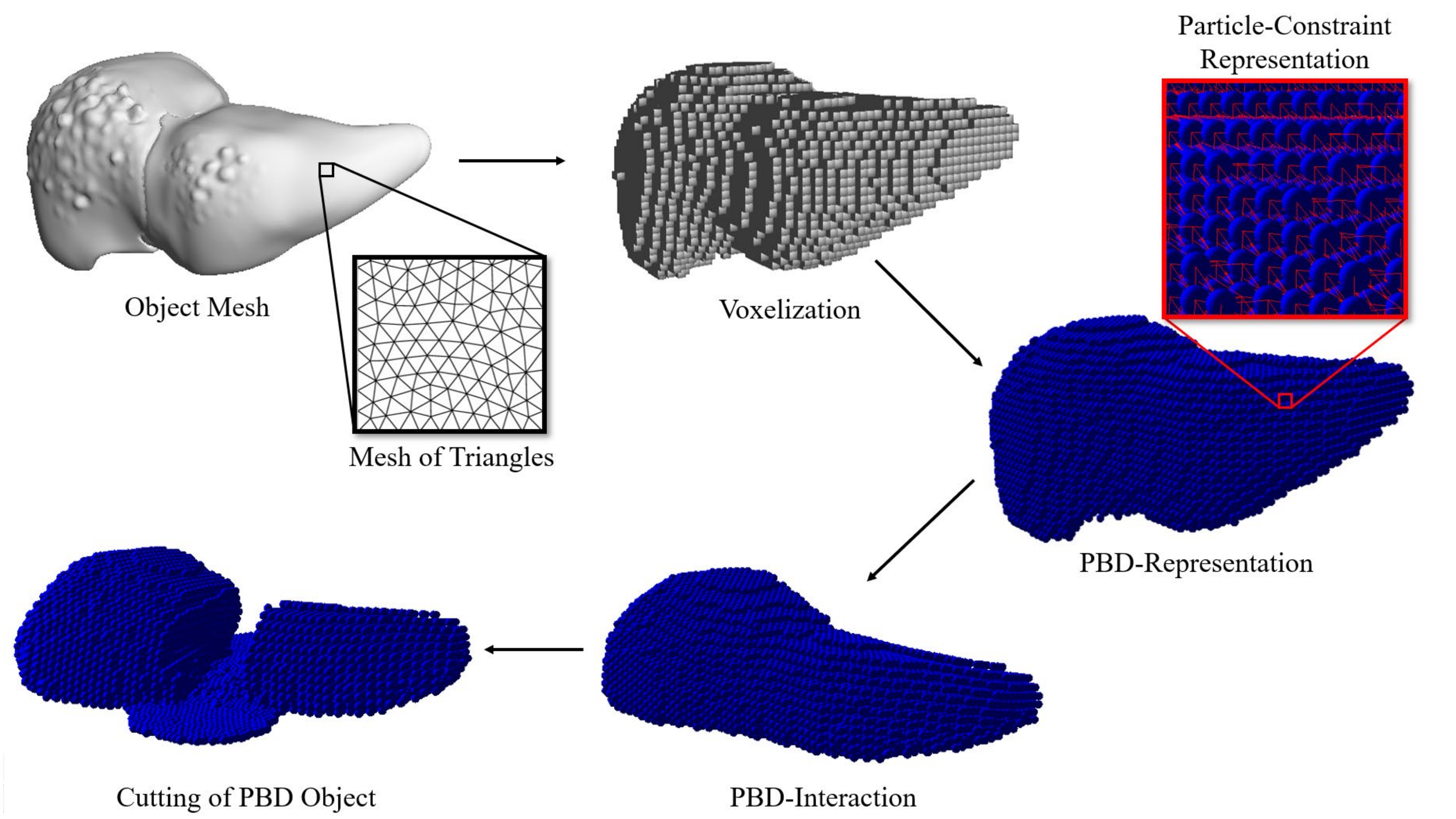


Surgical simulation, from Surega VR

Main Objectives

1) Creation of an **interactive real-time soft body simulation**
   - Position Based Dynamics was parallelized
   - Includes a cutting tool for user interaction

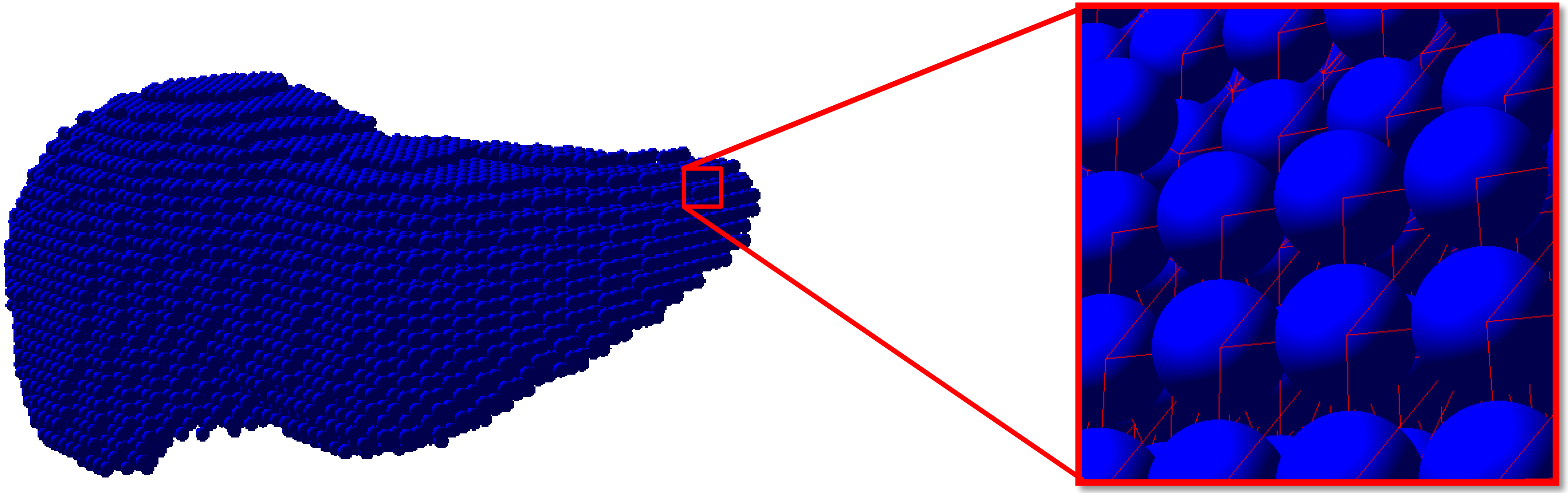
2) Visualization and **modeling of organic tissue**
   - Enable the usage of arbitrary triangle meshes within the simulation



Particle-Constraint
Representation
Object Mesh
Mesh of Triangles
Voxelization
PBD-Representation
Cutting of PBD Object
PBD-Interaction

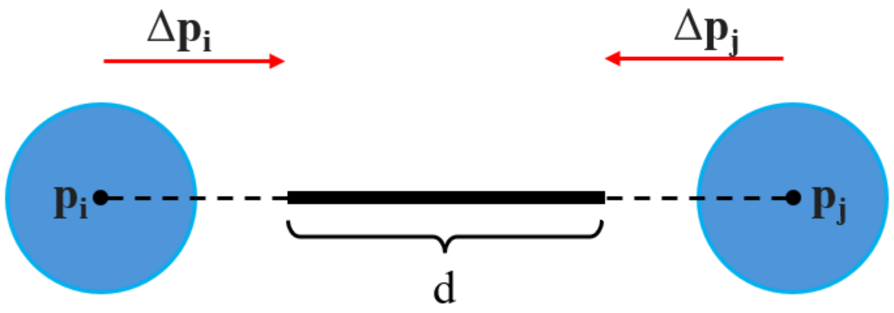
PBD – Position Based Dynamics

- Based on **particles** and **constraints**
- Uses a **Gauss-Seidel linear equation solver** for state updates



Constraints

- Different forms of constraints
  - Fixed, Distance, Collision, …
- Act on particle positions
- Categorized into permanent and temporary
  - Permanent are always active
  - Temporary are reset every loop iteration



Example of a distance constraint

PBD – Position Based Dynamics

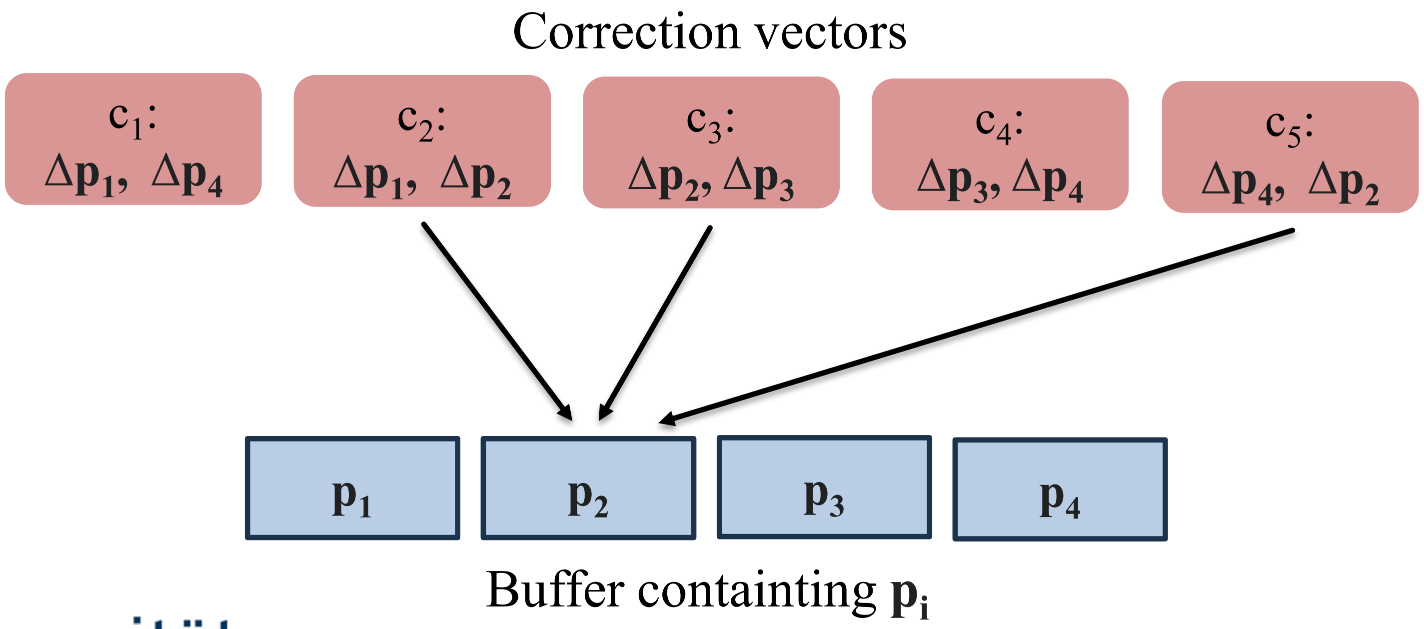
```
loop
    for all particles i do
        v_i ← v_i + Δt w_i f_ext,i(x_i)
        p_i ← x_i + Δt v_i
    end for
    for all particles i do
        genCollisionConstraints(x_i → p_i)
    end for
    loop solver_iteration times
        projectConstraints(C_1, ..., C_{M+M_Coll}, p_1, ..., p_N)
    end loop
    for all particles i do
        v_i ← (p_i − x_i)/Δt
        x_i ← p_i
    end for
    velocityUpdate(v_1, ..., v_N)
end loop
```

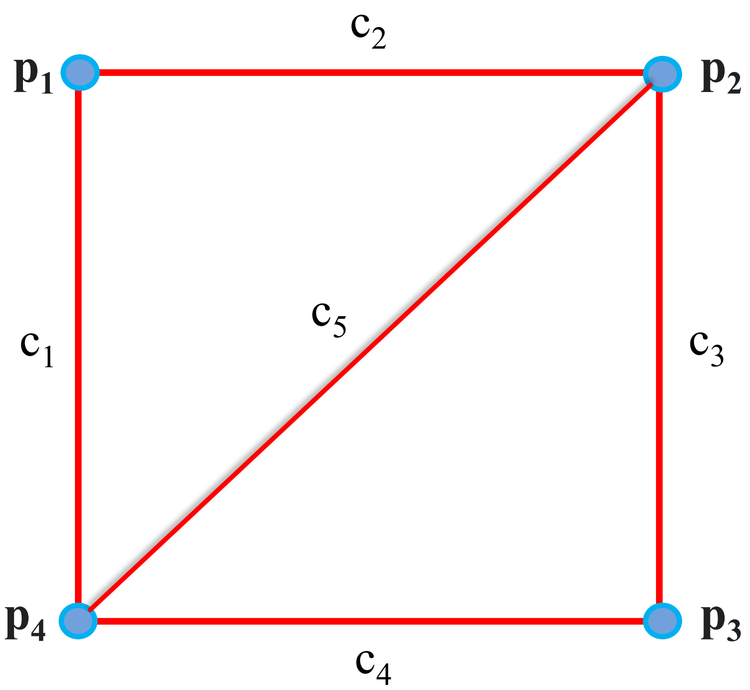
PBD algorithm [1]

- **Predict positions** using forces and velocity
- **Handle collisions**
- **Solve** all **constraints** (correction vectors)
- **Update particle state**

Challenges for Parallelism

- **Particles**
  - Each thread writes to **different** array entry
- **Constraints**
  - Multiple threads write to **the same** array entry





GPU-Implementation

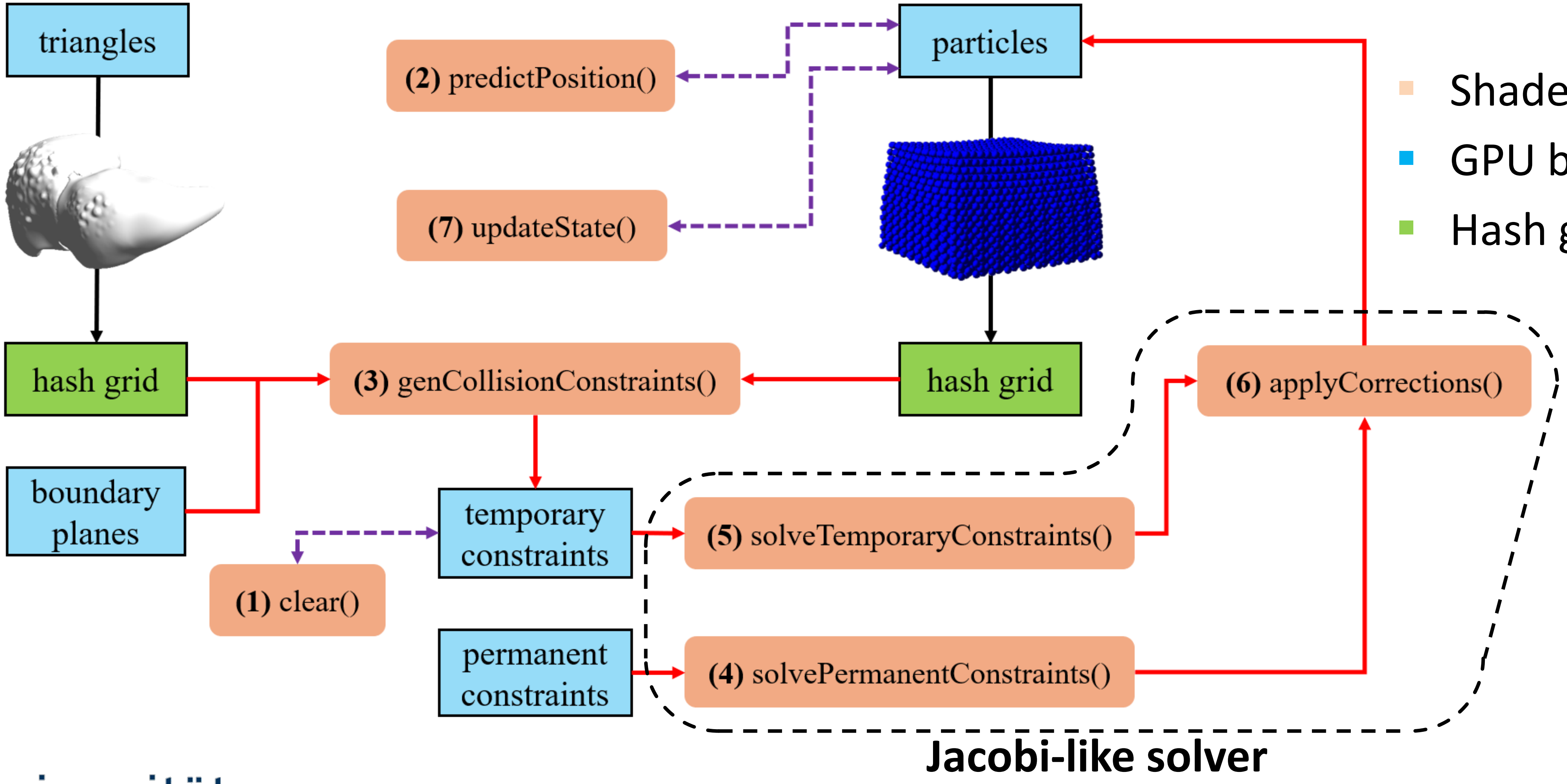
- Implemented using **C++** with **OpenGL**

- The algorithm is computed in parallel on the GPU using **Compute Shaders**

- Introduces a **weighted Jacobi-like solver** to avoid race conditions when solving constraints in parallel
  - Writes corrections to a buffer for every particle, applies them in a weighted form

- Used **GPU buffers** to store needed data (particles, constraints, …)
  - No data transfer between GPU and CPU

GPU-Implementation – Workflow

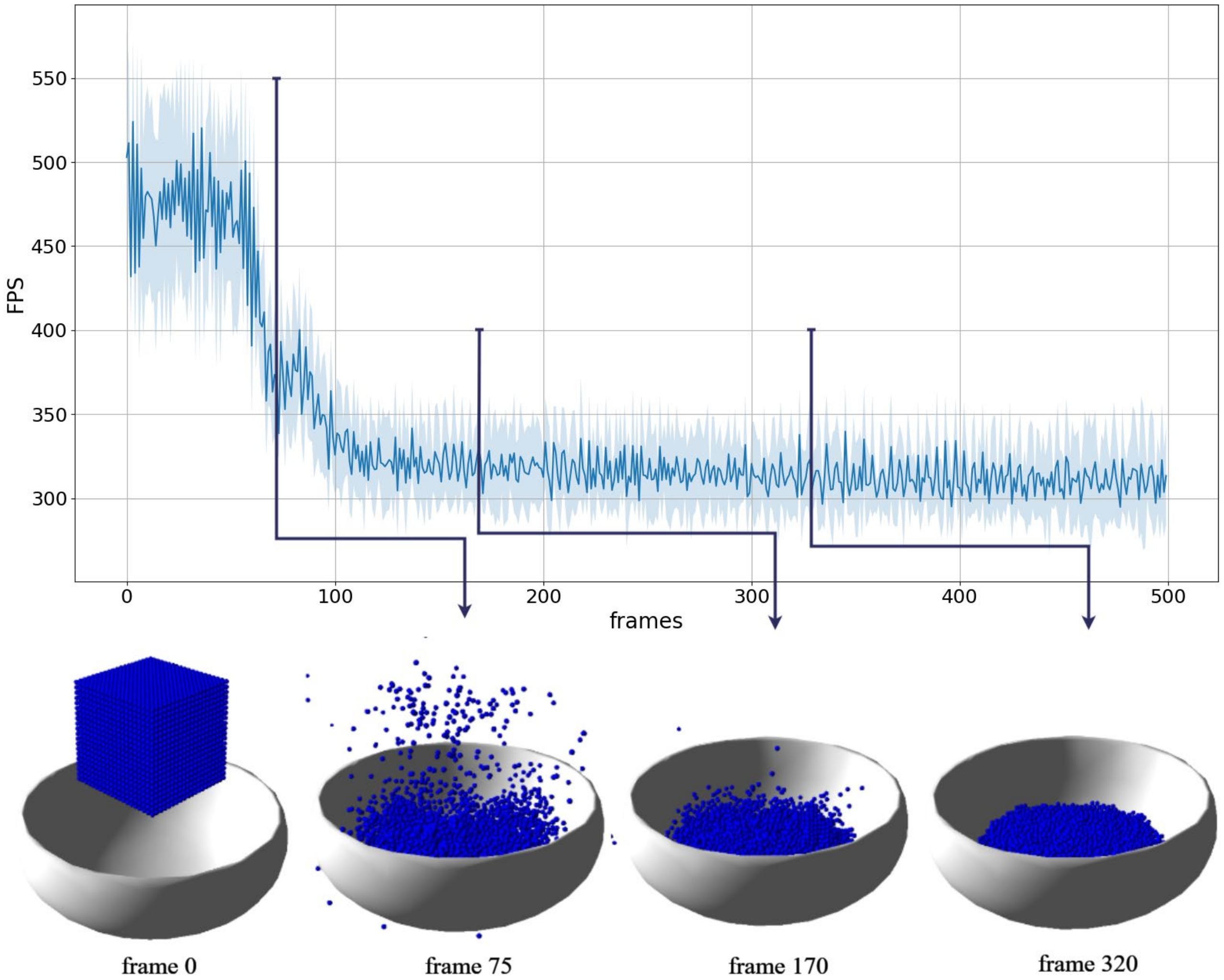


- Shader invocation
- GPU buffers
- Hash grids

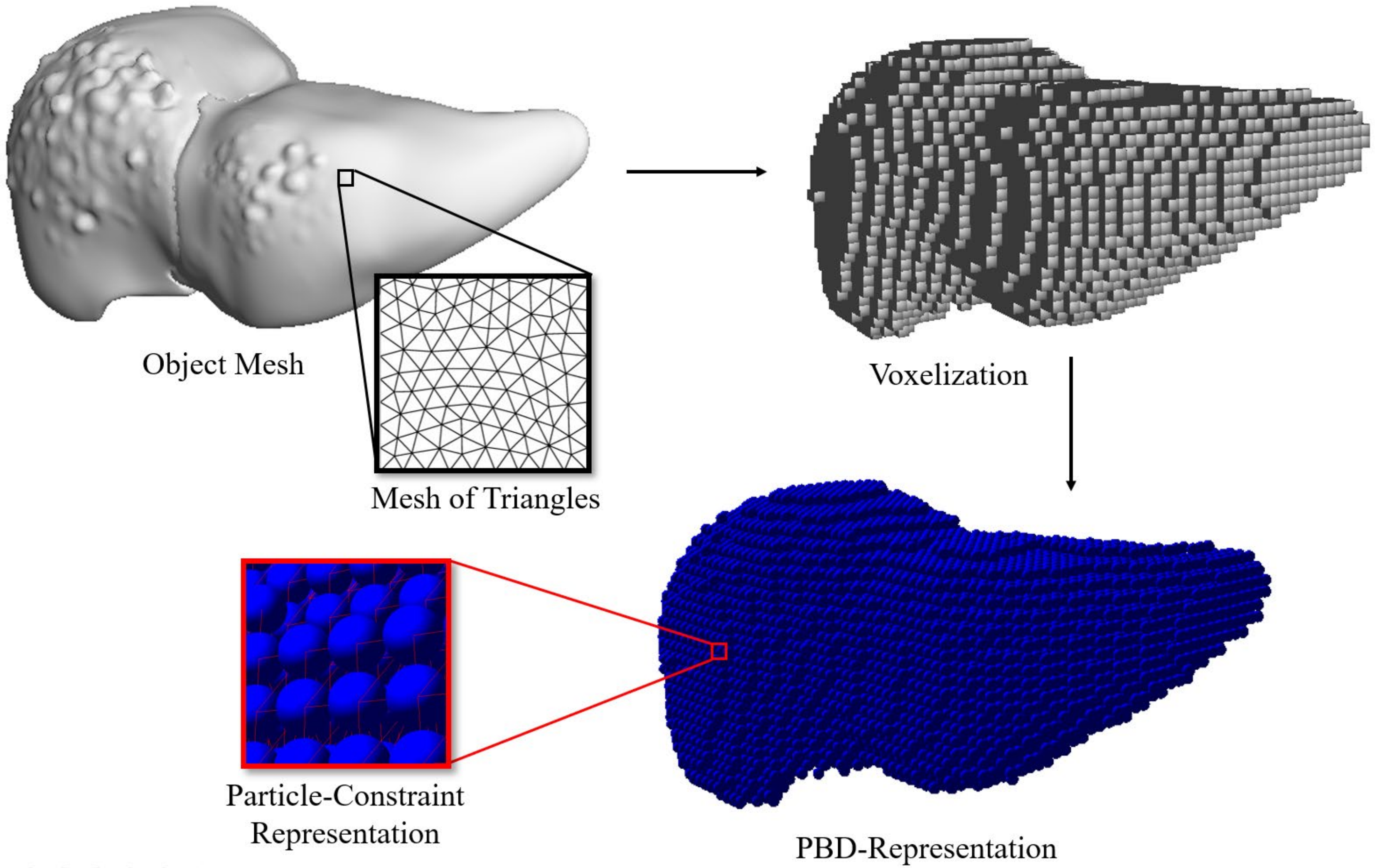
Collision Detection

- Realized by employing **hash grid-based** collision detection
  - For particles and faces
  - Based on position in space

- Depending on hash cell size, only neighboring cells are searched

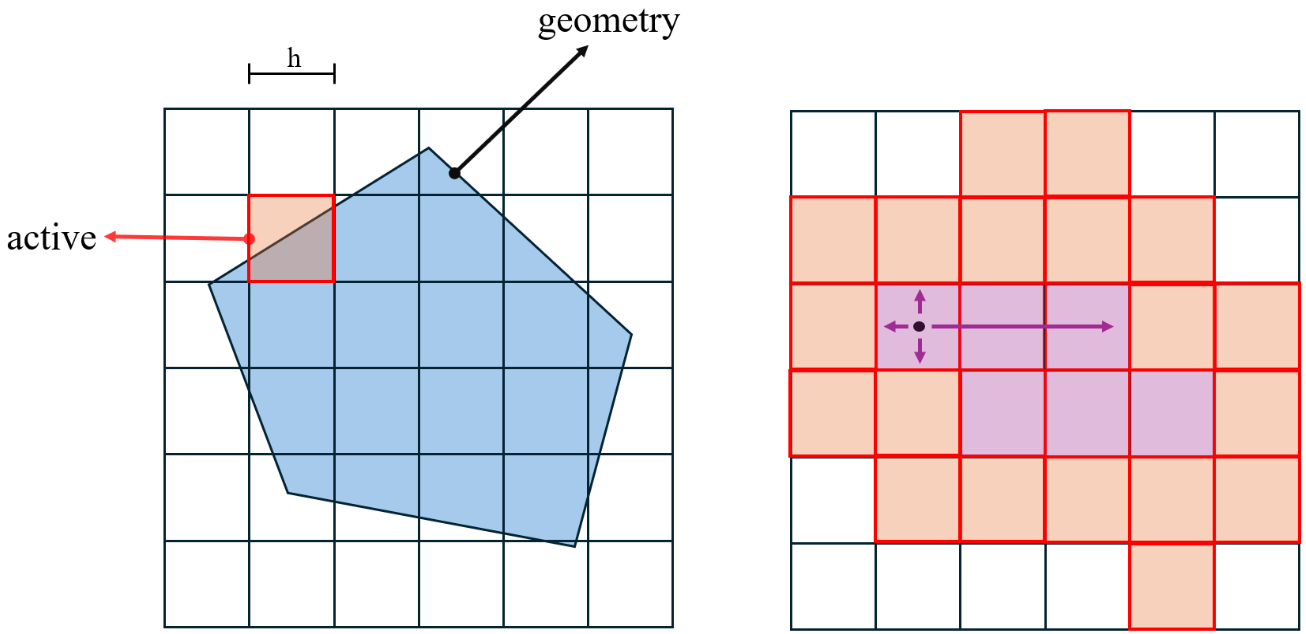


Model Descretization



Model Descretization

1) First a **two-stage voxelization process** turns loaded models in voxel representation
   - Essential for **volume preservation**
2) Then particles are placed in voxels and connected by distance constraints

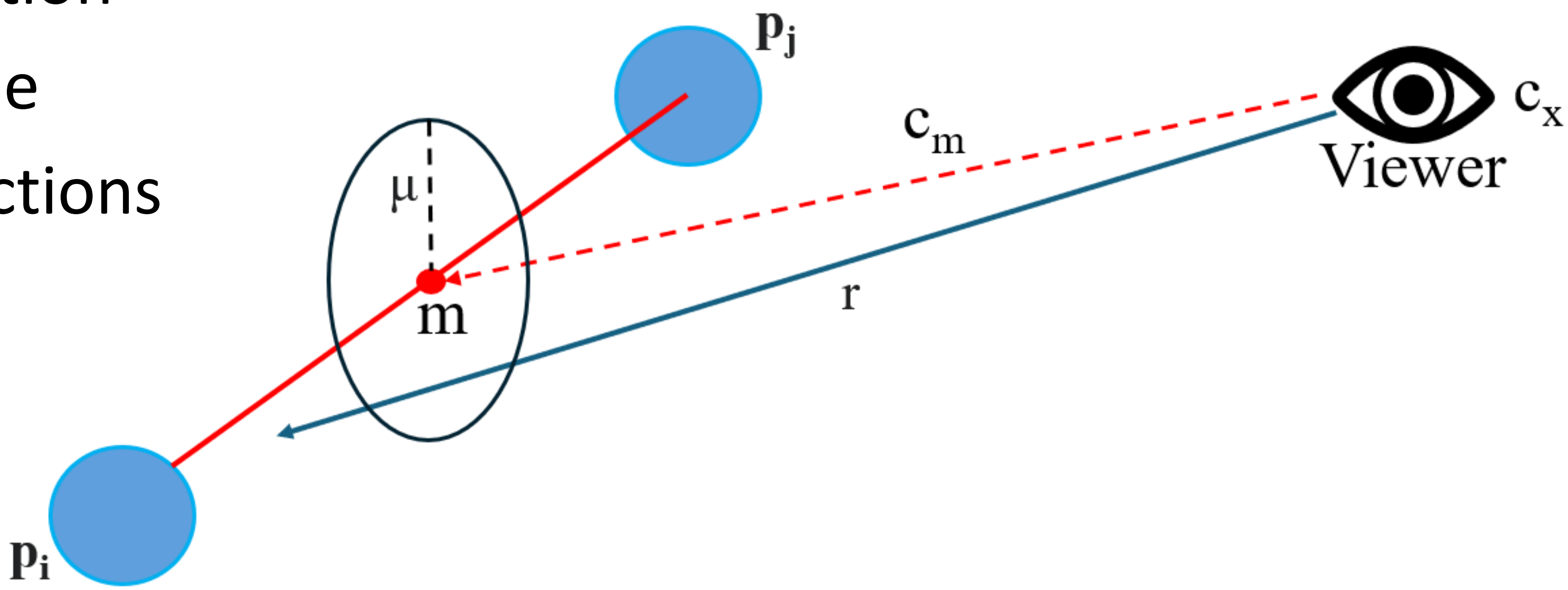


Hull voxelizer Core voxelizer

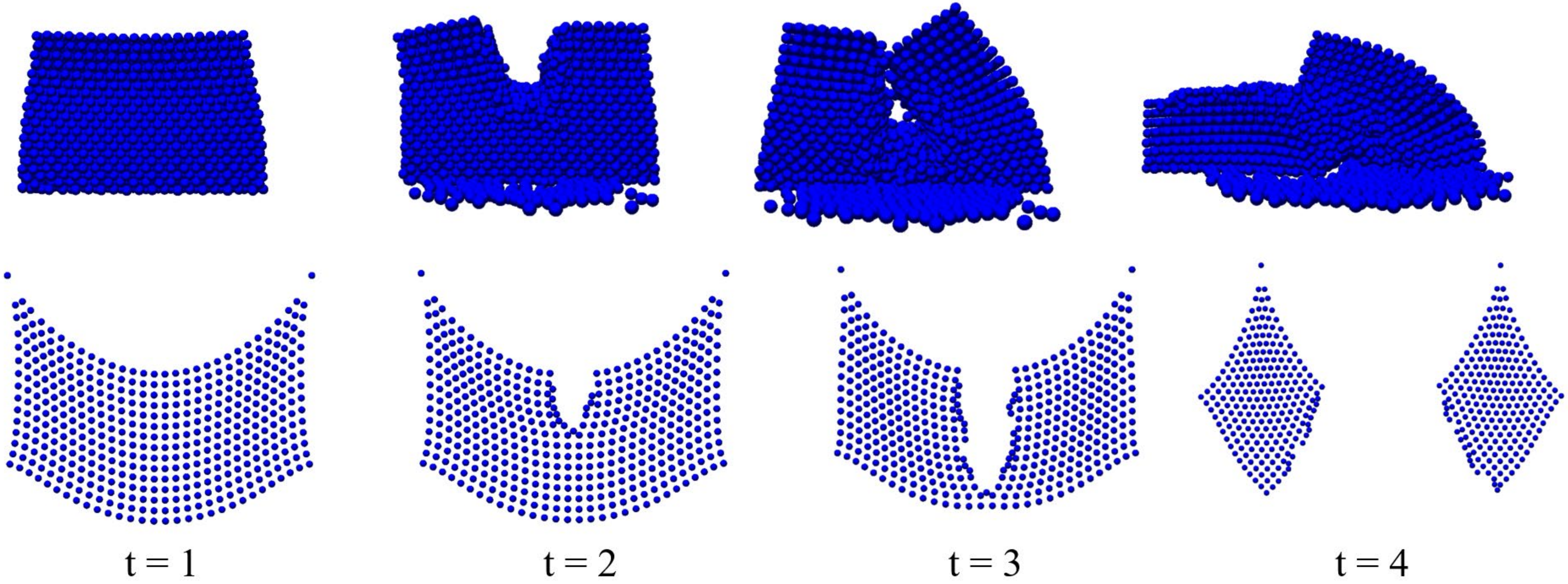
Cutting Algorithm

Based on **ray-casting** and **intersection testing** with distance constraints

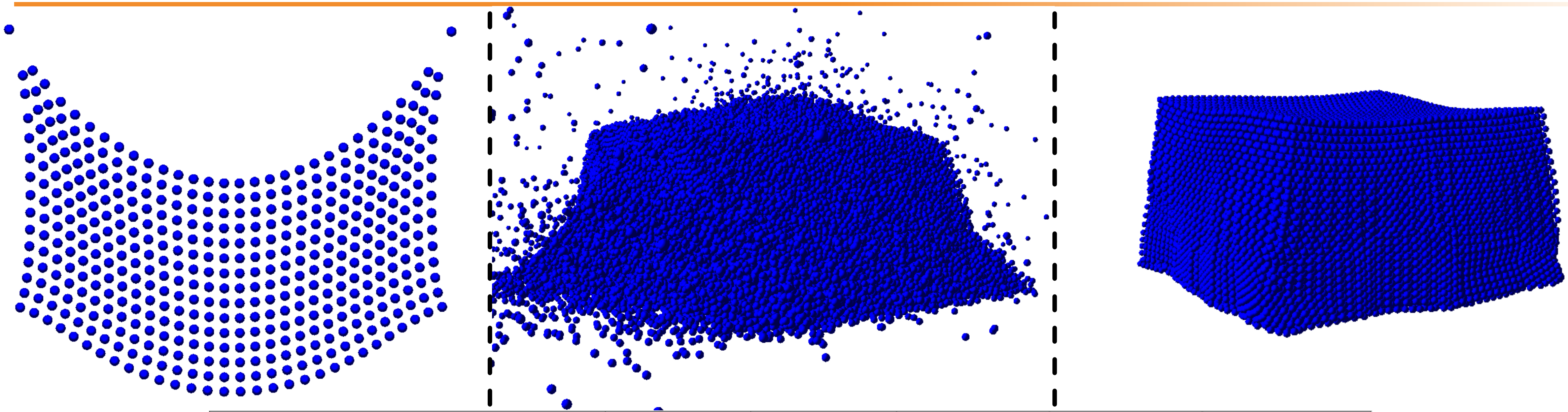
1) Track mouse position
2) Cast ray into scene
3) Check for intersections



Cutting Algorithm – Examples

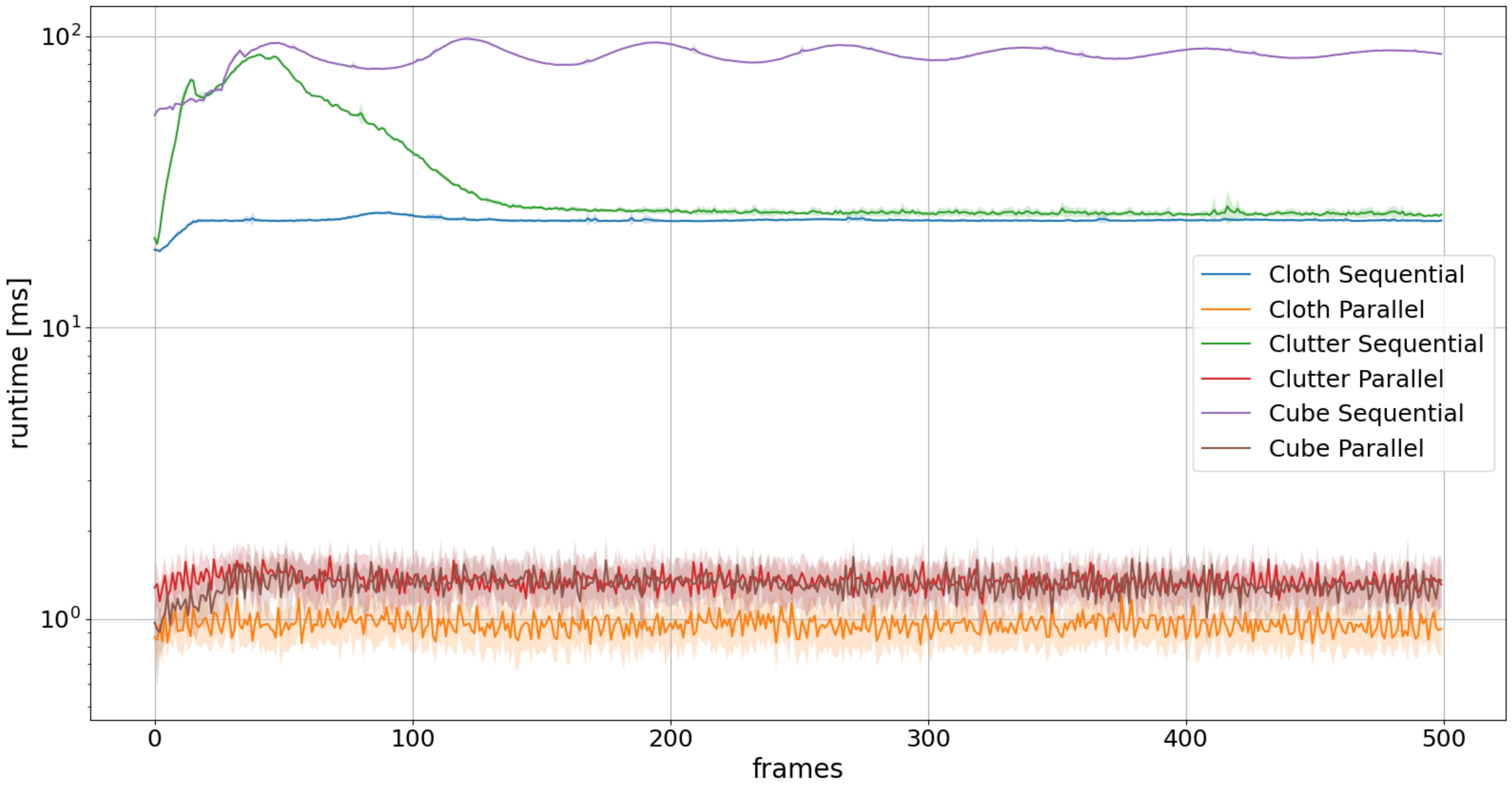


Results



Scene	FPS μ	FPS σ	Runtime μ [ms]	Runtime σ [ms]	Speedup
Cloth - Sequential	42.90	1.56	23.31	0.82	24.536
Cloth - Parallel	1048.64	235.74	0.95	0.18	
Clutter - Sequential	35.22	9.52	28.40	6.04	21.194
Clutter - Parallel	743.65	157.11	1.34	0.23	
Soft Body Cube - Sequential	11.81	1.35	84.68	8.67	65.138
Soft Body Cube - Parallel	766.30	180.47	1.30	0.25	

Results Visualization



References

- [1] M. Müller, B. Heidelberger, M. Hennix, J. Ratcliff “Position based dynamics”. Journal of Visual Communication and Image Representation 18(2), 2006. https://www.sciencedirect.com/science/article/pii/S1047320307000065
- [2] Joeyd de Vries. Learnopengl. (Website). Available online at: https://learnopengl.com