Learning 3D Granular Flow Simulations

Andreas Mayr[1] Sebastian Lehner[1] Arno Mayrhofer[2] Christoph Kloss[2] Sepp Hochreiter[1,3] Johannes Brandstetter[1,4]

[1] ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria
[2] DCS Computing GmbH, Linz, Austria
[3] Institute of Advanced Research in Artificial Intelligence (IARAI)
[4] University of Amsterdam, Amsterdam, Netherlands

Problem Statement & Main Contributions

Background:

- Granular flows ubiquitous in nature and many industrial processes
- **No underlying governing equations** for general granular flow exist!
- Simulations with the **Discrete Element Method** (DEM; Cundall et al., 1979): Granular flow simulation data with open-source DEM software LIGGGHTS (Kloss et al., 2012)
- LIGGGHTS allows simulation of particulate flows:
  - wide range of materials
  - complex **mesh-based wall geometries**

  ⇒ enables simulation of relevant industrial processes
- Interest in machine learning models, that can predict simulation trajectories
  ⇒ Gaining speedup by machine learning models

Compared to previous work (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020):
Focus on learning 3D granular particle flow simulations with nontrivial geometric boundary conditions

Main contributions:

- Triangular geometric boundaries for Graph Neural Networks (GNNs)
- Orientation independence of normal vectors
- Compare and analyse simulated processes

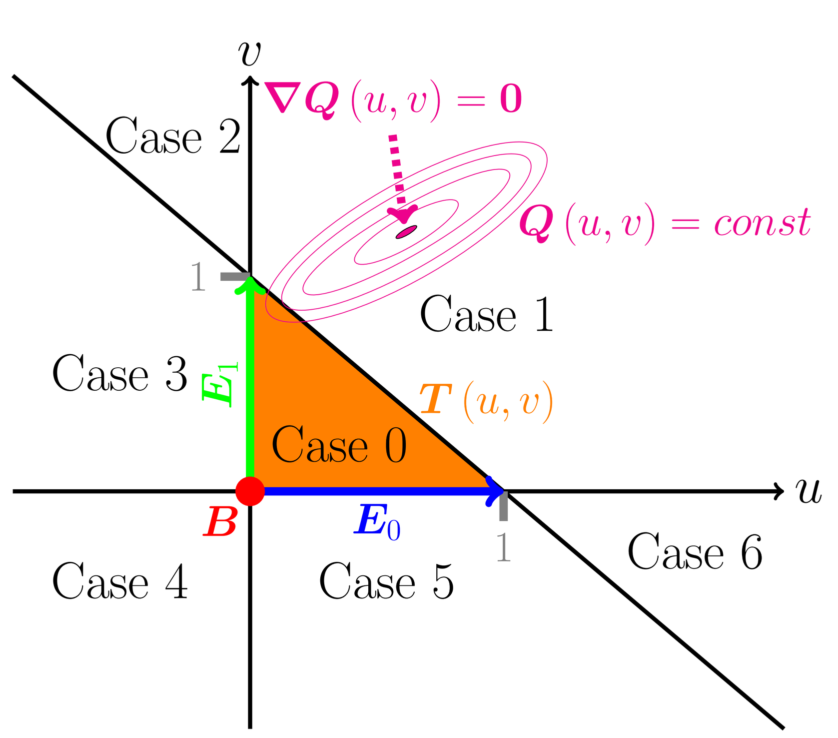
Time Transition Model: $t_k \rightarrow t_{k+1}$

suggested by Sanchez-Gonzalez et al. (2020)

$$\dot{\boldsymbol{p}}^{t_{k+1}} = \dot{\boldsymbol{p}}^{t_k} + \Delta t\, \ddot{\boldsymbol{p}}^{t_{k+1}}$$
$$\boldsymbol{p}^{t_{k+1}} = \boldsymbol{p}^{t_k} + \Delta t\, \dot{\boldsymbol{p}}^{t_{k+1}}$$

- based on GNNs with an encoder-processor-decoder architecture
  - **encoder**: construct neighbourhood graph, retrieve node and edge embeddings
  - **processor**: message passing neural network
  - **decoder**: extraction of acceleration
- $\boldsymbol{p}$: particle location
- $\dot{\boldsymbol{p}}$: particle velocity
- $\ddot{\boldsymbol{p}}$: particle acceleration (**to be predicted**)
- $\Delta t = 1$ (fixed)
- usage of relative encoder version:
  - take only relative positional information into account

Triangular Geometric Boundaries



$$\boldsymbol{T}(u,v) = \boldsymbol{B} + u\, \boldsymbol{E}_0 + v\, \boldsymbol{E}_1$$

- geometry described by triangular mesh
- static boundary particles ⇒ large number of additional particles
- insert virtual particles as needed into graph
  - needs distances from particles to triangles
  - usage of algorithms as adopted from Eberly (1999) (see figure)

Orientation Independence

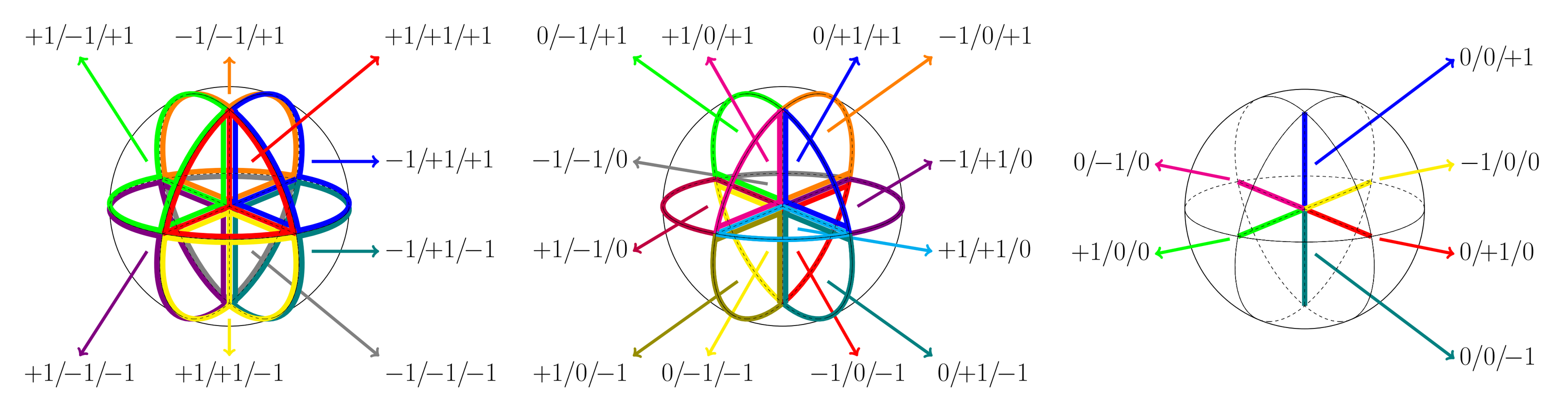
Particle - Wall Interactions:

- Normal vector components as features to describe walls
- Vector representation is orientation dependent, while particle - wall interactions do not dependent on this representation!

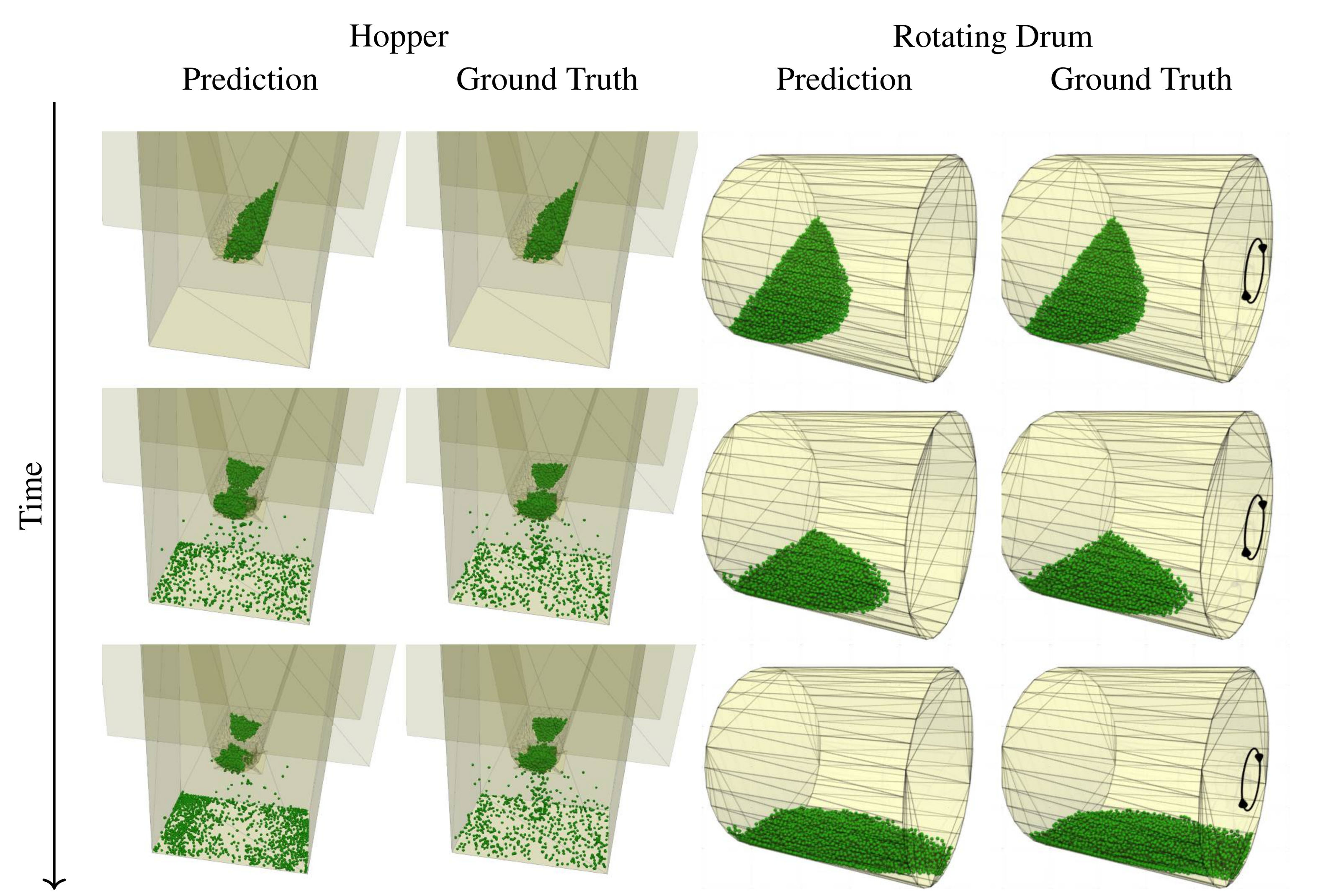
Just using both orientations has problem that an order still is there.

⇒ Define partial ordering:

$$f_o(\boldsymbol{n}) = \sum_{i=1}^{3} 3^{i-1} \left(\operatorname{sgn}(n_i) + 1\right)$$
$$o_1 = f_o(\boldsymbol{n})$$
$$o_2 = f_o(-\boldsymbol{n})$$
$$\Downarrow$$
$$repr(\boldsymbol{n}) = \begin{cases} \boldsymbol{n}, -\boldsymbol{n} & \text{if } o_1 \leqslant o_2 \\ -\boldsymbol{n}, \boldsymbol{n} & \text{otherwise} \end{cases}$$



Application to Granular Flow Data

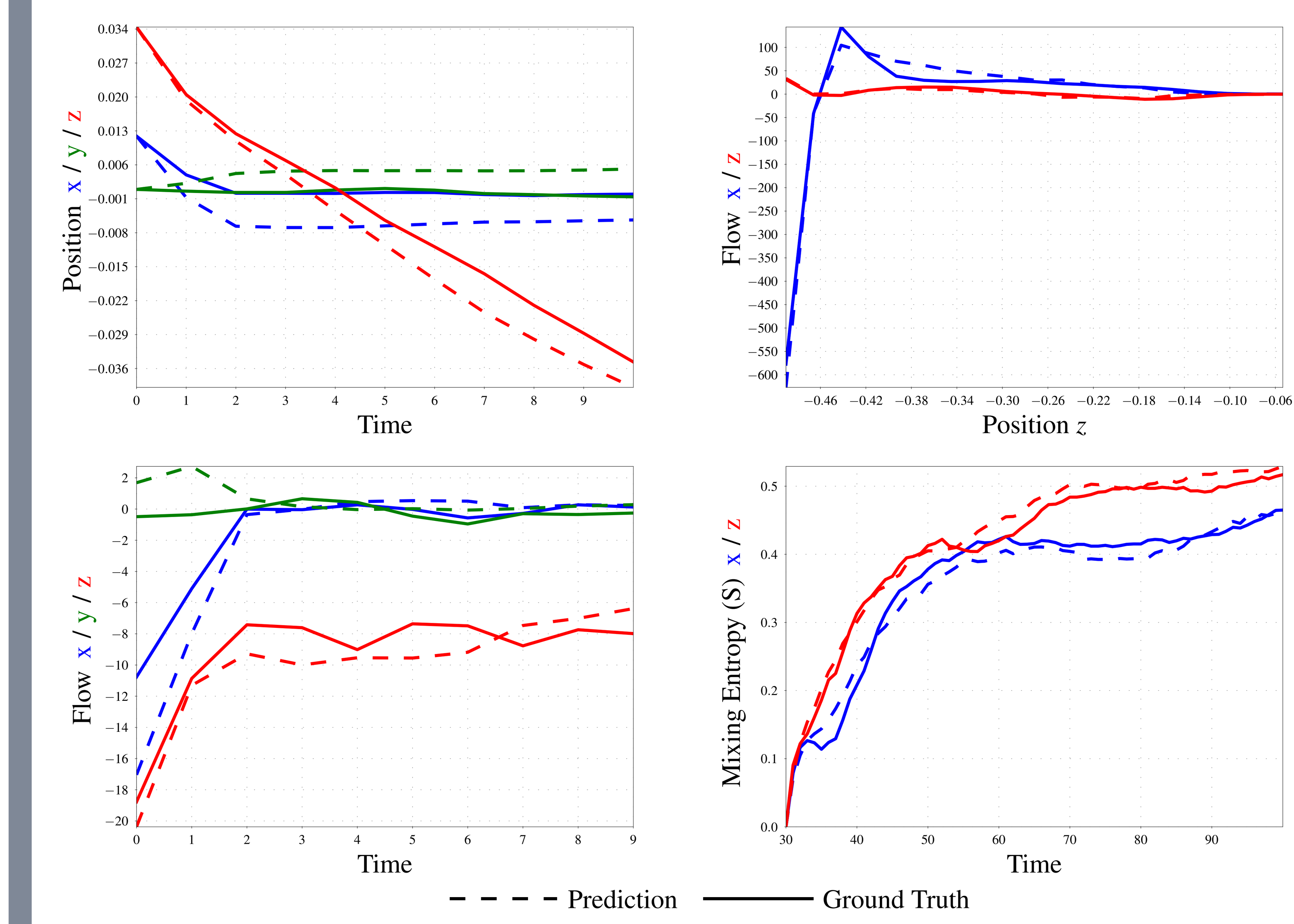


Mixing Entropy

- proposed by Lai et al. (1975)
- quantify extend of particle mixing
- local entropy $s(\boldsymbol{x}_{klm}, t)$ at grid cell $\boldsymbol{x}_{klm}$
- splitting particles into two classes +1, -1 at a certain time step t_0

$$n(\boldsymbol{x}_{klm}, t) = n_{+1}(\boldsymbol{x}_{klm}, t) + n_{-1}(\boldsymbol{x}_{klm}, t)$$
$$f_{\pm 1}(\boldsymbol{x}_{klm}, t) = \frac{n_{\pm 1}(\boldsymbol{x}_{klm}, t)}{n(\boldsymbol{x}_{klm}, t)}$$
$$s(\boldsymbol{x}_{klm}, t) = -f_{+1}(\boldsymbol{x}_{klm}, t) \log f_{+1}(\boldsymbol{x}_{klm}, t) - f_{-1}(\boldsymbol{x}_{klm}, t) \log f_{-1}(\boldsymbol{x}_{klm}, t)$$
$$\mathrm{S}(t) = \frac{1}{\sum_{k,l,m} n(\boldsymbol{x}_{klm}, t)} \sum_{k,l,m} n(\boldsymbol{x}_{klm}, t) s(\boldsymbol{x}_{klm}, t)$$

Analysis of ML Simulation Outputs



References

Cundall P. et al. (1979). A discrete numerical model for granular assemblies.

Eberly D. (1999). Distance between point and triangle in 3D.

Kloss C. et al. (2012). Models, algorithms and validation for opensource DEM and CFD-DEM.

Lai F. et al. (1975). Application of a Discrete Mixing Model to the Study of Mixing of Multicomponent Solid Particles.

Pfaff T. et al. (2020). Learning Mesh-Based Simulation with Graph Networks.

Sanchez-Gonzalez A. et al. (2020). Learning to Simulate Complex Physics with Graph Networks.