A Novel Approach to the Simulation of Particles on a Large Size-Range

Alexander Schinner

Otto-von-Guericke Universität Magdeburg,

Germany

1997

MD-Simulation of granular materials

$\Downarrow$

slowly changing system

$\Downarrow$

The last time-step's information is nearly correct.

$\Downarrow$

Reuse this information as good starting conditions.

The steps of our MD-simulation

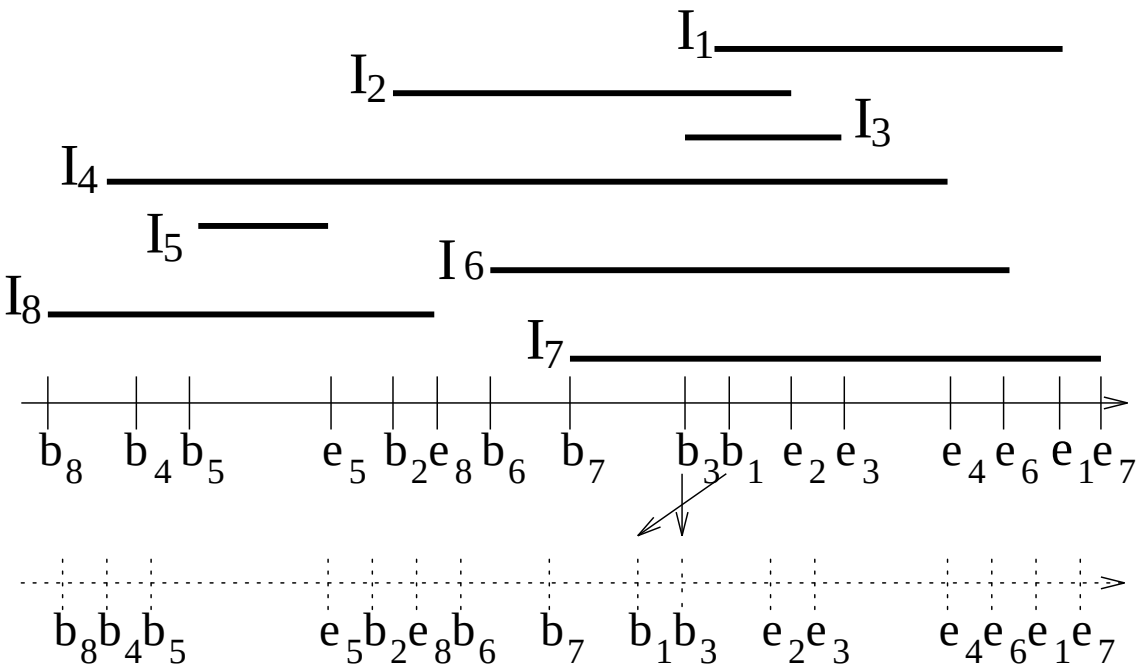
1. Check for bounding box collisions
2. Check for particle collisions
3. Calculate overlap
4. Do the physics (not discussed here)

Increasing need of time

Try to exclude as many particle-pairs as possible from being considered in the next level.

Bounding Boxes

First consider the one dimensional case:

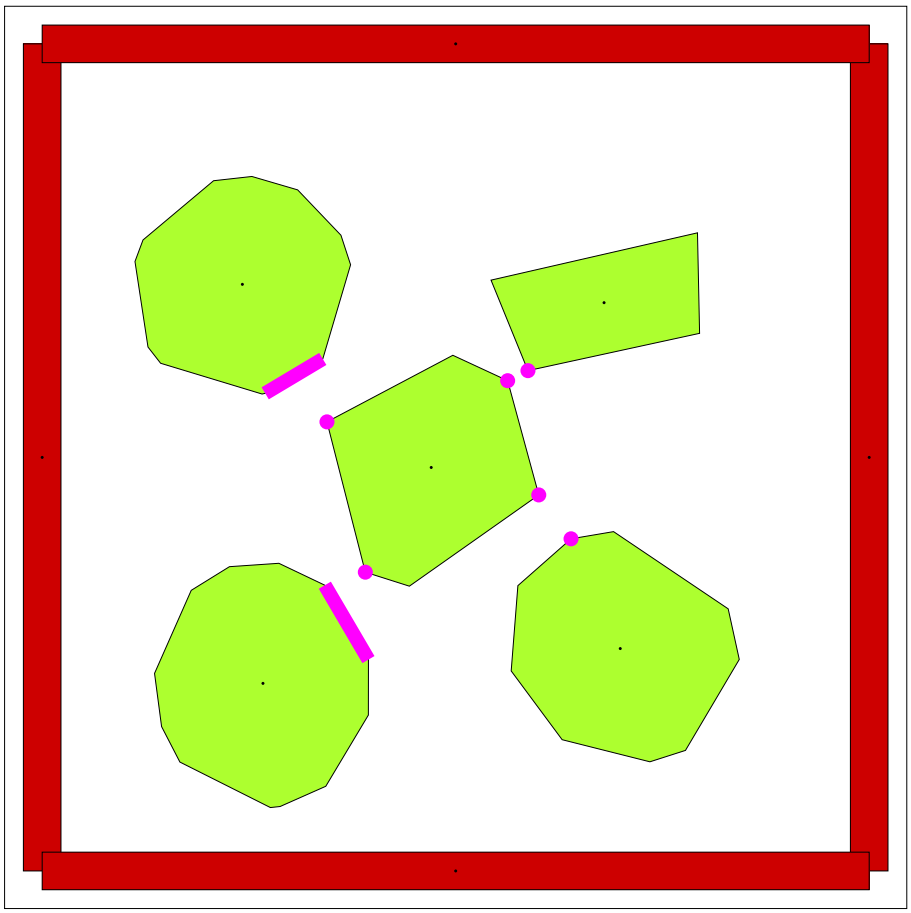


- Resorting the List of Boundingbox boundaries is possible with $\mathcal{O}(N)$ steps, if we use Insertion Sort
- During the sorting we can update the list of all collisions

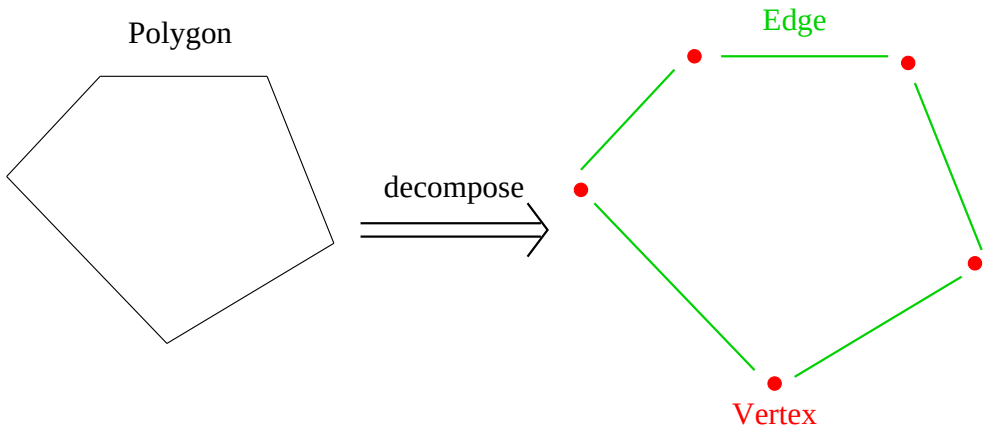
$\Rightarrow$ Total cost of $\mathcal{O}(N)$.

Closest Feature Algorithm

- Calculate distance of particles
- keep track of the closest features
- local test to confirm the distance



Feature



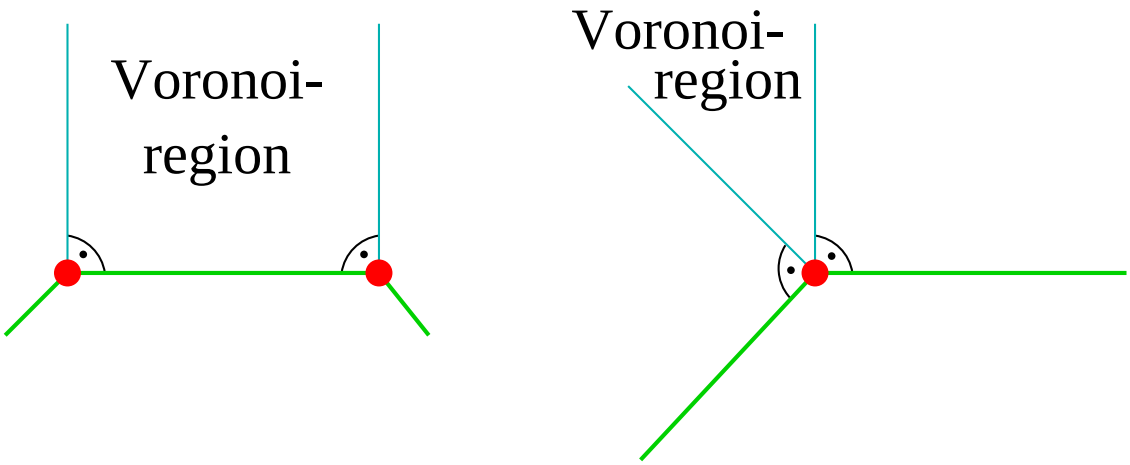
A feature has information on its

- type (edge or vertex)
- geometry
- neighbors
- Voronoi regions
- and more...

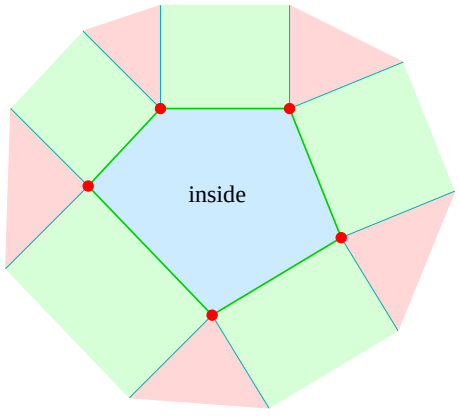
Voronoi Region

If a point p lies inside the Voronoi region of a feature f_a, then

$$\text{dist}(p, f_a) \leq \text{dist}(p, f_b)$$

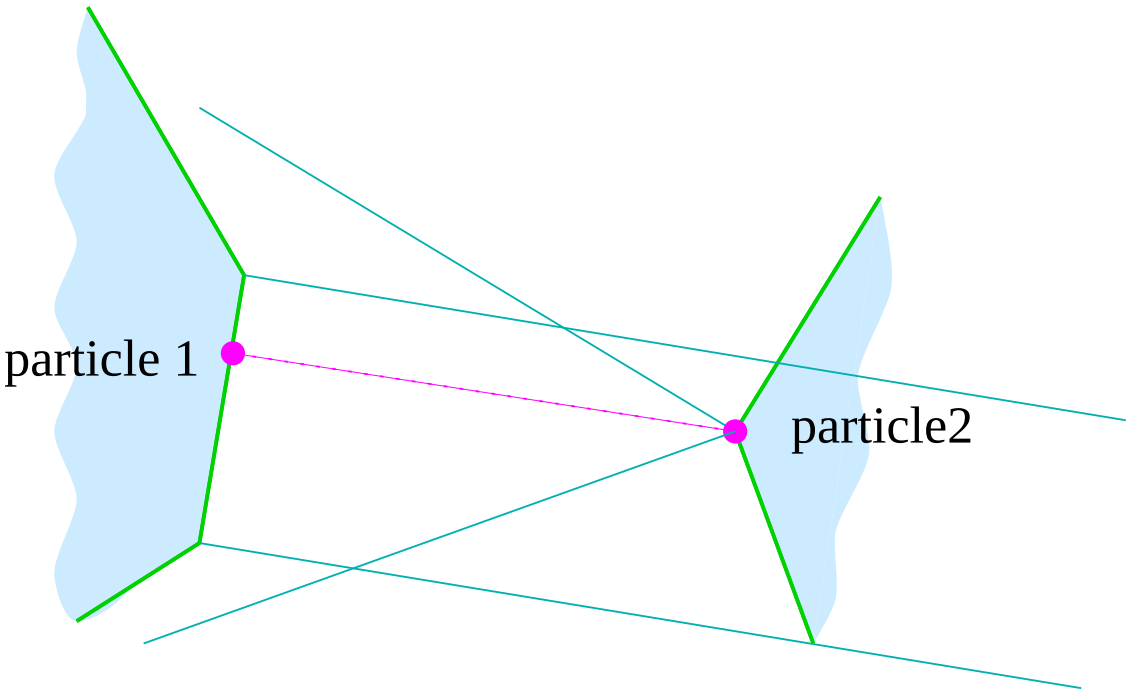


This divides the space outside a particle:



Closest features

If a point P on object P_1 lies inside the Voronoi region of f_2 on object P_2, then f_2 is a closest feature to the point P and vice versa for an Voronoi region of f_1. If we have a pair of features fulfilling the above condition, we have a pair of closest features.



Closest Feature Algorithm

We are looking on two features f_1 and f_b on two polyhedra P_1 and P_2.

1. Calculate the Voronoi regions V_1 and V_2
2. Calculate a point p_1 on f_1 that is the closest to f_2 and a point p_2 on f_2 that is the closest to f_a
3. Check for $p_1 \in V_2$. If not: choose new f_1 and restart algorithm
4. Check for $p_2 \in V_1$. If not: choose new f_2 and restart algorithm