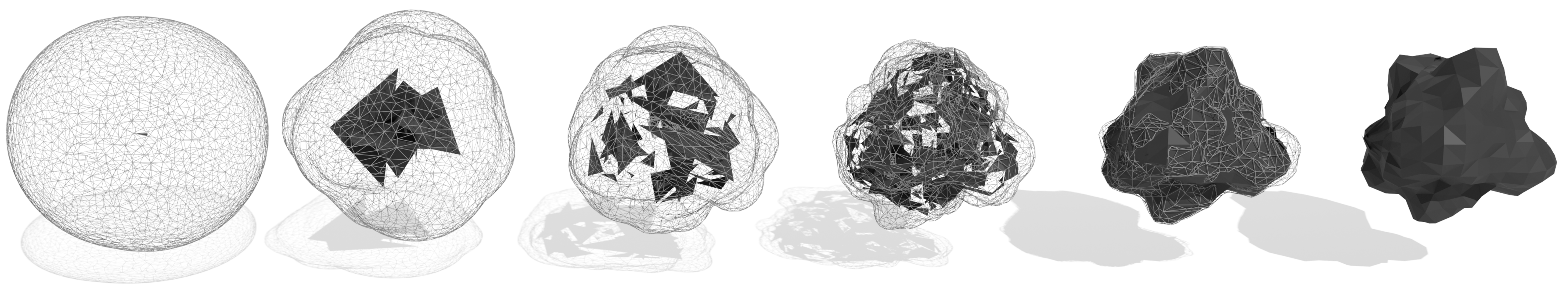
Durham University
Department of Computer Science

Discrete Element Methods for simulating non-spherical rigid body particles

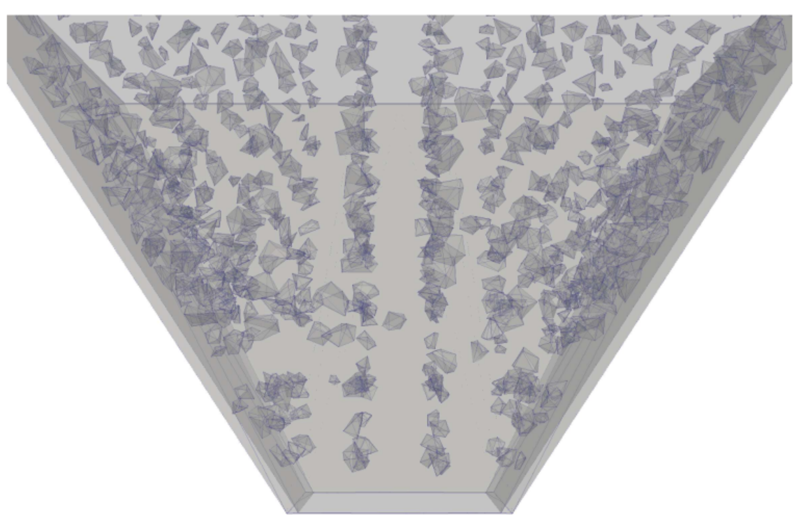
Peter Noble, peter.j.noble@durham.ac.uk   Tobias Weinzierl, tobias.weinzierl@durham.ac.uk



1. Introduction/Application

Discrete Element Methods (DEM) simulate the interaction of large numbers of rigid, incompressible objects with each other. Mainstream DEM codes focus on analytical shapes to streamline the identification of contacts between objects. This step dominates the simulation time. We manage to support triangulated particles with a wide variety of sizes in an efficient DEM code due to a combination of several new algorithmic ideas. Our model utilises the spring-dash-pot elastic repulsion force model.

DEM is used by a wide variety of scientific applications to help understand natural phenomena, engineering applications involving granular materials and in entertainment applications for games and film effects.



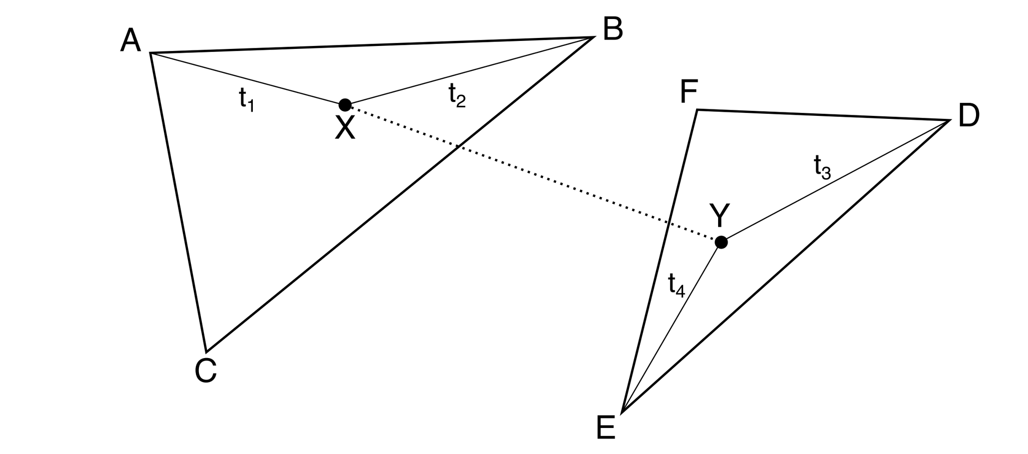
Hopper example application.

2. Triangle-triangle distance

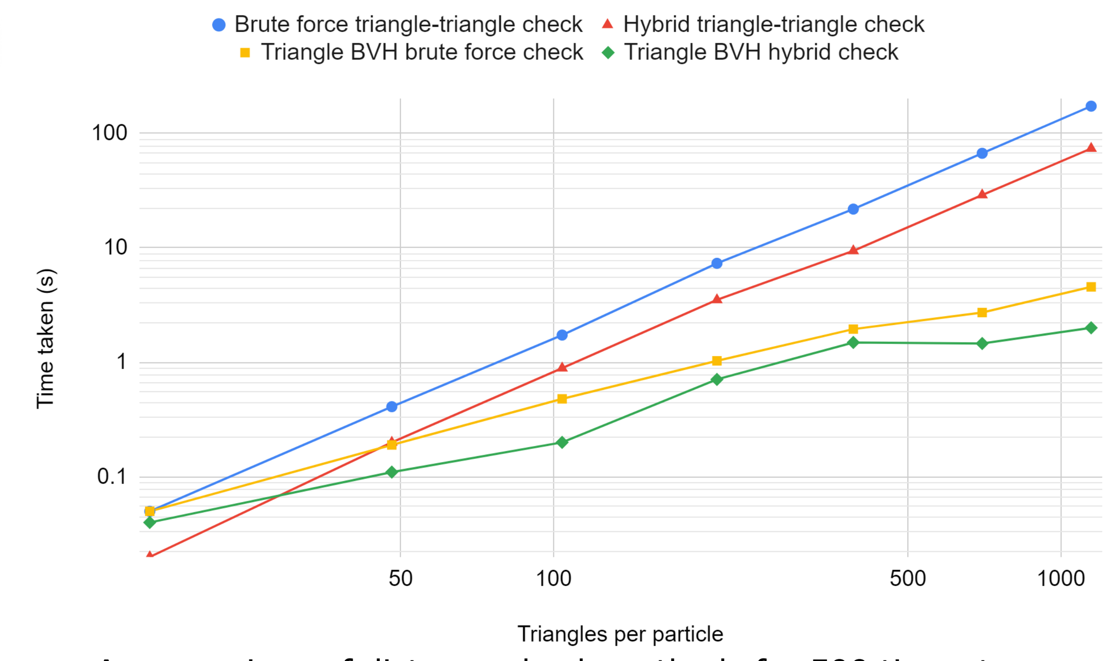
Triangle-triangle distance check formulated as minimisation problem over barycentric coordinates. Penalty functions are applied when constraints broken.

$$t_1 \geq 0,\ t_2 \geq 0,\ t_1 + t_2 \leq 1$$
$$t_3 \geq 0, t_4 \geq 0,\ t_3 + t_4 \leq 1$$
$$X = A \cdot t_1 + B \cdot t_2 + C \cdot (1 - t_1 - t_2)$$
$$Y = D \cdot t_3 + E \cdot t_4 + F \cdot (1 - t_3 - t_4)$$
$$\text{Minimise } |X - Y|$$

Efficient vectorisation. Fallback to robust (but expessive) method for ill-posed configurations.



The distance between a point, represented by barycentric coordinates, on each triangle is minimised to find the distance between the triangles.



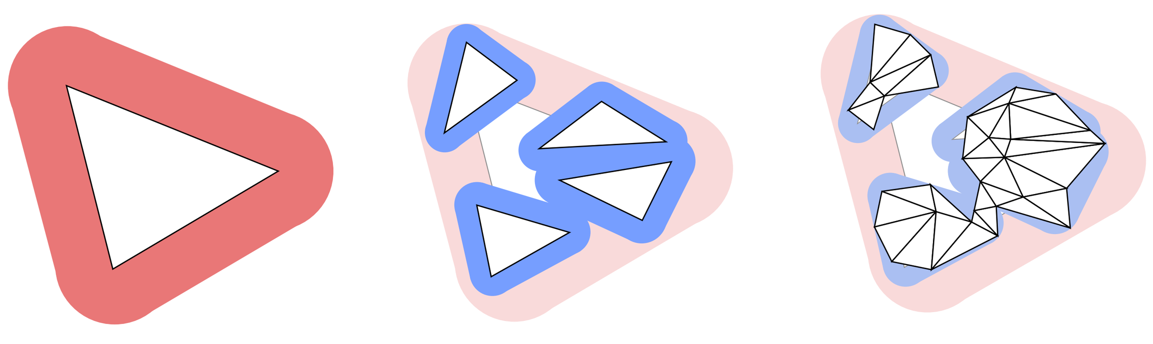
A comparison of distance check methods for 500 time steps

5. Triangle based hierarchy

Recursively constructed tree:

1) Randomly select triangles equal to the branching factor
2) Sort every original triangle into groups based on the distance to each of the selected triangles
3) For each group perform the triangle fitting method described below
4) Using the newly fitted triangle repeat from step (2) for a given number of iterations

High branching factor selected so that comparisons between hierarchies can be computed using SIMD instructions.
The triangle based hierarchy is easily transformed as the particle translates and rotates. Therefore, it doesn't have to be reconstructed each timestep.



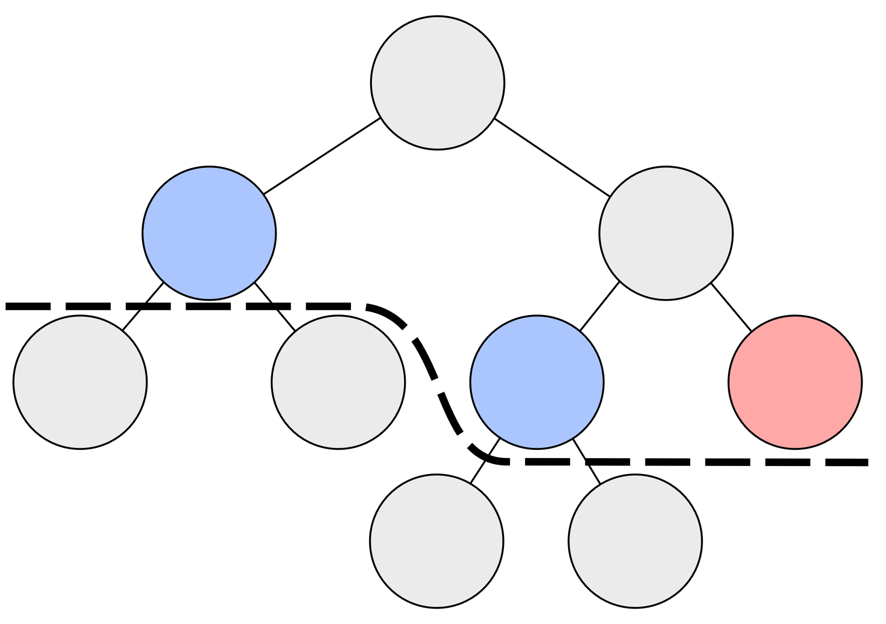
A 2D illustration of a triangle based bounding volume hierarchy. The volume of each branch of the tree is defined by a triangle and a radius.

7. Surrogate hierarchy results

Time taken in seconds for a three object 500 frame simulation.

Mesh size	Hybrid	Triangle hierarchy comparison based	Triangle hierarchy hybrid
16	0.04	0.05	0.04
48	0.20	0.19	**0.11**
390	9.40	1.95	**1.49**
1148	73.64	4.56	**2.00**

Department of Computer Science
Durham University
Durham University
Department of Computer Science



7. Active sets

We use surrogate hierarchies to create models of different resolutions by cutting the tree. A valid cut is where every path from the root to a leaf node crosses no more than one cut edge. The resulting leaf nodes form a surrogate model.
While the cut remains close to the root node there are few triangles in the surrogate model.

The tree is unfolded by removing a node and adding its children. The active set is updated with the children of inner nodes that result in a contact with another particles.

Finding initial contacts between two particle's active sets and updating them after is a more expensive operation than updating contact points after a transform update using the same active set.

```
state = initial_state_estimate();
while (!converged(state))
    active_sets = {}
    for (s in surrogate levels)
        contacts = find_contacts(state, active_sets)
        active_sets = update_active_sets(active_sets, contacts)
    state = update_state(state, contacts)
```

```
active_sets = {}
state = initial_state_estimate();
for (s in surrogate levels)
    while (!converged(state))
        contacts = find_contacts(state, active_sets)
        state = update_state(state, contacts)
    active_sets = update_active_sets(active_sets, contacts)
```

Our two versions of the Picard algorithm. Top - The active set is only used to accelerate finding contacts. Bottom - The loops are permuted. We wait for convergence before updating the active set. The number of active set updates is reduced.

7. Implicit time stepping

We achieve an implicit time stepping scheme through Picard iterations. Early predications of the future state are iteratively used to computer increasingly accurate predictions. A lot of effort is expended computing contact points where the finest level surrogate model is used. However, early iterations may have a bad approximation of the transformation so the contact points won't be accurate.

We propose an alteration to the algorithm where contacts generated using the **coarser level surrogates are used to estimate the forces during early iterations**. The aim is to reduce the number of expensive fine level iterations that are required.

7. Multiscale Picard results

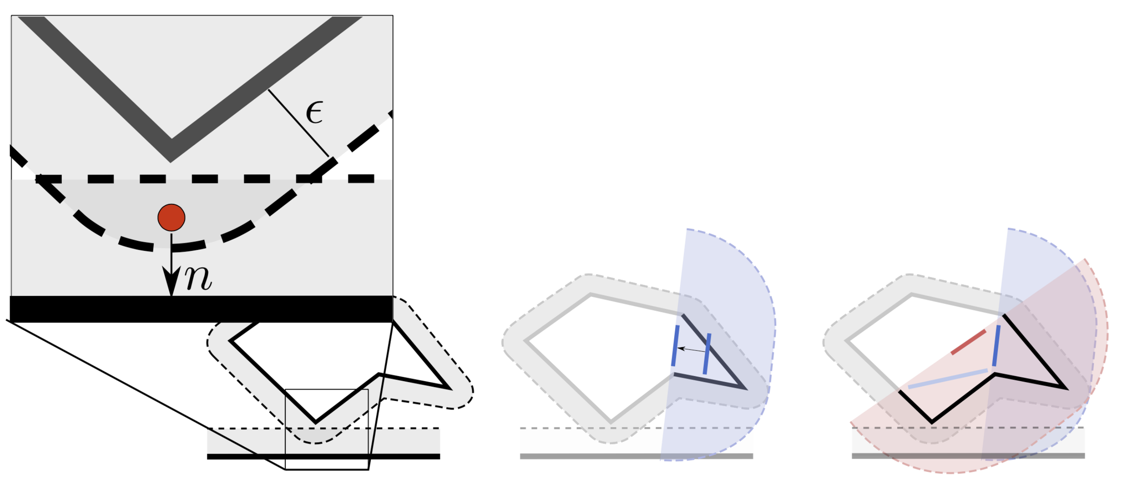
Time taken for a fixed number of implicit time steps using 1) no acceleration data structure 2) surrogate hierarchies to accelerate contact point look-ups inside a Picard iteration 3) our multi scale Picard algorithm.

Scenario	Method	Comparison based	Hybrid
Particle-particle	Single level	169.16	74.42
	Surrogate within Picard	1.08	0.72
	Multiscale Picard	0.77	**0.47**
Particle-plane	Single level	96.06	44.10
	Surrogate within Picard	0.48	0.22
	Multiscale Picard	0.49	**0.19**

7. Complex particle results

Time taken for a fixed number of implicit time steps for a collision between two particles. One particle with 80 triangles and the other with a varying number of triangles.

Triangles	80	320	1,280	5,120
Runtime	0.087	0.23	0.26	0.24



Contacts are located in the middle of the overlap regions between two particles. Two levels of surrogates are shown (blue and red). The volume defined by a surrogate doesn't have to encapsulate child surrogates but it much encapsulate the original mesh.

8. Continuous collision detection

Selecting an appropriate time step size is important but challenging when the velocity of each particle can vary so much compared to the interaction distance.
The time of contact can be estimated by introducing a time component to the triangle-triangle distance check.
This can be thought of as a 4D mesh where we search for contacts.

8. Future work - Variable precision

Lower precision used when comparing the upper levels of the hierarchy (with extra tolerance to avoid introducing errors).
Future ideas
GPU variable precision to add 16 bit (even lower in the future).
Store arrays of double as two arrays of floats (approximation and a delta). Begin computation on the first while loading the second for later iterations.

References

Krestenitis K, Weinzierl T, Koziara T. Fast DEM collision checks on multicore nodes. In: Parallel Processing and Applied Mathematics; 2018; Lublin, Poland.
Krestenitis K, Weinzierl T. A multi-core ready discrete element method with triangles using dynamically adaptive multiscale grids. Concurrency Computat Pract Exper. 2019;