Durham University
Department of Computer Science

Department of Computer Science
Durham University

Durham University
Department of Computer Science

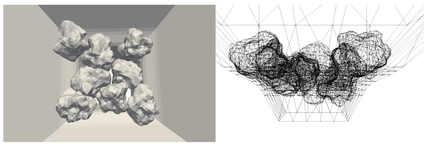
An implicit time stepping, multiresolution Discrete Element Method for triangulated objects

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 identification dominates the simulation time. Once we switch to implicit time stepping, it becomes overwhelming.

We manage to support triangulated particles with efficient implicit DEM code using a multiresolution and multiprecision hierarchy of shapes. Early Piccard iterations of the implicit solve are done with lower geometric resolution and precision and are significantly quicker to evaluate than the full resolution geometry while still converging to an approximate solution. Early iterations yield an initial guess and thus reduce the number of later, more expensive iterations that are required to converge to the correct solution.



A simple example of a hopper getting jammed

2. Triangle-triangle distance

Geometry distance checks are calculated using a distance minimisation algorithm.
• The precision can be tuned to produce a result only as precise as we need.
• Early Piccard iterations with low geometric precision only require a few iterations.
• Late Piccard iterations with fine geometric resolution can use more iterations.

Triangle-triangle distance check formulated as minimisation problem over barycentric coordinates. Penalty functions ensure constraints are weakly enforced.

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

Efficient vectorisation. Fallback to robust method for ill-posed configurations, which is also vectorised but is slower.

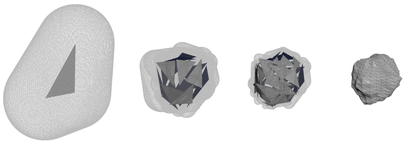
We utilise a computational algebra package with code generation to implement this minimisation.

3. Triangle based hierarchy

Recursively constructed tree:

1) Randomly select triangles depending on a branching factor
2) Sort every original triangle into groups based on the distance to each of the selected triangles
3) For each group construct a surrogate triangle by greedily selecting 3 vertices from the children to minimise the maximum distance between the surrogate and each of it's children.
4) Using the newly fitted triangle repeat from step (2) for a given number of iterations

High branching factor selected to maximise 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.



The search radius around each triangle is shown as a dotted mesh. Left: The top level of the hierarchy. A single triangle with a check distance. Mid left: The second level of the hierachy. Notice the tighter bounds. Mid right: All the remaining levels of the hierachy. Has the tightest bounds. Right: The original mesh.

4. Inner and outer epsilon

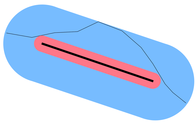
Like a traditional BVH each node has a volume, defined as all points within ϵ distance of the parent triangle, such that:

$$\forall P : \text{CONTACT}(P, child) \rightarrow \text{CONTACT}(P, parent^{\epsilon})$$

Additionally we introduce a second volume, defined as all point within E of the parent triangle, such that:

$$\forall P : \text{CONTACT}(P, parent^{E}) \rightarrow \text{CONTACT}(P, child)$$

We use E to estimate the force on an object without reaching the fine resolution in a way that won't push the estimate away from the final value.



A 2D example of a surface (thin line), a surrogate triangle (thick line), an outer volume defined by Ɛ (blue) and an inner volume defined by E (red).

Durham University
Department of Computer Science

Department of Computer Science
Durham University

Durham University
Department of Computer Science

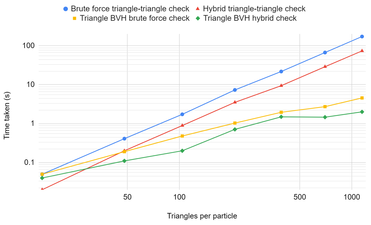
6. Explicit Euler

Algorithm A.1 Multiresolution explicit time stepping.

```
S: The current state of simulation
C ← ∅
for R ∈ RESOLUTIONS do
    C ← CONTACTS(S, R, C)
end for
S ← S + STATE_CHANGE(C)
```

The simpliest way to apply the multiresolution model is to use it as a tree data structure to accelerate the lookup of contact points. Starting from the coarsest level of the tree step down the levels searching for new contacts at each level of the tree based on where there were contacts on the previous level. Once the finest resolution level of the tree is reached use the current set of contact points to calculate the forces exerted on the objects.

A comparison of distance check methods for 500 time steps using explicit timestepping



7. Implicit Euler

Algorithm A.2 Multiresolution Piccard implicit time stepping.

```
S: The current state of simulation
P: The current prediction of the state update
while ΔP > ε do
    C ← ∅
    for R ∈ RESOLUTIONS do
        C ← CONTACTS(S + P, R, C)
    end for
    P ← STATE_CHANGE(C)
end while
S ← S + STATE_CHANGE(C)
```

We implement implicit Euler timestepping by wrapping the explicit loop in a piccard iteration. In each iteration we take the current prediction of the future state and use that to construct a better approximation of the future state.

Algorithm A.3 Multiresolution Piccard implicit time stepping with inverted loops.

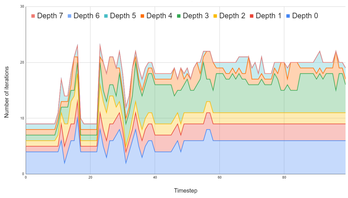
```
C ← ∅
S: The current state of simulation
P: The current prediction of the state update
for R ∈ RESOLUTIONS do
    while ΔP > ε do
        C ← CONTACTS(S + P, R, C)
        P ← STATE_CHANGE(C)
    end while
end for
S ← S + P
```

Our key idea is to permute the order of the loops. Instead of repeatedly traversing the tree data structures we instead perform in-situ force calculations based on the contact points generated at the coarse levels of the tree.

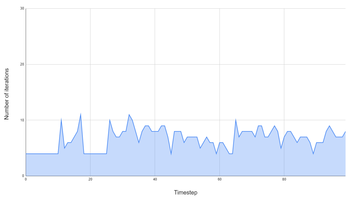
Scenario	Full Runtime |s|	Full Hybrid Runtime |s|	Hierarchy Hybrid Runtime |s|
Particle collision	1e+03	718	2.98
Particle on slope	485	336	14.6
Hopper	2.73e+04	1.88e+04	126

Runtimes for 3 different example applications.

The number of iterations taken per timestep for a sphere that bounces once on a slop and then roles down it



The same graph using full resolution iterations only. Each of these iterations is about as expensive as a depth 6 iteration in the multiresolution model



8. Performance benefits

Tree traversal comparisons
By traversing the data as a tree fewer triangle-triangle distance checks and comparisons are needed.
Tree traversal memory access
Only the sections of the object's mesh that are in contact with another object are required so only these need to be read from memory.
Permuted inner loops contact detection
While in the early iterations where coarse resolution geometry is used the contact detection is faster to perform. Early iterations are cheap and reduce the number of later more expensive iterations.
Permuted inner loops memory access
By traversing the tree only once we can ensure that most nodes are visited at most once. Only situations where the state is alterned enough to bring a new section of the mesh into contact do we need to load more of the tree. More work is performed on each item we load from memory.

Scenario	L1-L2 Full	L1-L2 ours	L2-L3 Full	L2-L3 ours	L3-system Full	L3-system ours
Particle collision	410	101	9.29	27.1	11.6	13.8
Particle on slope	186	24.2	1.14	4.75	3.52	2.51
Hopper	1.01e+04	300	63	61.5	94	28.1

Data transfer volume (GBytes) between memory cache levels for a full resolution only simulation and our hierarchy hybrid method

References

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