Part I: "In-silico microscopy" methods for large-scale LAMMPS simulations

Part II: OVITO 3.0 - A powerful data analysis and visualization tool for LAMMPS users

Dr. Alexander Stukowski

Darmstadt University of Technology, Germany
Materials and Earth Sciences

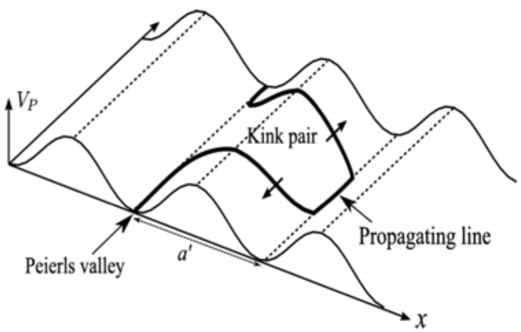
Studying crystal plasticity with MD

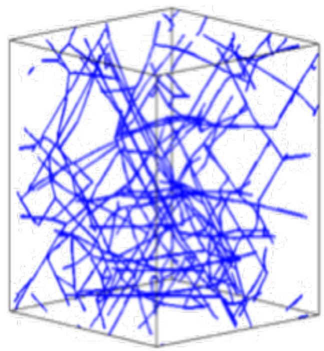
Zepeda-Ruiz et al., *Nature* 550 (2017), 492

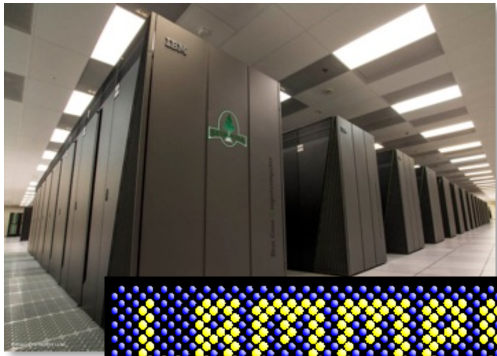
Dislocations: a multi-scale modelling problem

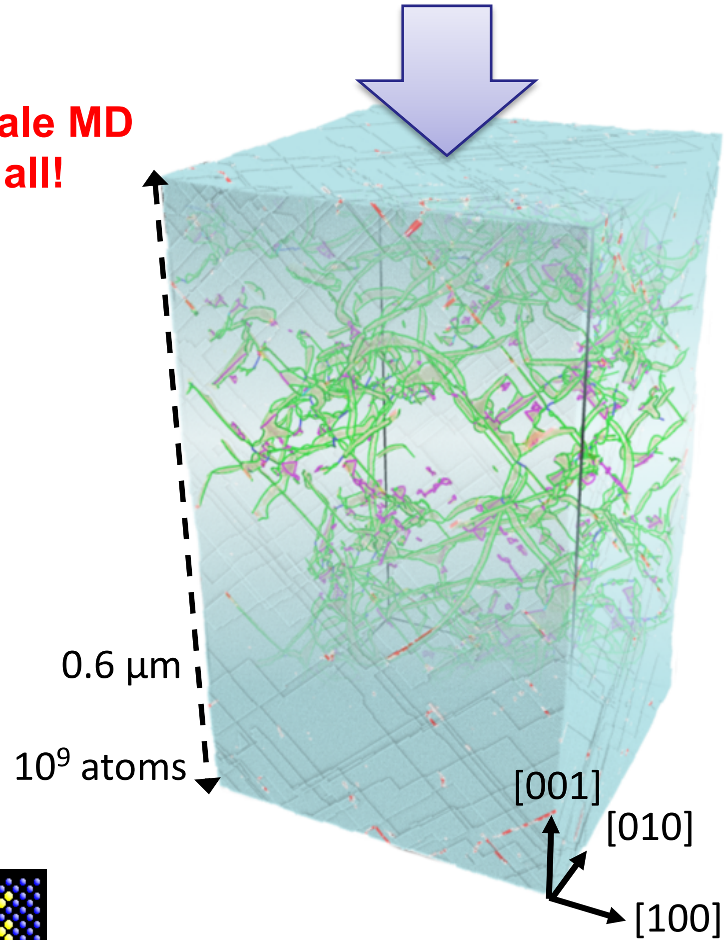
- Long-range interactions, collective behavior
- "Core effects" on the atomic scale

Large-scale MD includes all!



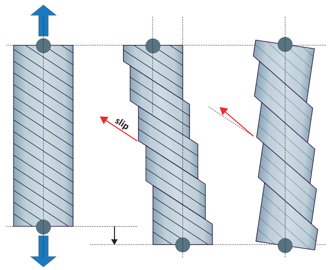






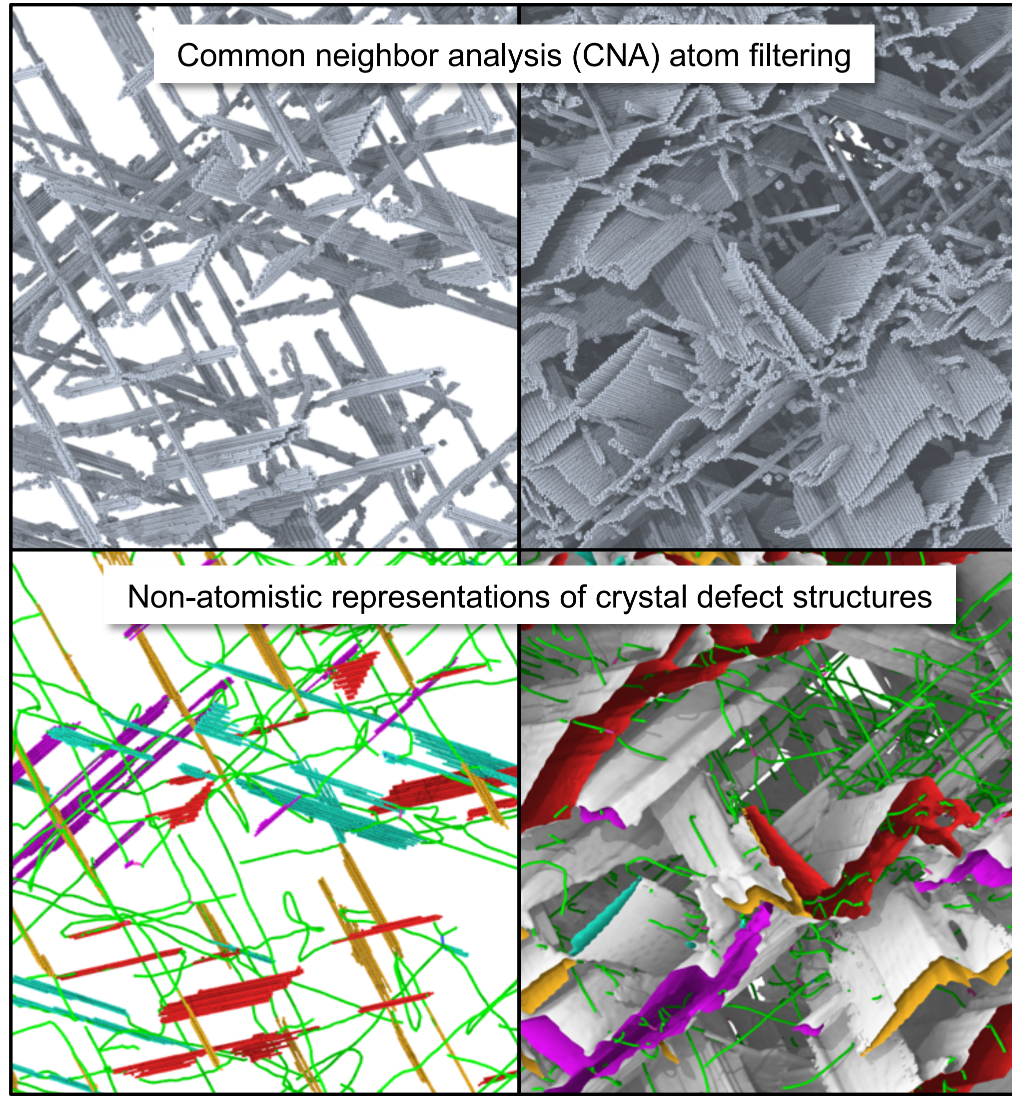
Our analysis needs

- Detection and visualization of crystal twinning



- Tracking crystal rotation
- Identifying dislocation lines / measuring defect densities
- Tracking dislocation motion

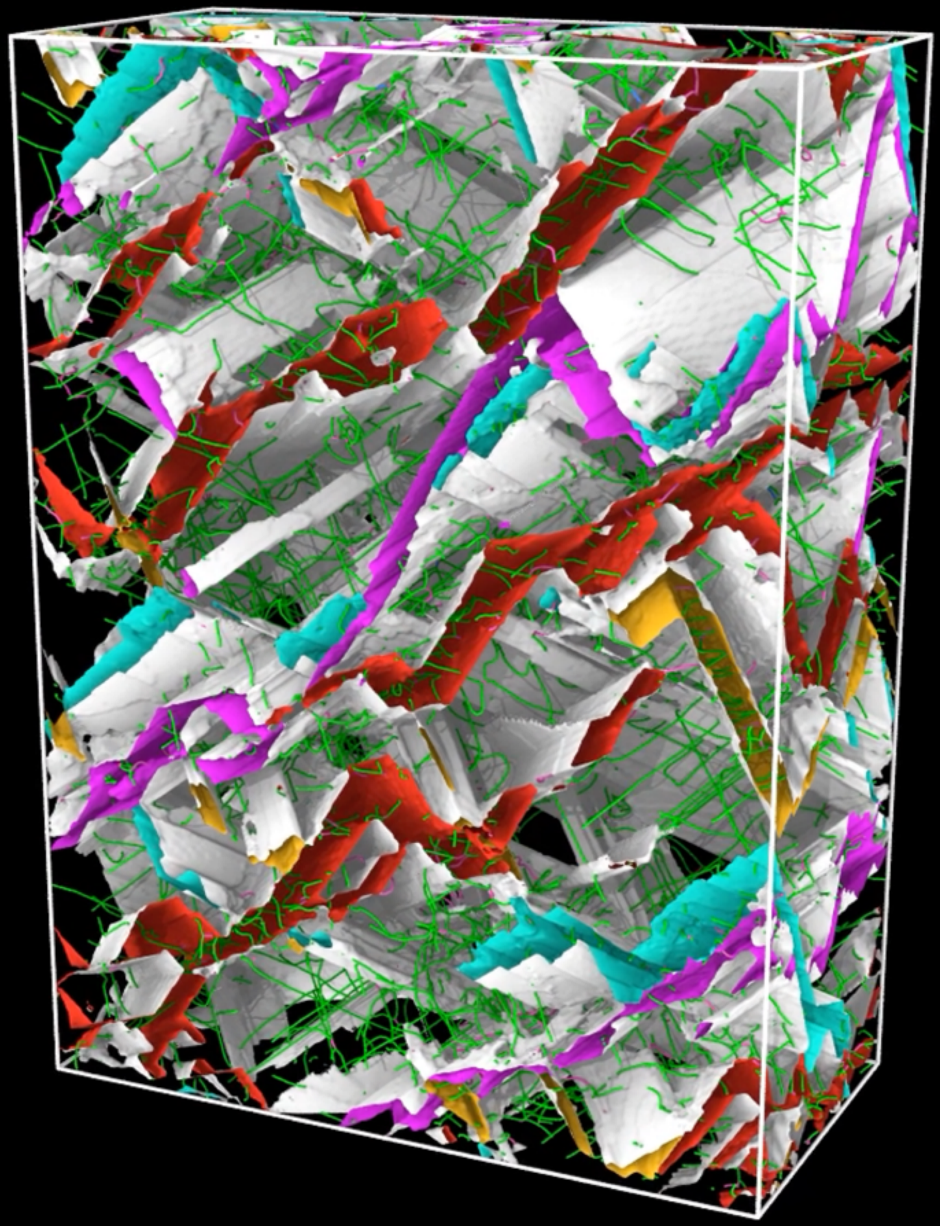
Ideally do these on the fly!



Zepeda-Ruiz et al., *Nature* 550 (2017), 492

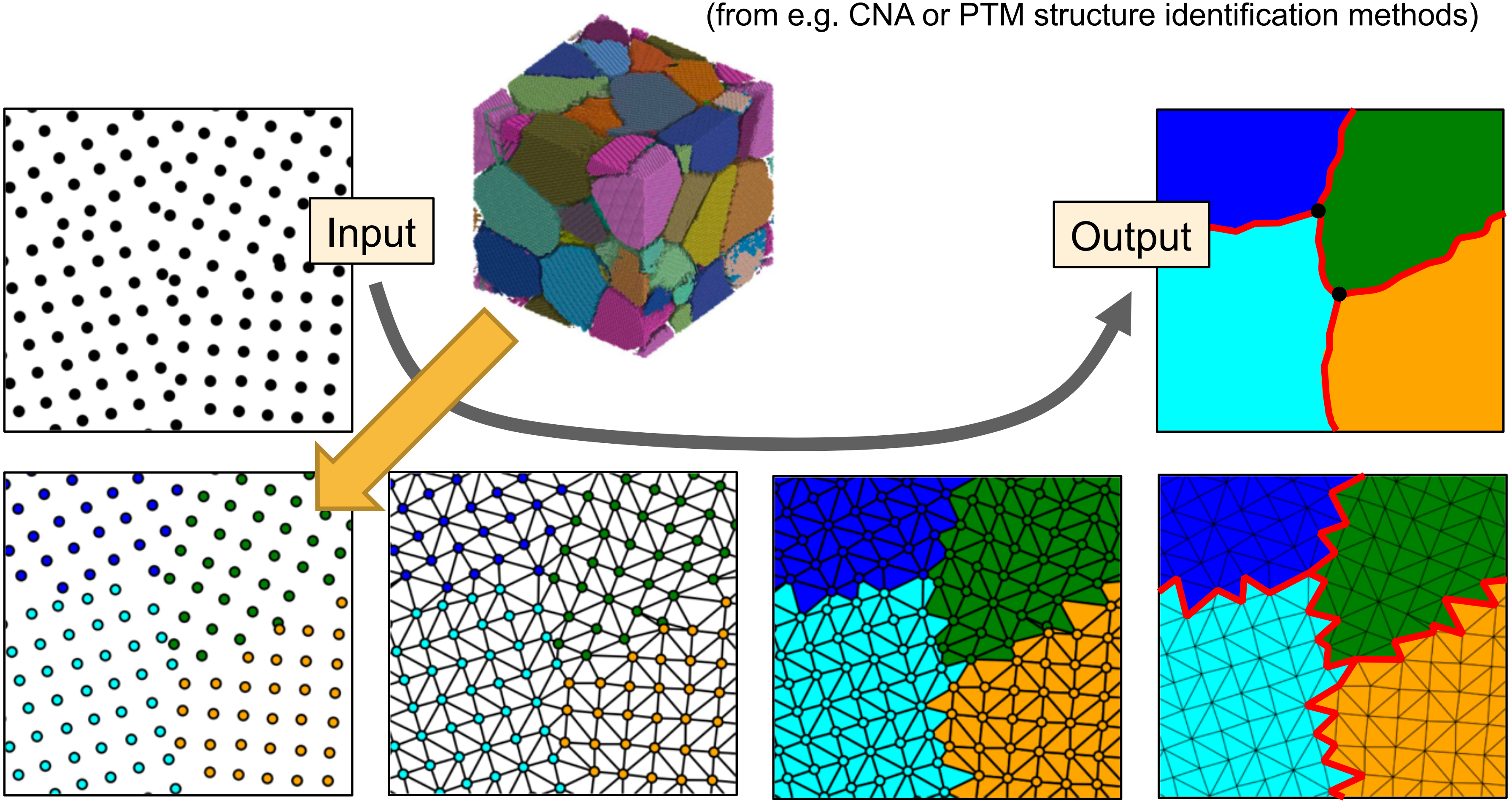
MD simulation of Ta crystal being strained at a supercritical rate

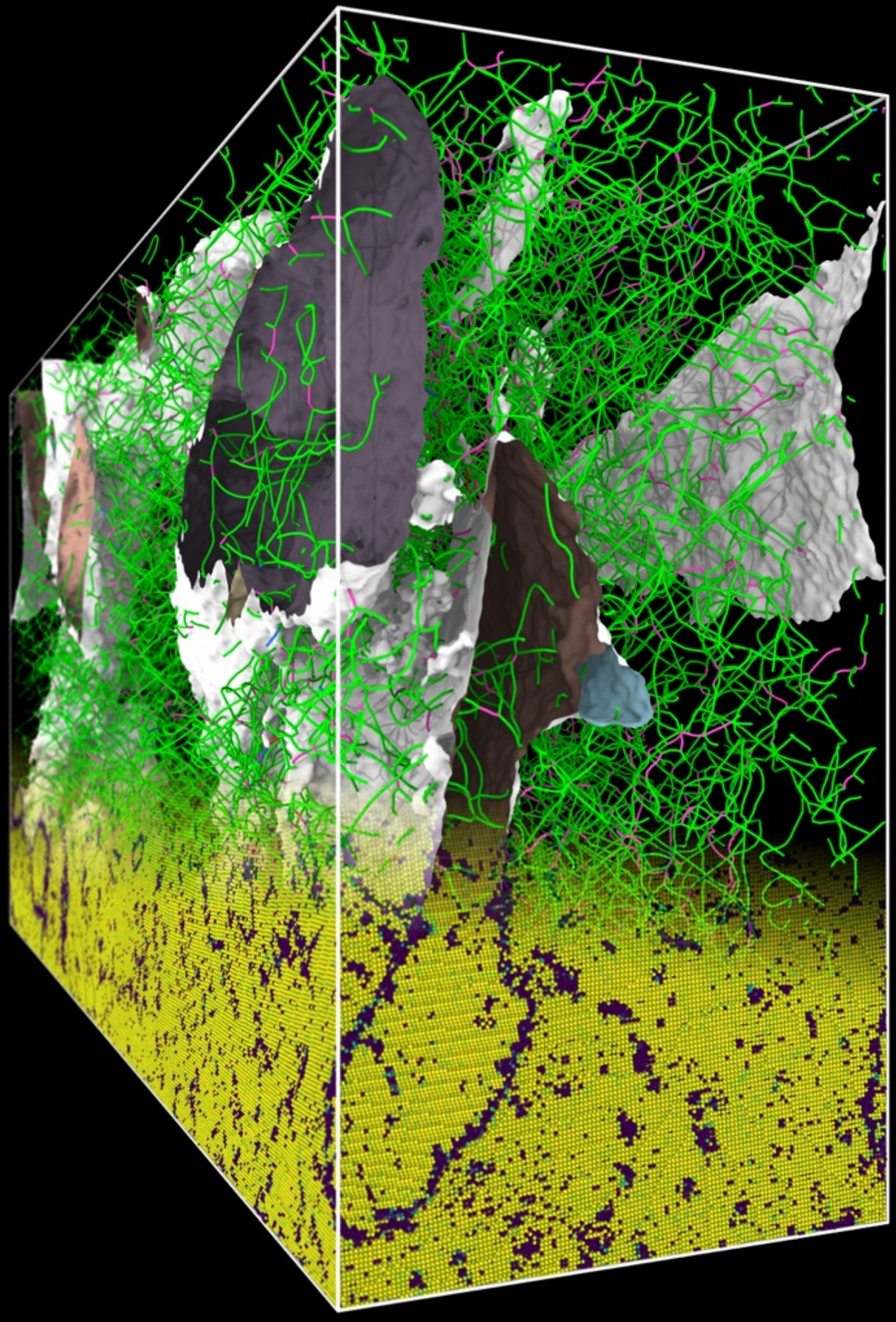
~33M atoms



Grain segmentation algorithm

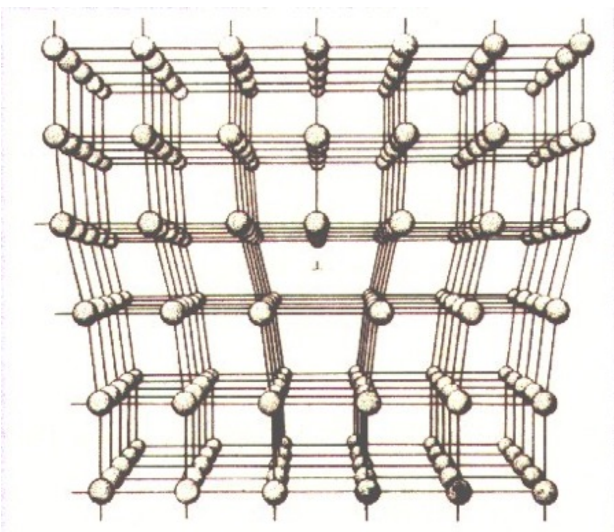
Atomic-level lattice orientation
(from e.g. CNA or PTM structure identification methods)

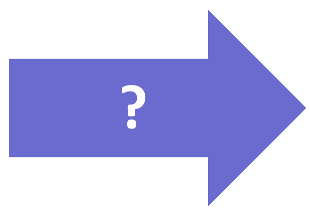




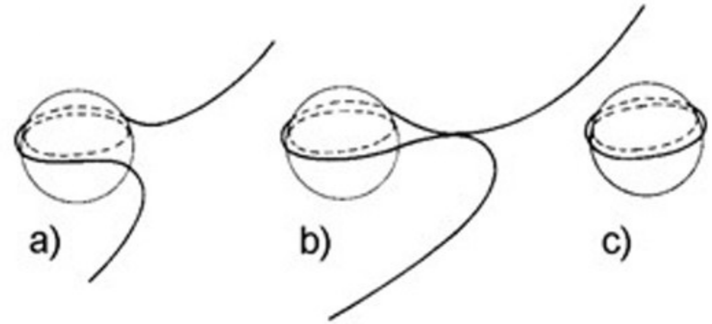
Dislocation defects

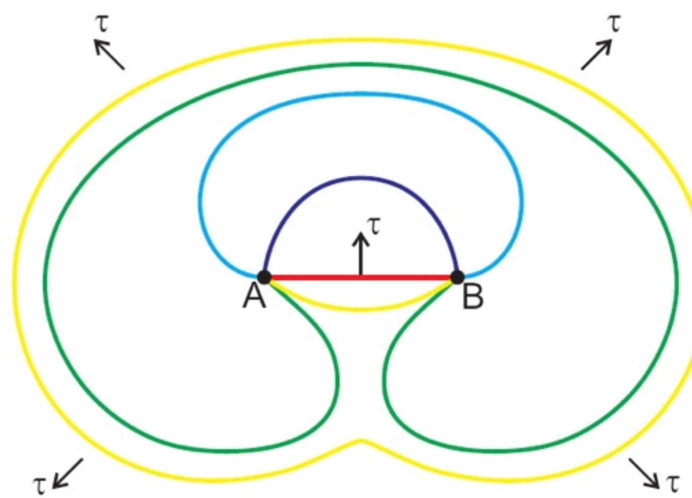
Atomistic picture of dislocations





Discrete dislocation line theory

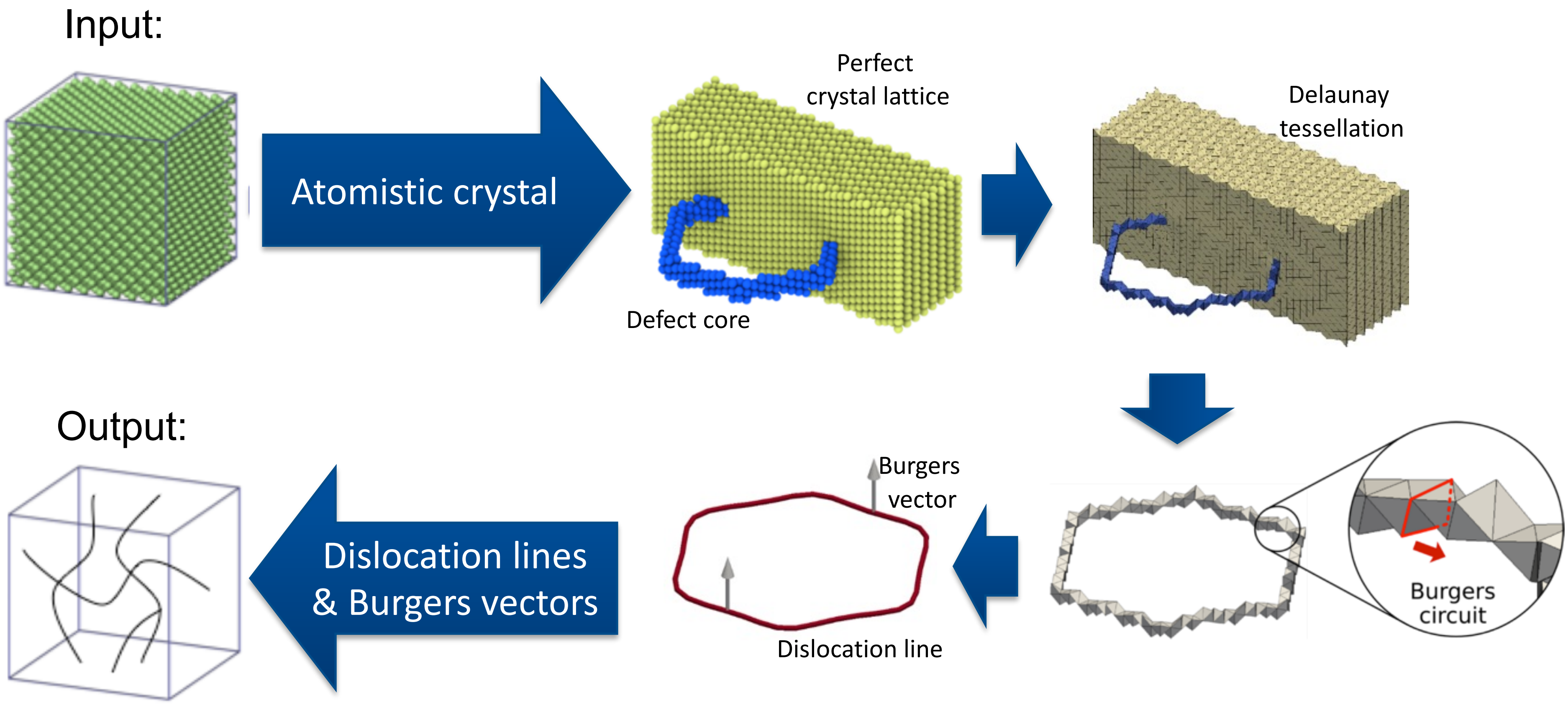




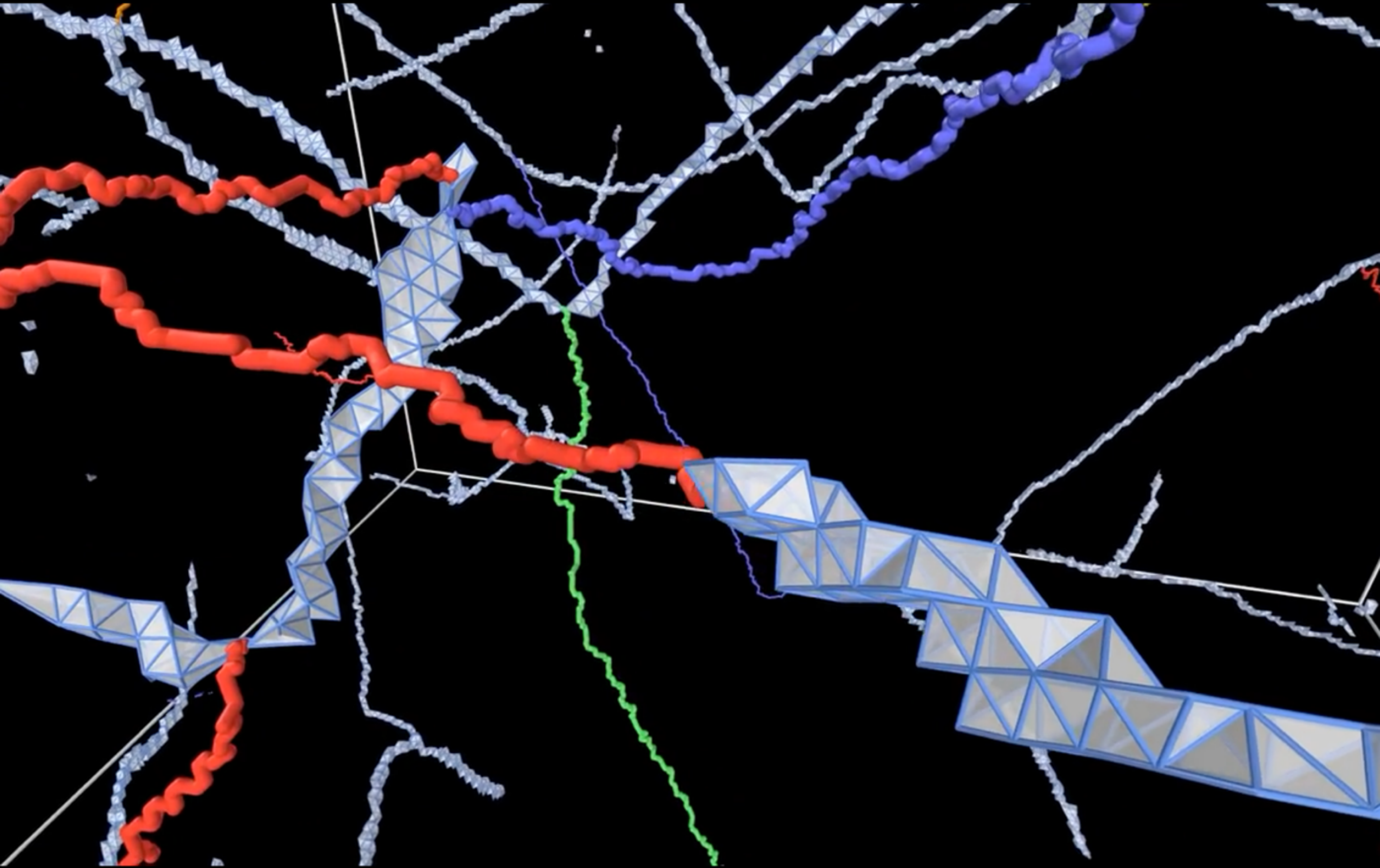
How to measure dislocation content in MD simulations?

$$\rho = \frac{1}{V}\int dl$$

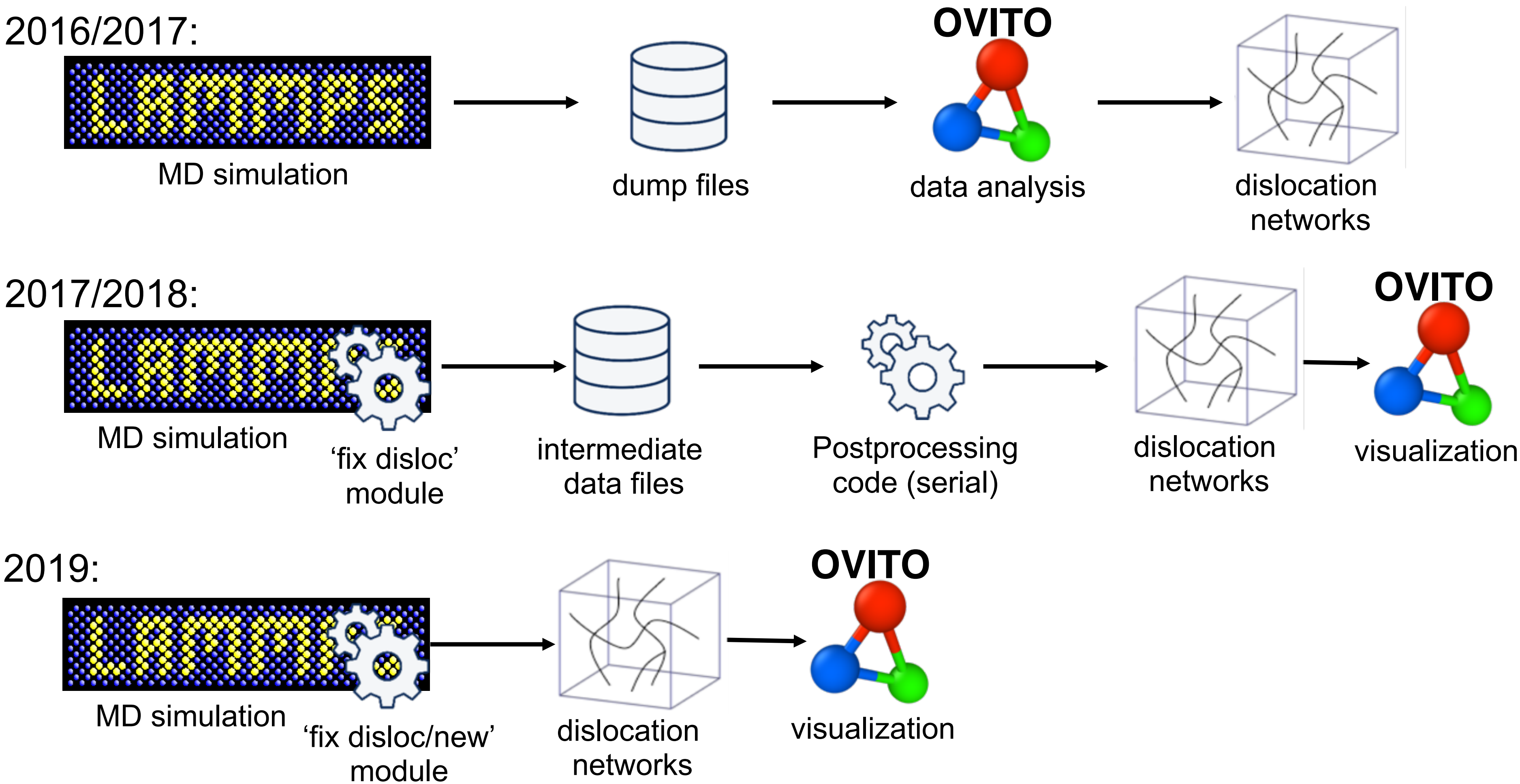
Dislocation Extraction Algorithm (DXA) in OVITO



2012 Stukowski et al. MSMSE 20, 085007

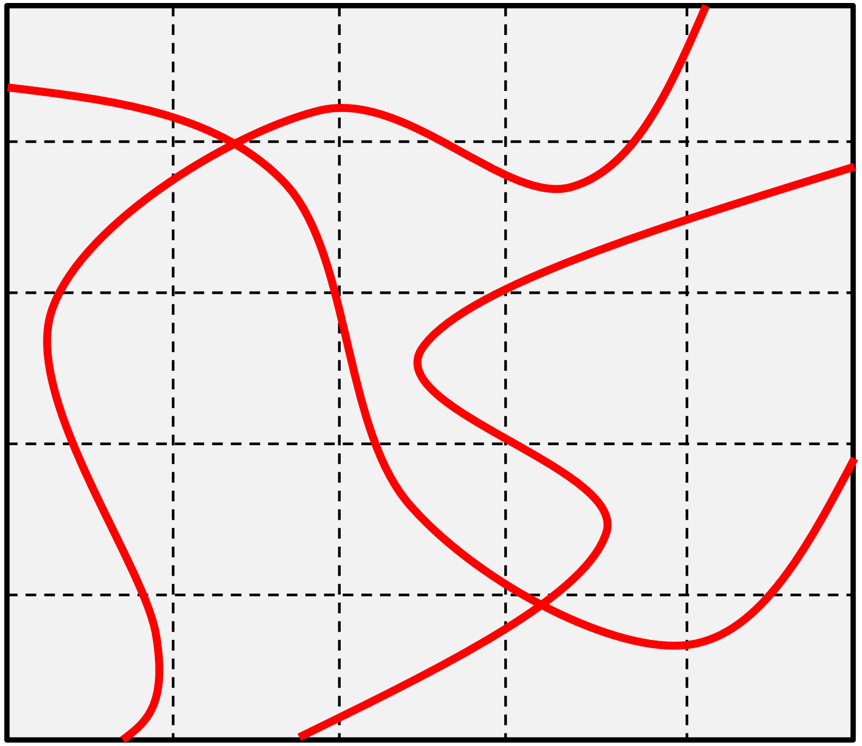


Data analysis workflows

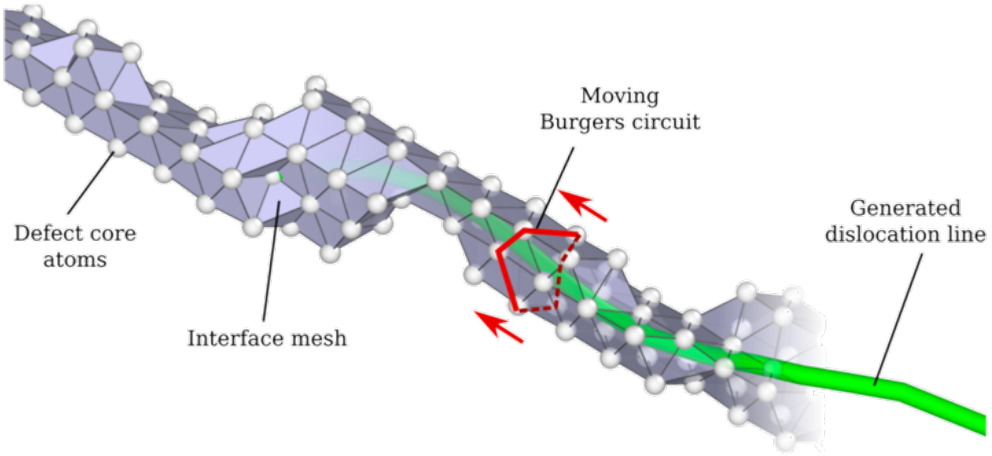


How to parallelize the identification of extended structures?

MD spatial decomposition scheme:

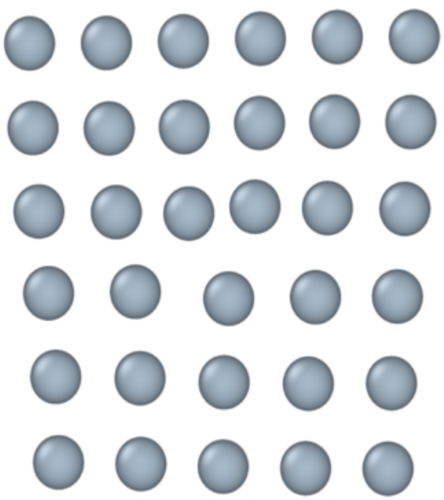


DXA line sweeping process:

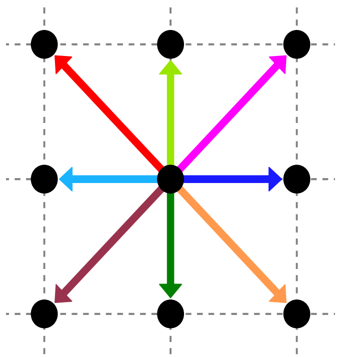


Dislocations as incompatibilities in a discretized elastic field

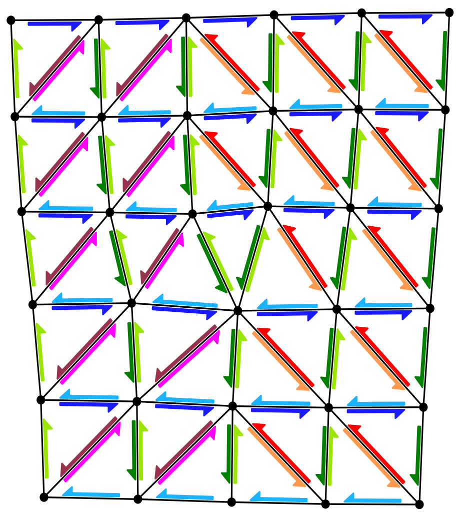
Input crystal



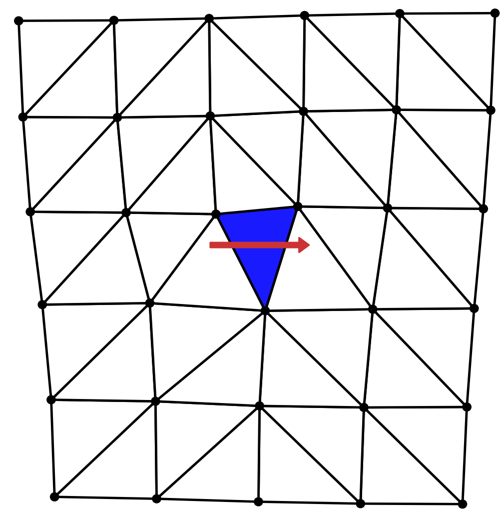
Set of reference vectors



Delaunay triangulation



Dislocated cells



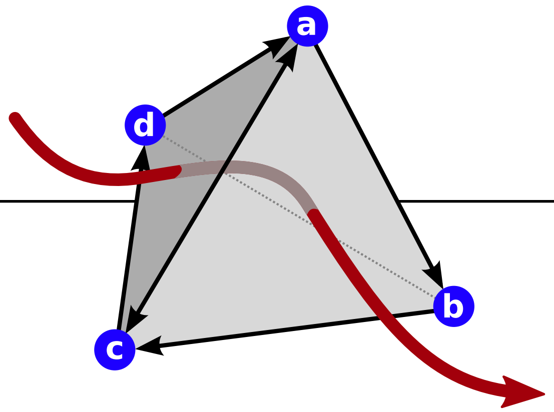
Algorithm:

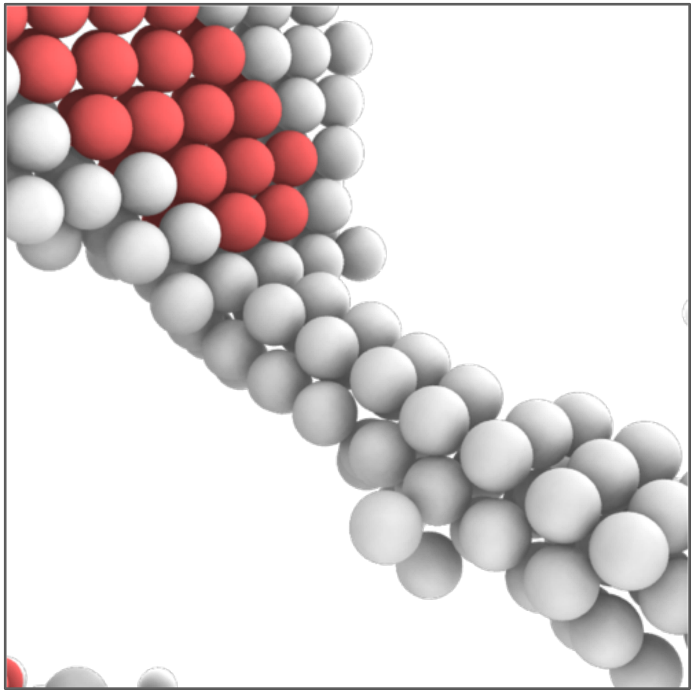
1. Compute Delaunay triangulation.
2. Assign an ideal reference vector $\boldsymbol{L}_{ab}$ to each edge:

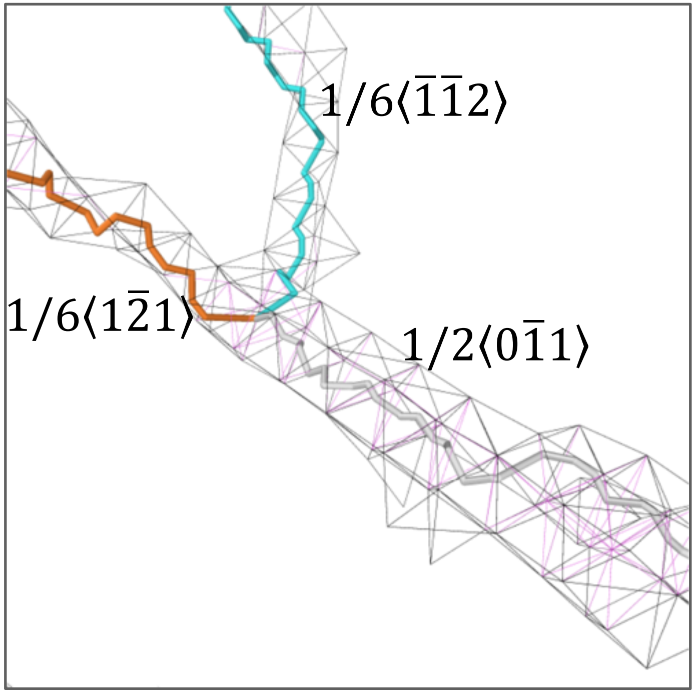
$$\boldsymbol{x}_{ab} = \boldsymbol{x}_b - \boldsymbol{x}_a \qquad \boldsymbol{L}_{ab} = \min \arg_{\boldsymbol{L}_i \in \mathcal{L}} |\boldsymbol{x}_{ab} - \boldsymbol{L}_i|$$

A. Stukowski.
JMPS 70 (2014), 314

Extraction of dislocation segments







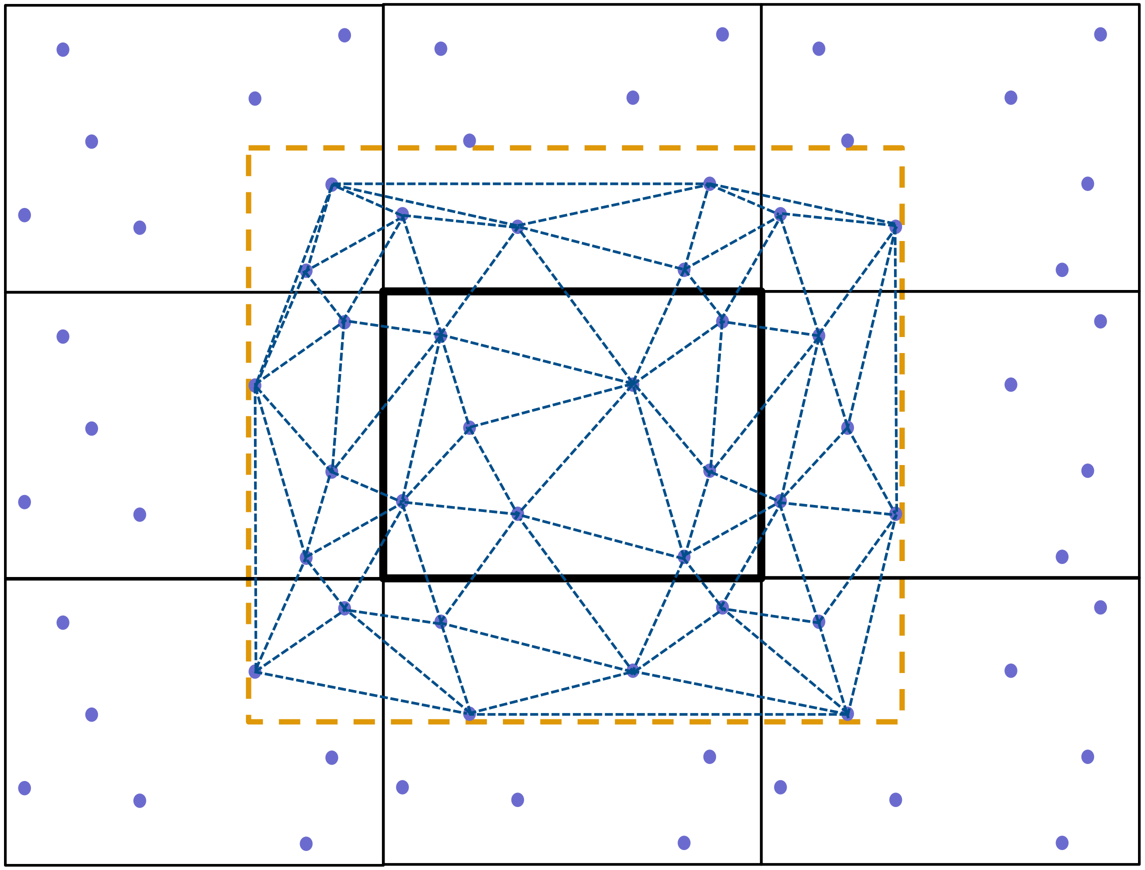
Delaunay tessellation with PBCs/domain decomposition

Ghost atom layer must be wide enough to guarantee a consistent Delaunay topology in the overlap region:

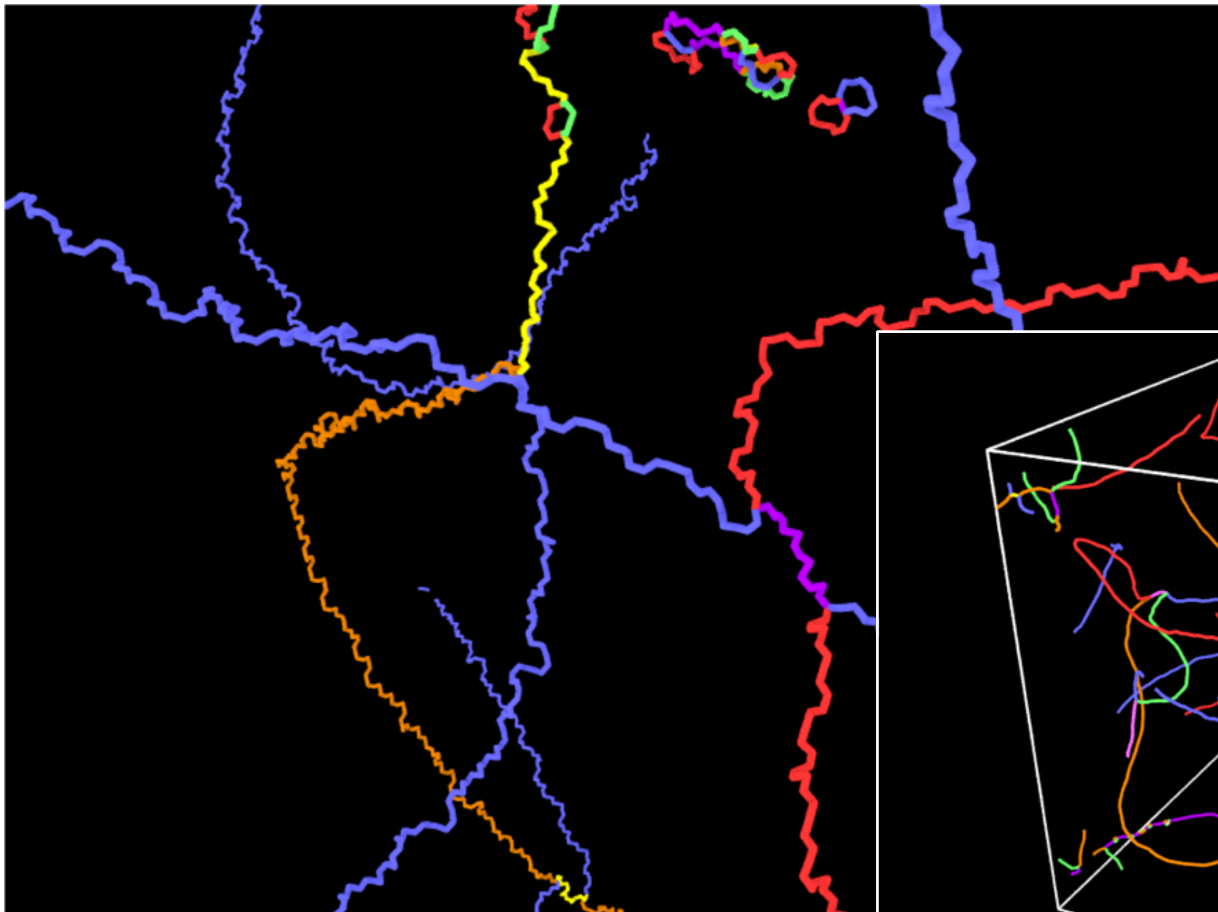
comm_modify cutoff 12.0

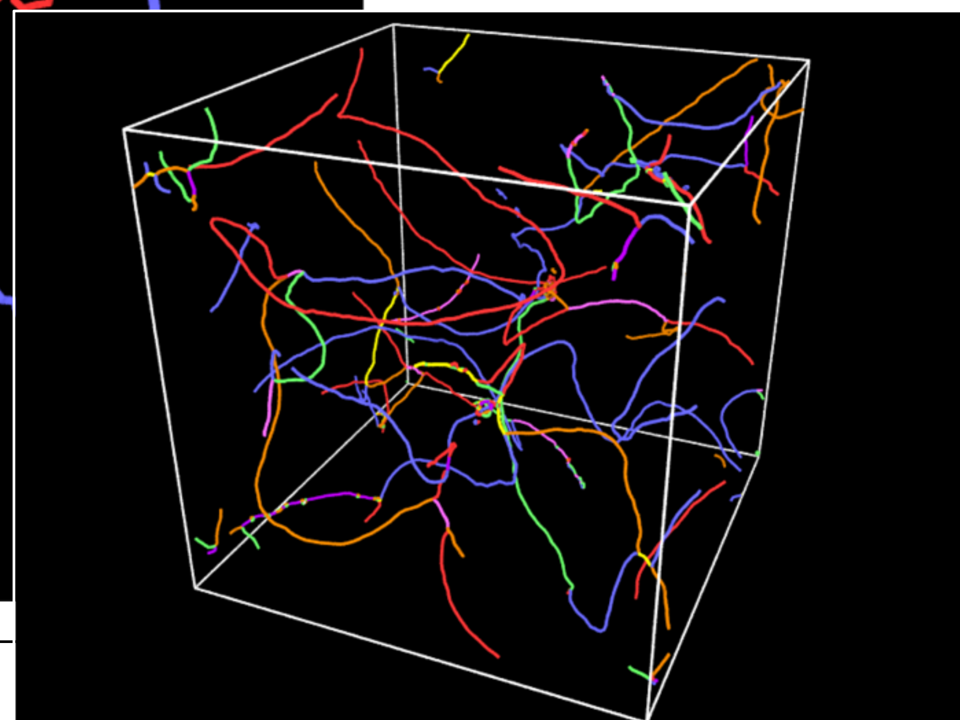
Delaunay triangulation:

Geogram library by INRIA (France)



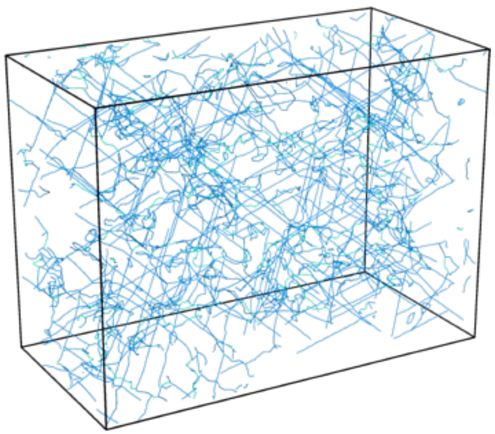
Parallelized dislocation detection algorithm

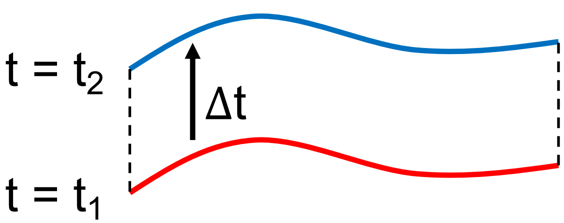




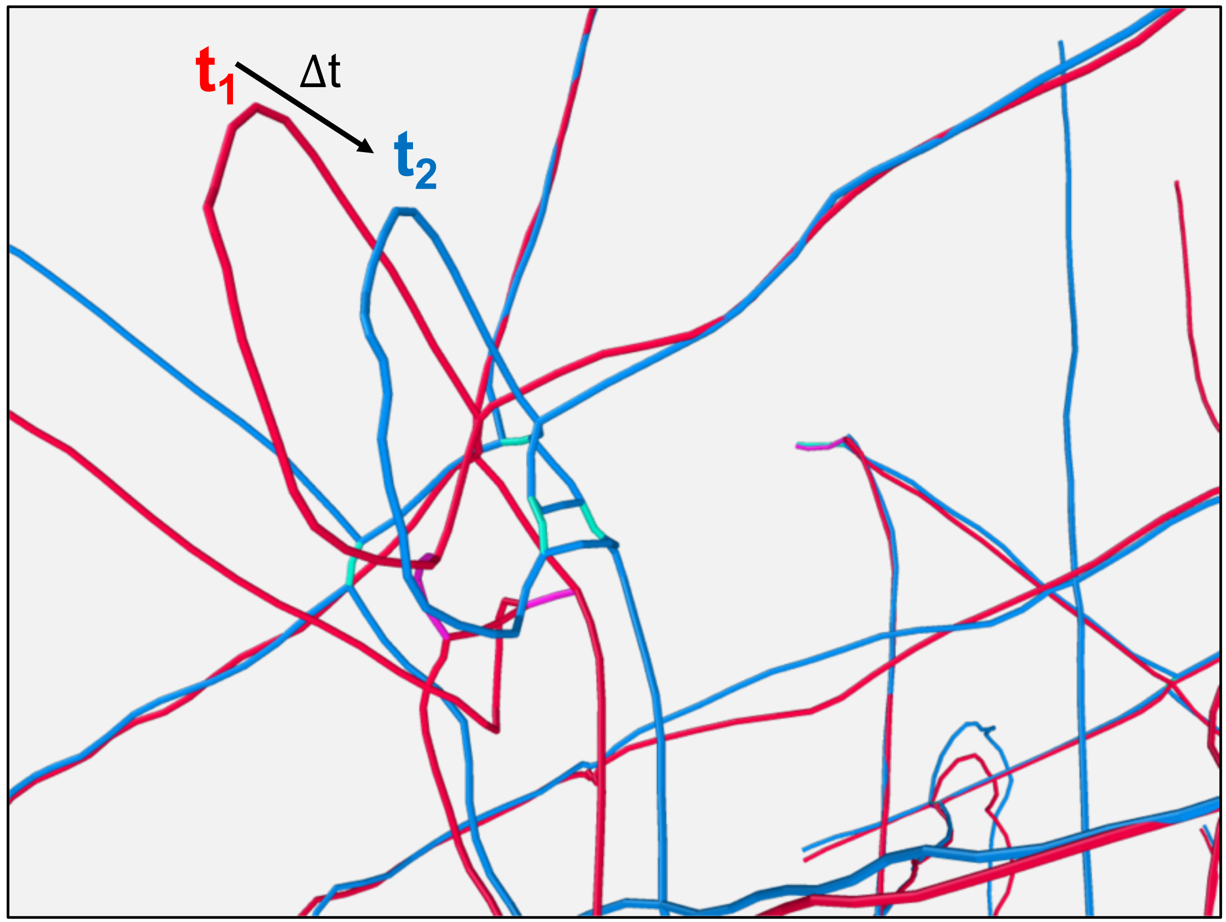
- Implementation currently limited to fully dense and periodic single crystals

Incremental analysis: How do dislocations move?

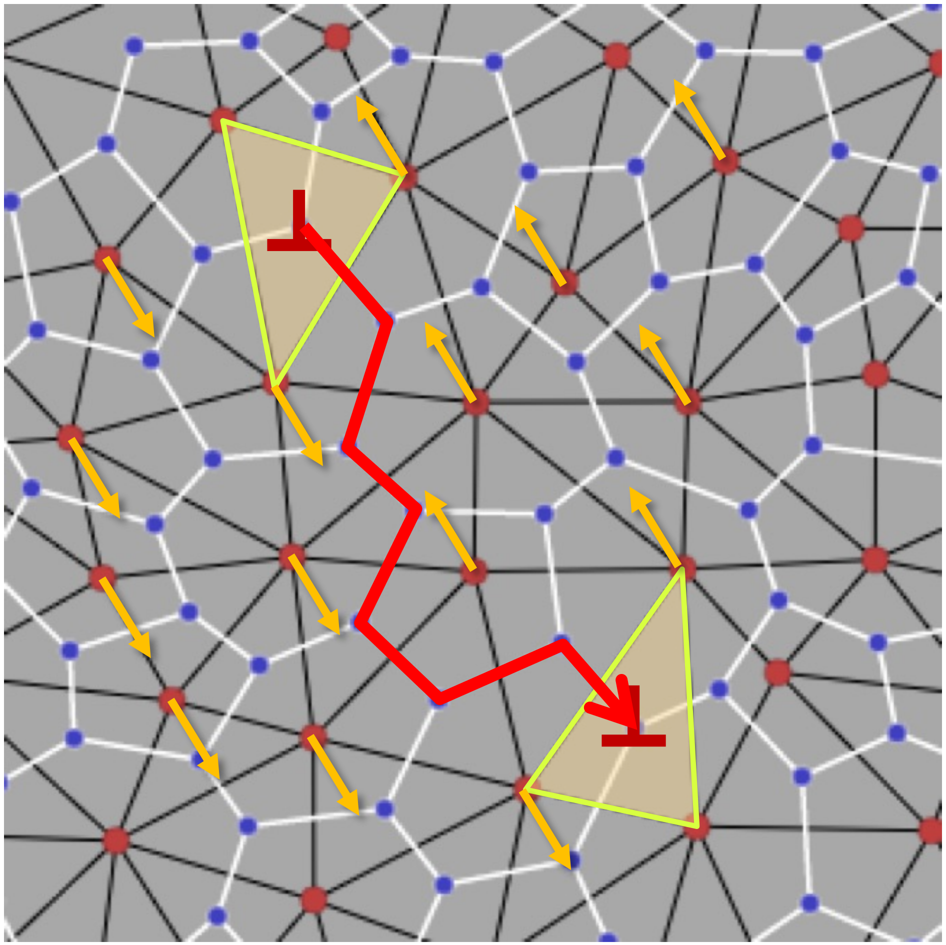


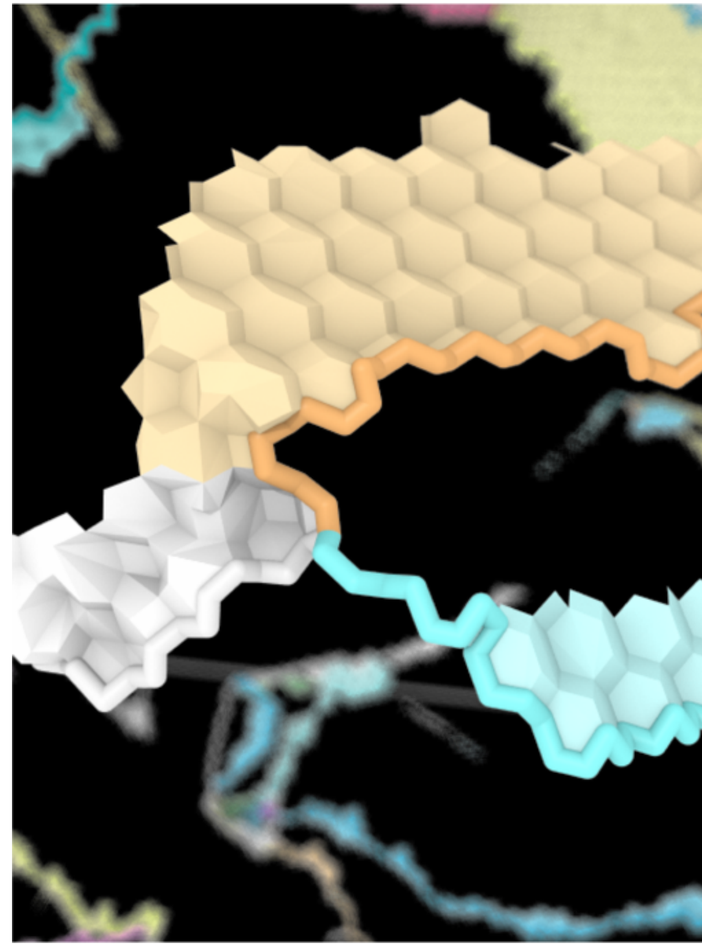


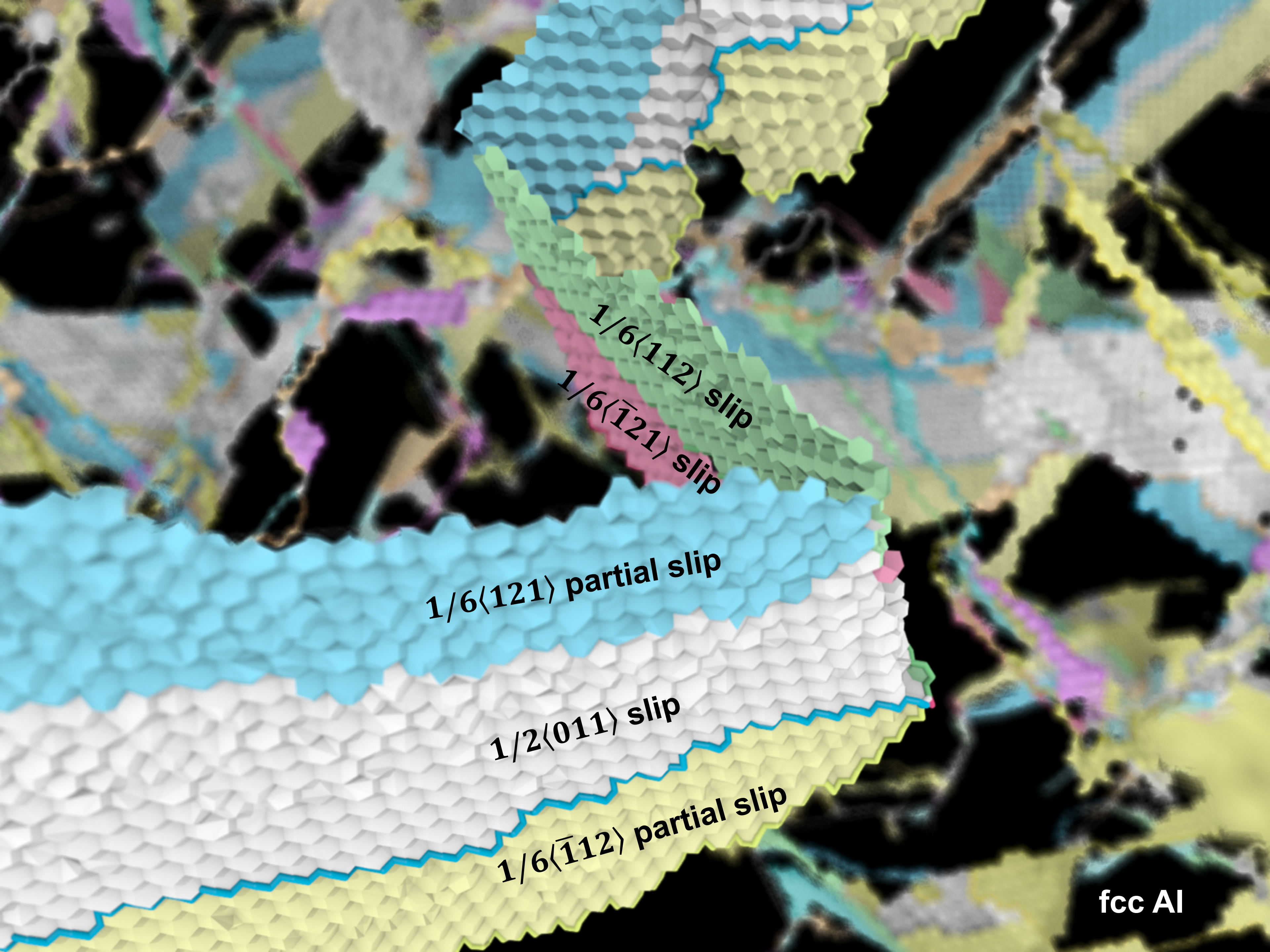
$$\Delta\boldsymbol{\varepsilon}^{\mathrm{p}} = \sum_i \frac{\Delta A}{2V}\left(\mathbf{n}_i \otimes \mathbf{b}_i + \mathbf{b}_i \otimes \mathbf{n}_i\right)$$



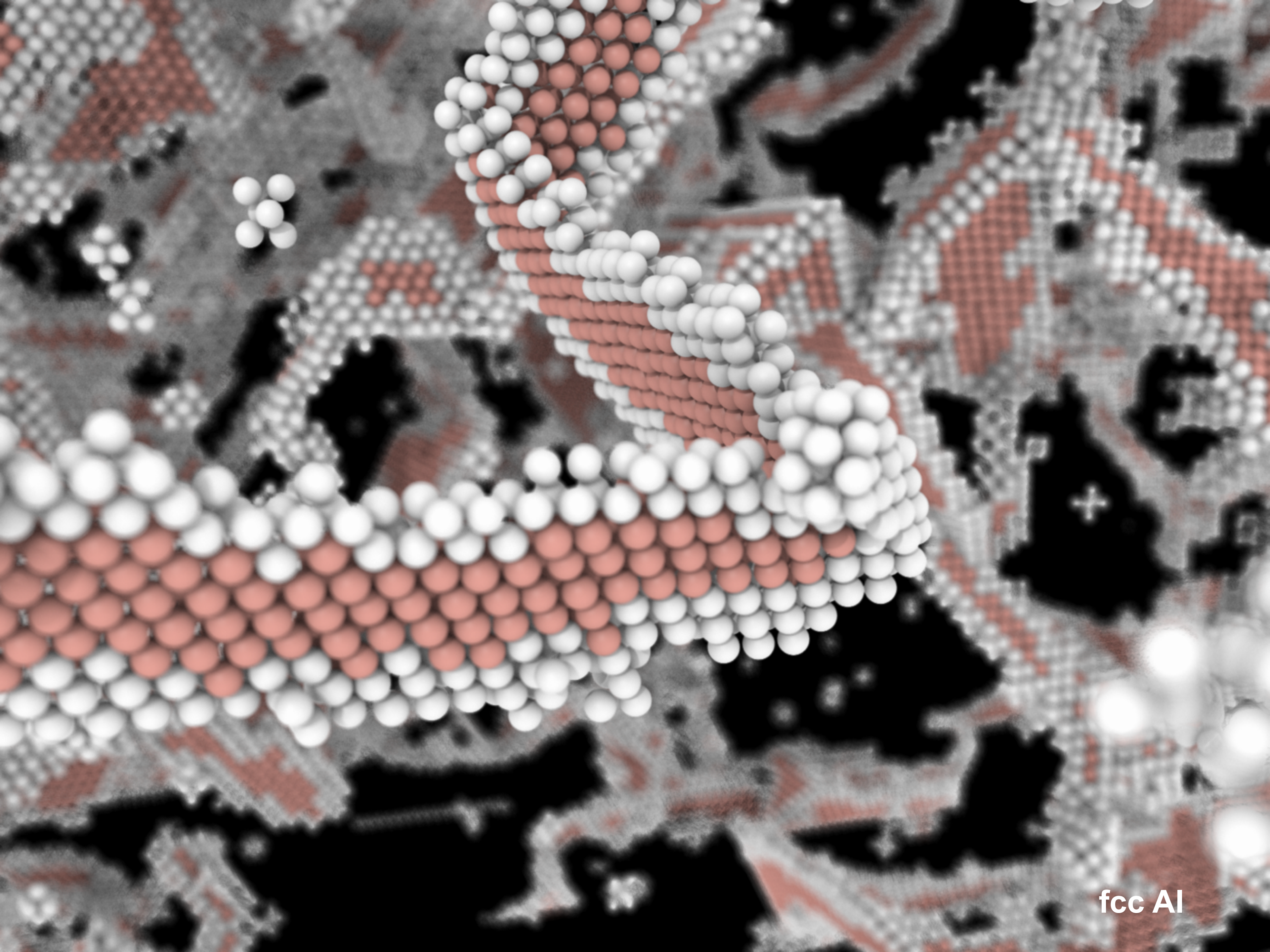
Reconstructing slip surfaces from MD trajectories

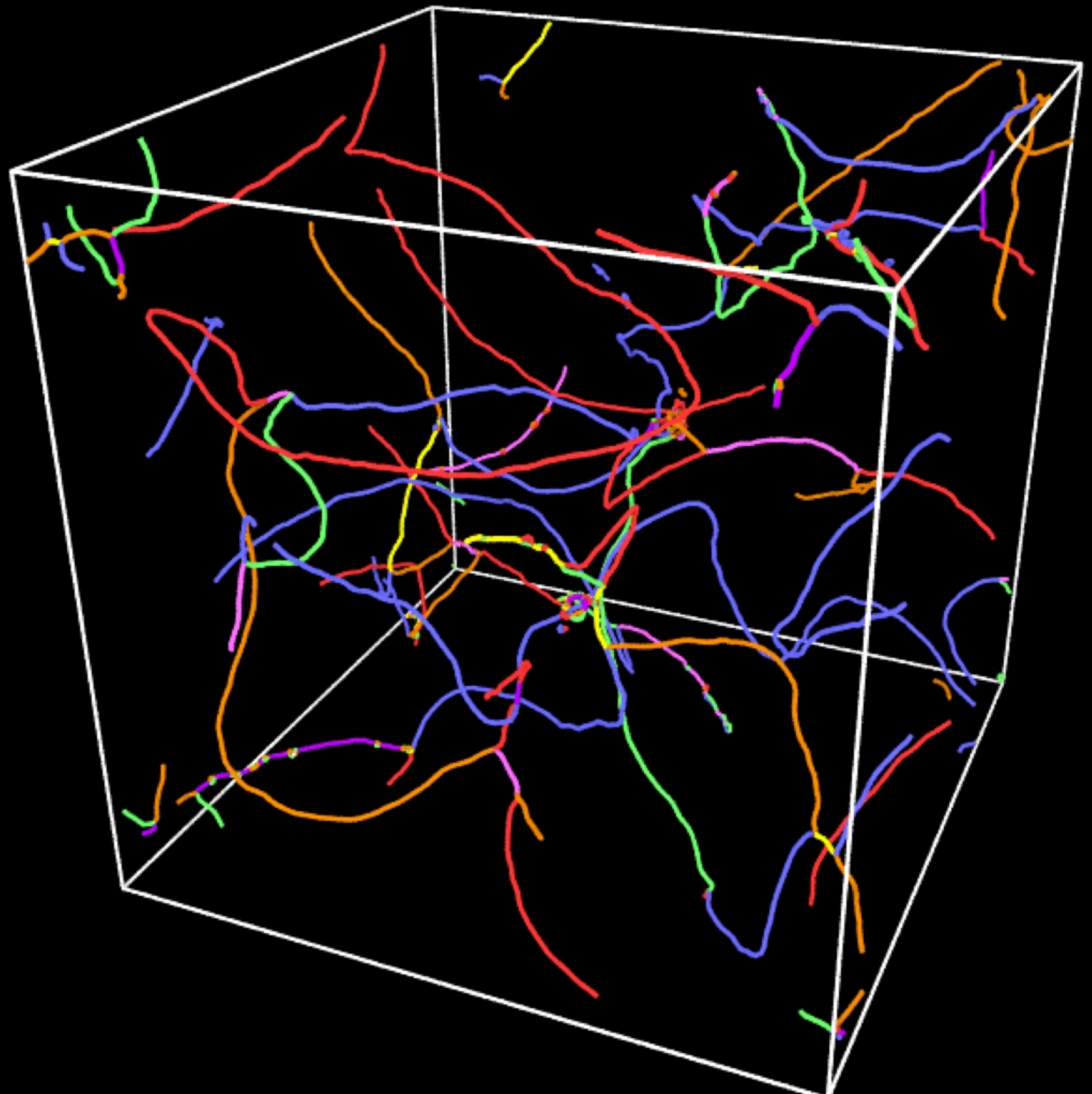






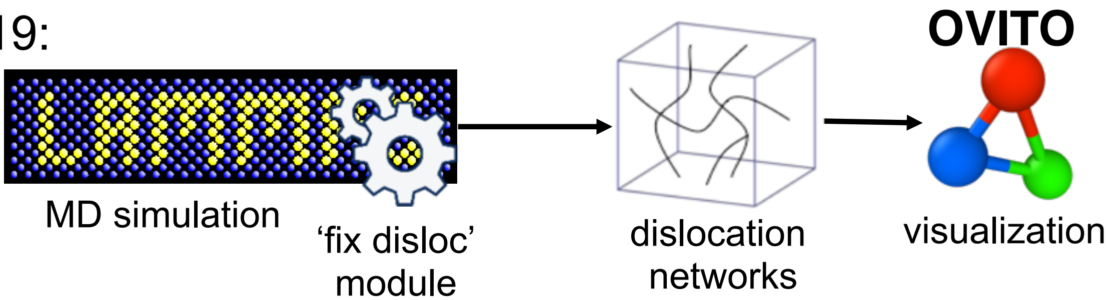
1/6⟨112⟩ slip
1/6⟨$\bar{1}$21⟩ slip
1/6⟨121⟩ partial slip
1/2⟨011⟩ slip
1/6⟨$\bar{1}$12⟩ partial slip
fcc Al


fcc Al

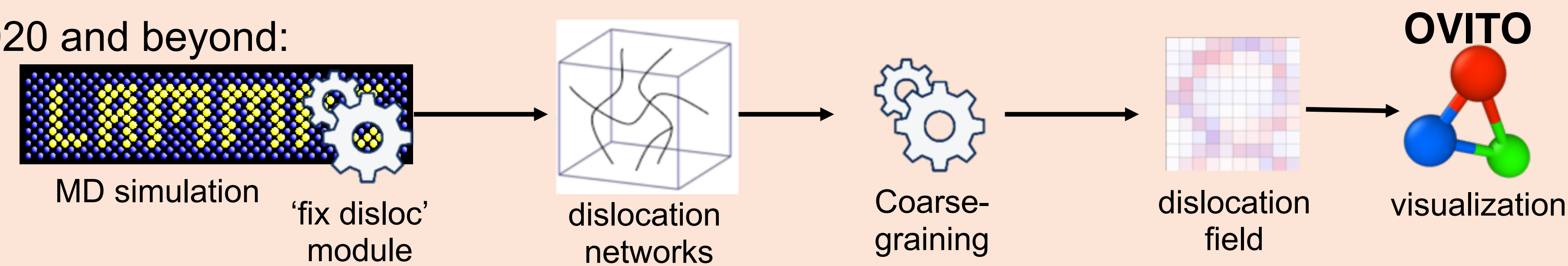


Data analysis workflows

2019:



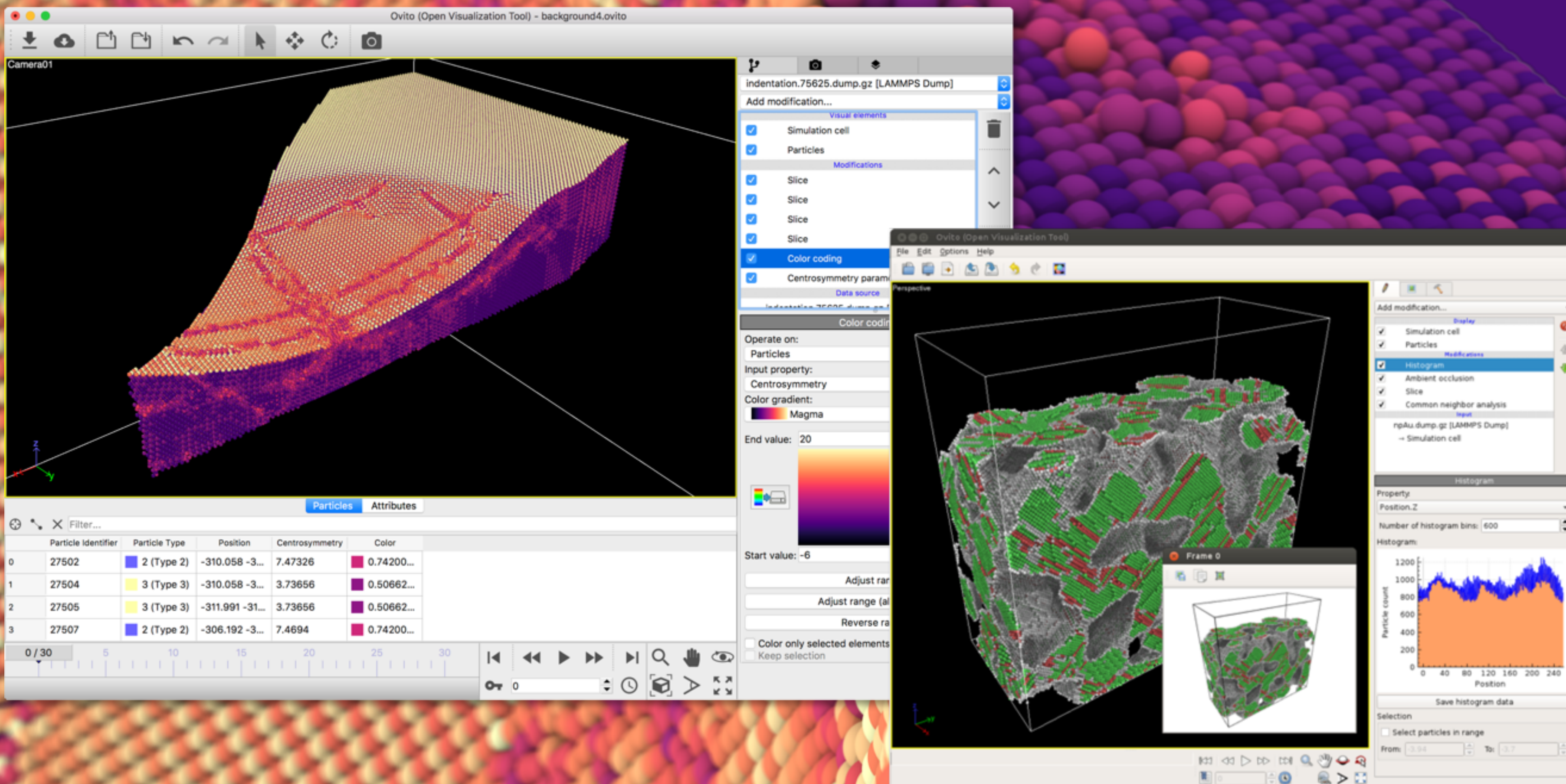
2020 and beyond:



OVITO

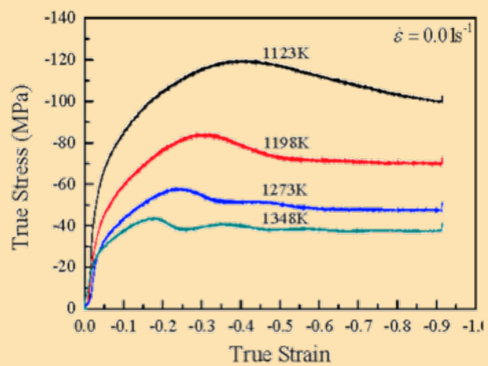
www.ovito.org

Scientific data analysis and visualization
software for materials simulations

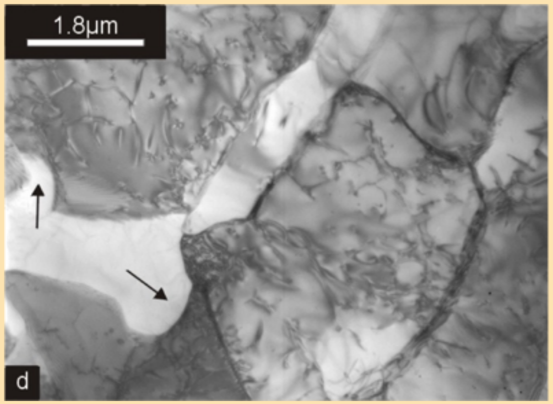


Data visualization and analysis – From MD simulations to insights

Experiment



Microscopy & Imaging

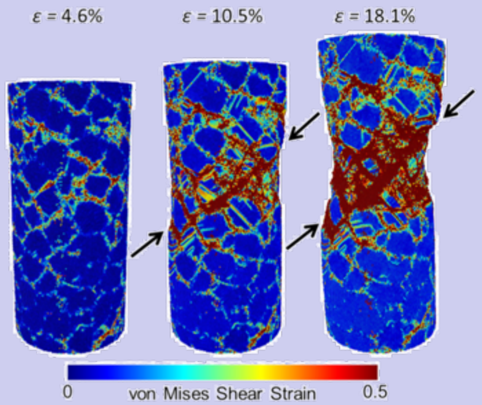


Simulation

Raw output data

Data analysis & Visualization

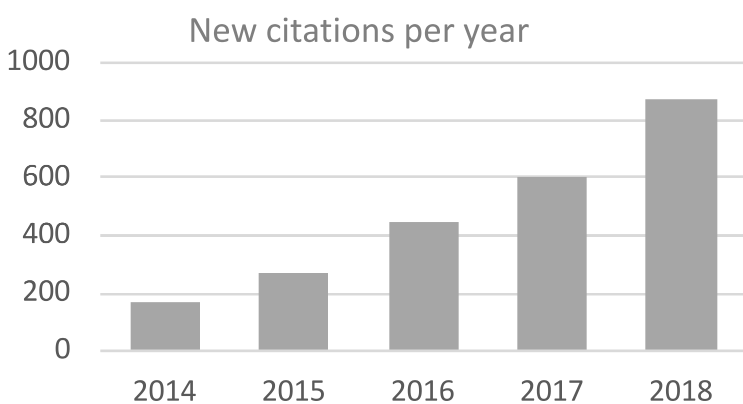
“in-silico microscopy“



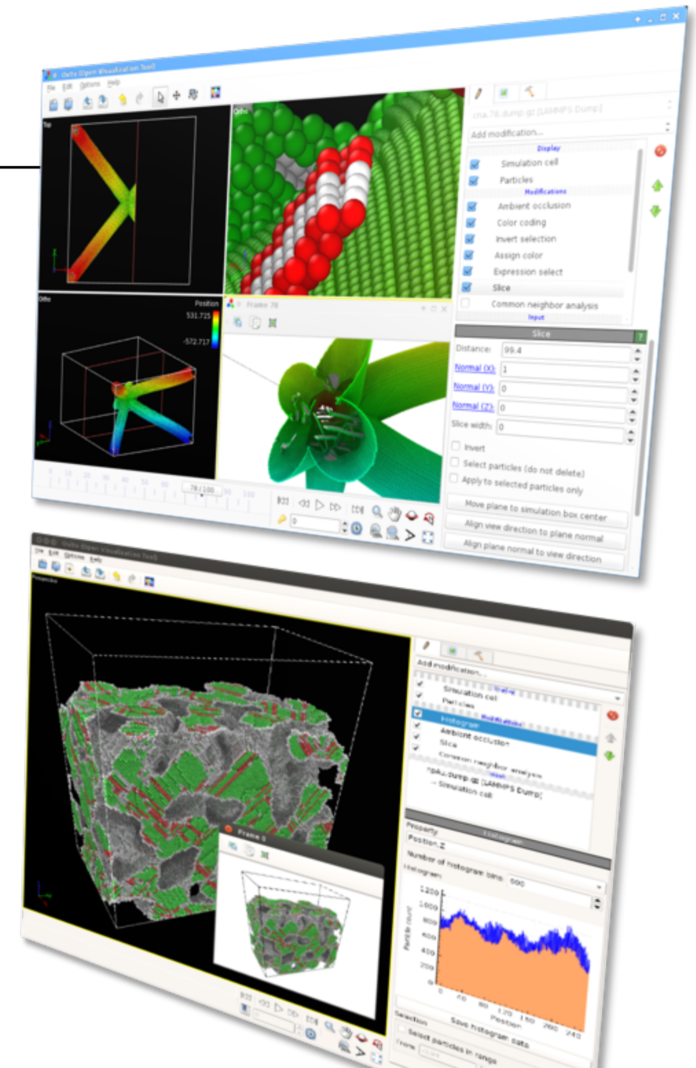
OVITO Software Package

OVITO is among the most widely-used data analysis and visualization solutions for atomistic simulations

- **3200+ scientific publications using OVITO**
- **3+ new publications per day**

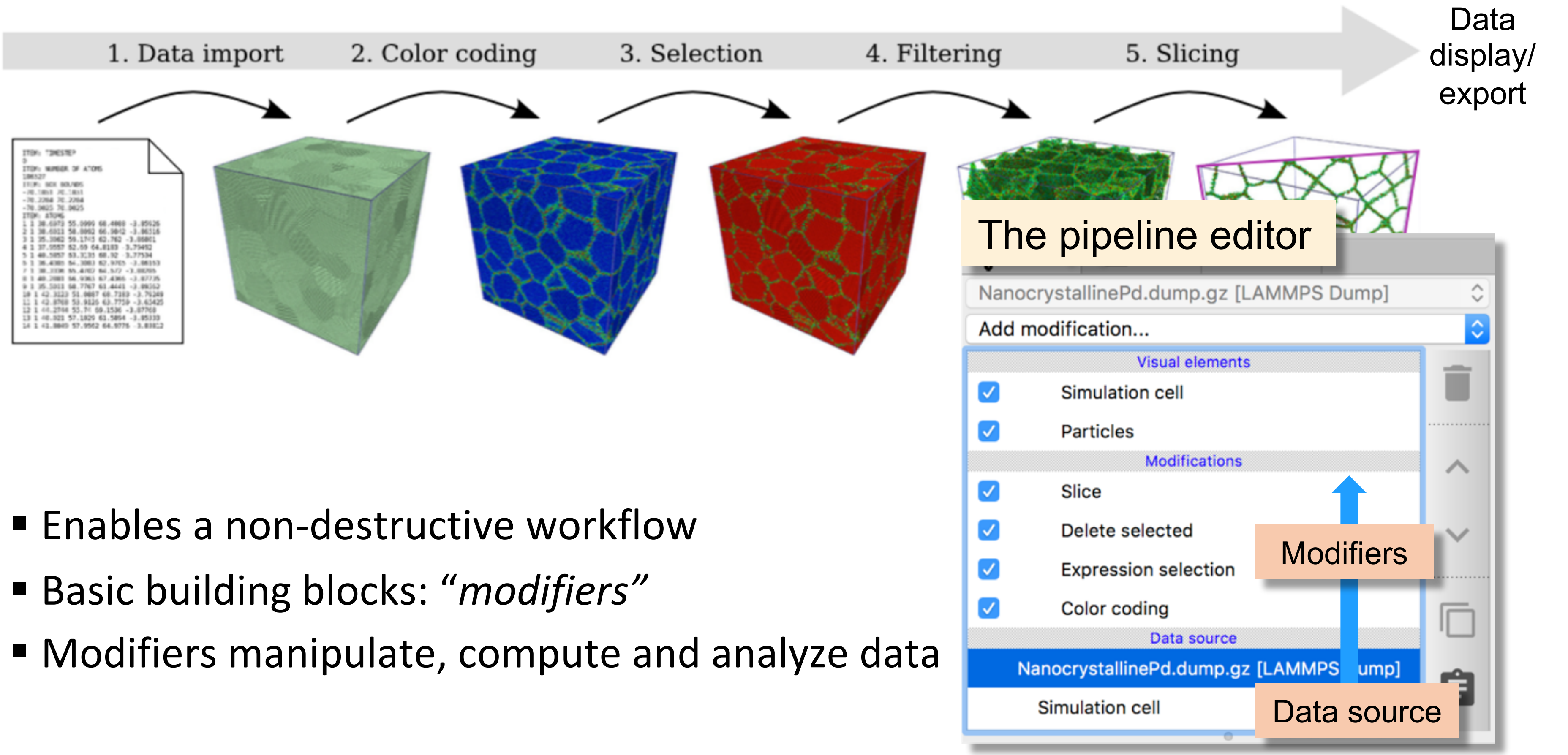


Source:
Google Scholar



A. Stukowski. Modelling Simul. Mater. Sci. Eng. 18 (2010), 015012

Processing pipeline

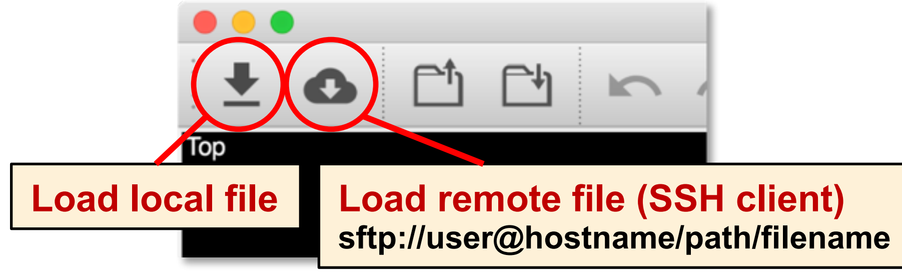


- Enables a non-destructive workflow
- Basic building blocks: *"modifiers"*
- Modifiers manipulate, compute and analyze data

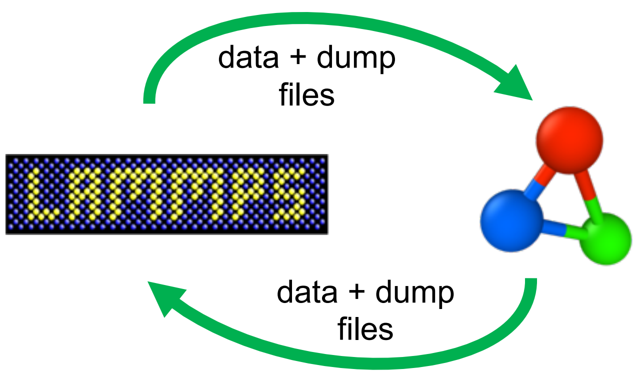
Simulation data input/output

- Automatic detection of file sequences:

```
simc_0.dump
simc_100.dump
simc_200.dump
simc_300.dump
…
→ simc_*.dump
```



- Support for .gz compressed files
- Built-in SSH/SCP client for reading remote files



- More input formats:
  - XYZ (basic and extended variants)
  - NetCDF (written with '*dump netcdf*' command)
  - CFG, GSD/HOOMD, IMD, PDB, GALAMOST, DL_POLY
  - **Ab initio codes:** POSCAR, FHI-aims, QE, CASTEP
  - **Volumetric:** XSF, Cube, CHGCAR
  - **Geometry:** OBJ, STL, VTK

OVITO's data model

- Atomistic models: A set of *particle properties*, i.e. named data arrays:

	Particle 1	Particle 2	Particle 3	Particle 4	Particle 5	Particle N
Property A	*Value*	*Value*	*Value*	*Value*	*Value*	…
Property B	*Value*	*Value*	*Value*	*Value*	*Value*	…
Property …	…	…	…	…	…	…

- Number of particle properties is not limited, e.g.
  - Position
  - Type
  - Identifier
  - Velocity vector
  - Energy
  - Charge
  - Selection state
  - Color
  - ……

Data types:
Integer, real

Dimensionalities:
Scalars, vectors, tensors

- Property arrays are also used for storing elements other than particles, e.g.

bonds, surface meshes, voxel grids …

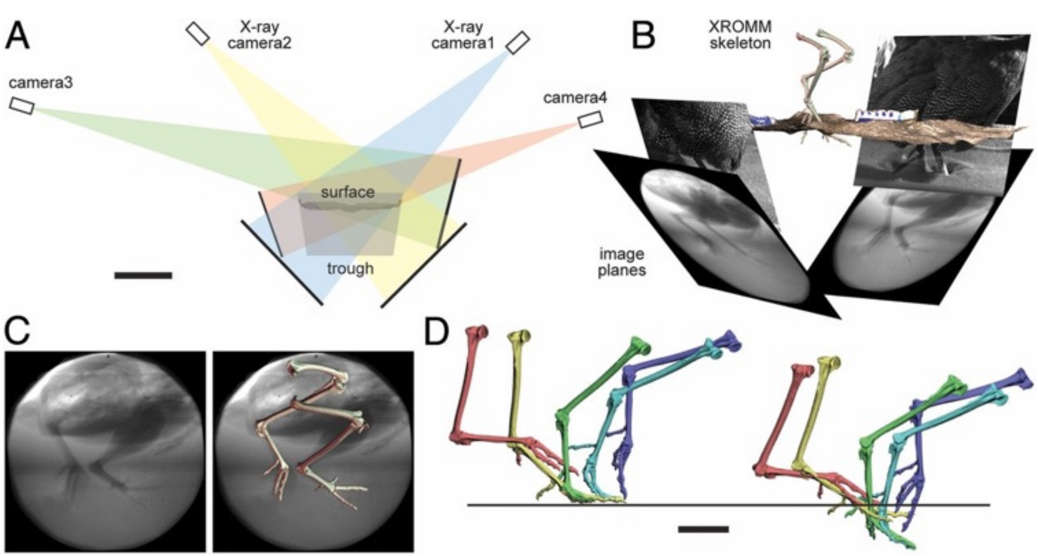
Other types of particle simulations

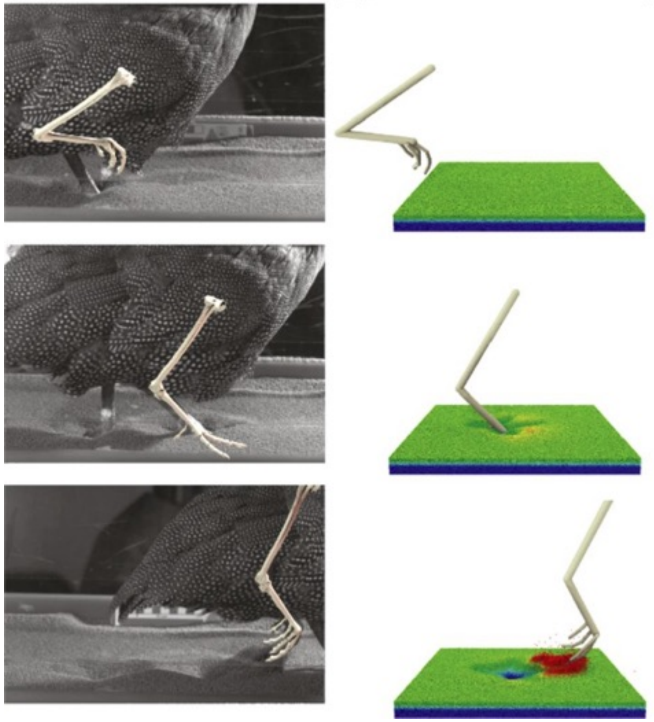
The birth of a dinosaur footprint: Subsurface 3D motion reconstruction and discrete element simulation reveal track ontogeny

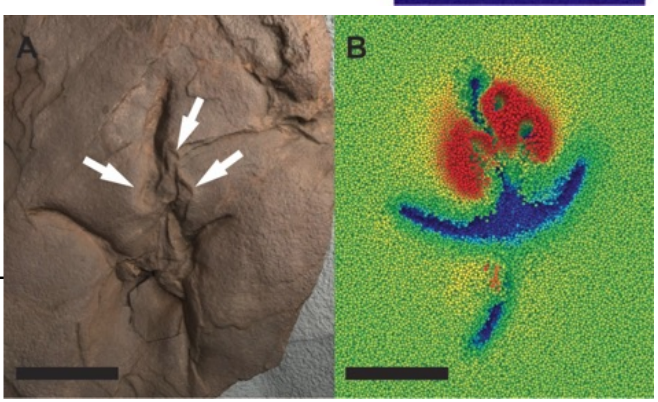
Peter L. Falkingham[a,b,1] and Stephen M. Gatesy[b]

[a]Structure and Motion Laboratory, Department of Comparative Biomedical Sciences, Royal Veterinary College, Hatfield AL97TA, United Kingdom; and [b]Department of Ecology and Evolutionary Biology, Brown University, Providence, RI 02912

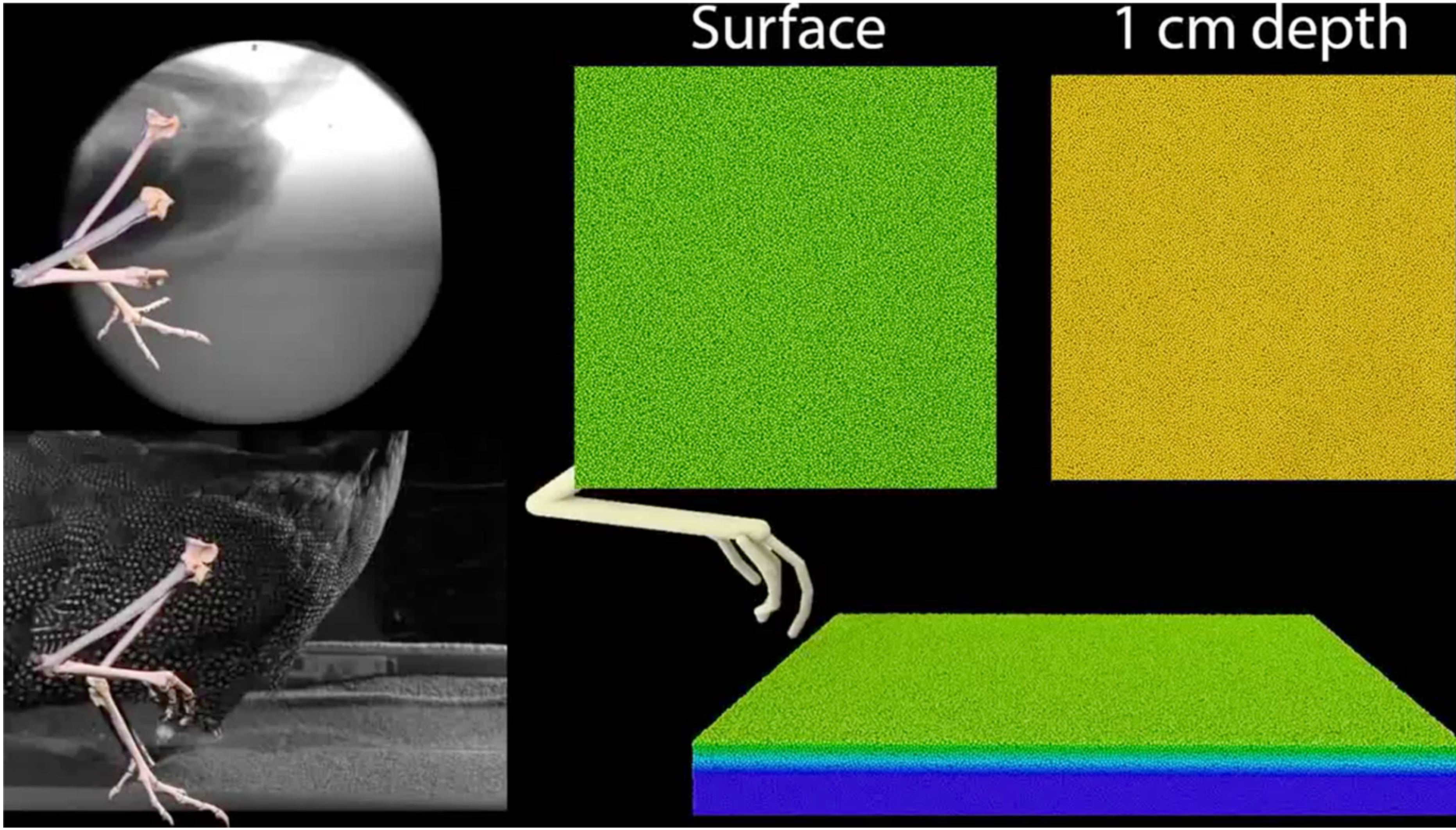
Edited by Neil H. Shubin, The University of Chicago, Chicago, IL, and approved October 30, 2014 (received for review August 22, 2014)



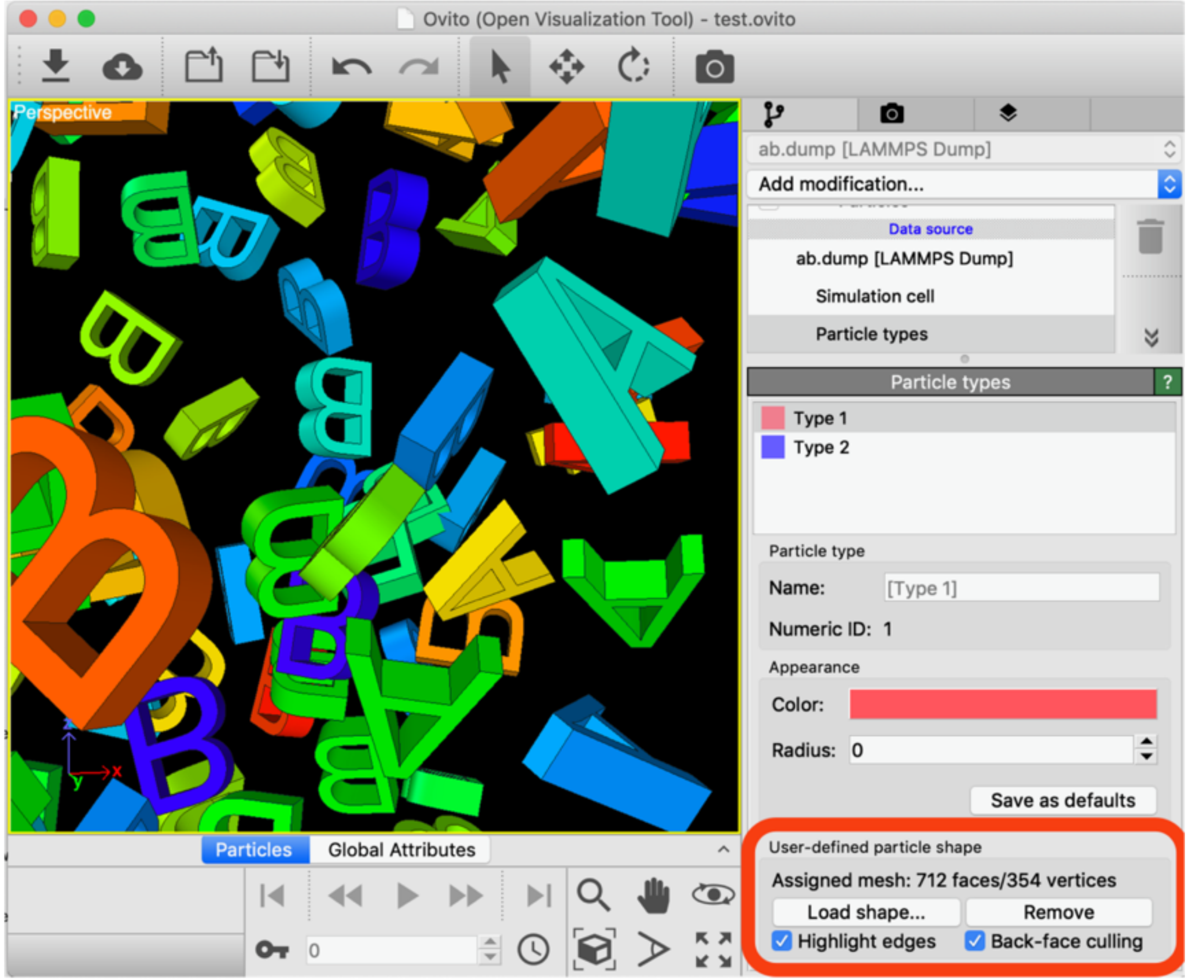




"The birth of a dinosaur track"



OVITO 3 will support user-defined particle shapes



- Particle shapes are loaded from geometry files (.obj, .stl, .vtk)
- Particle orientations controlled by ‘Orientation’ quaternion property.

Predictive Self-Assembly of Polyhedra into Complex Structures

Pablo F. Damasceno[1,*], **Michael Engel**[2,*], **Sharon C. Glotzer**[1,2,3,†]
+ See all authors and affiliations

Science 27 Jul 2012:
Vol. 337, Issue 6093, pp. 453-457
DOI: 10.1126/science.1220869

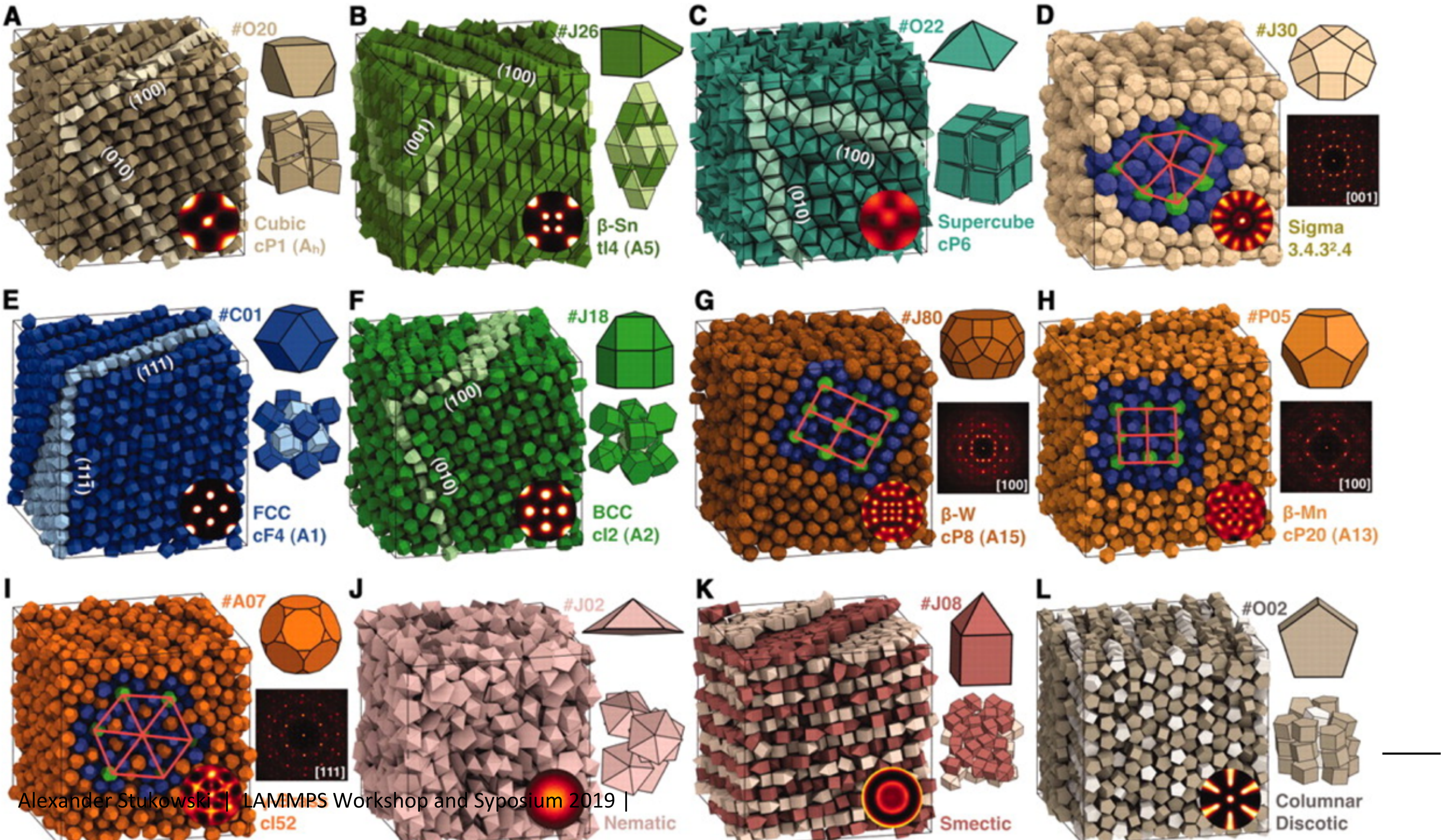
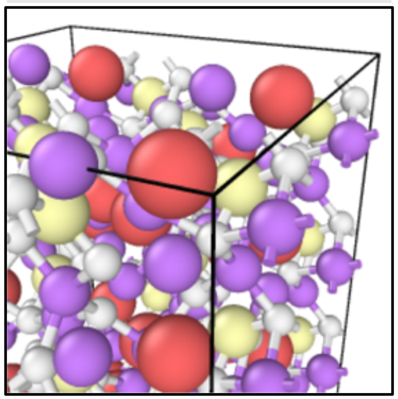


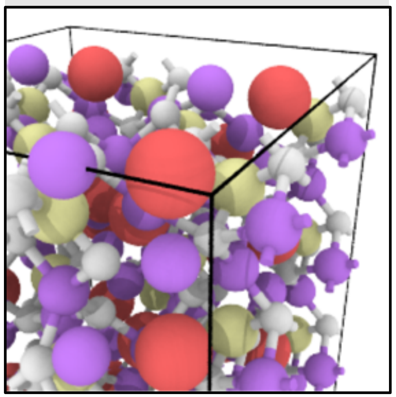
Image rendering

Integrated rendering engines:

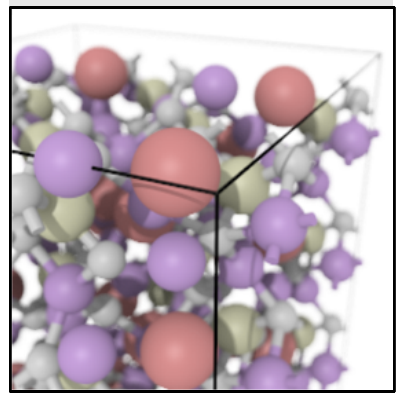
OpenGL

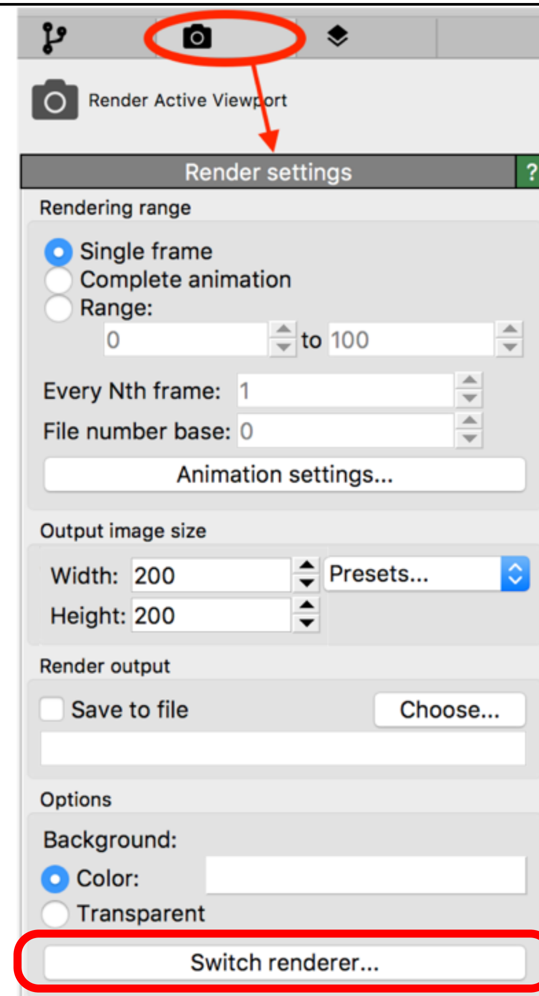


Tachyon (VMD)



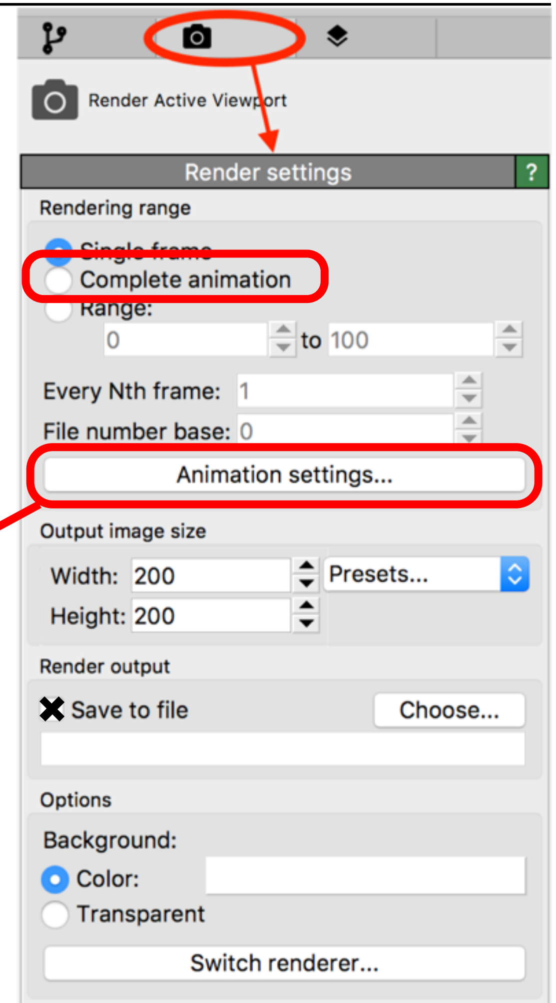
Intel OSPRay

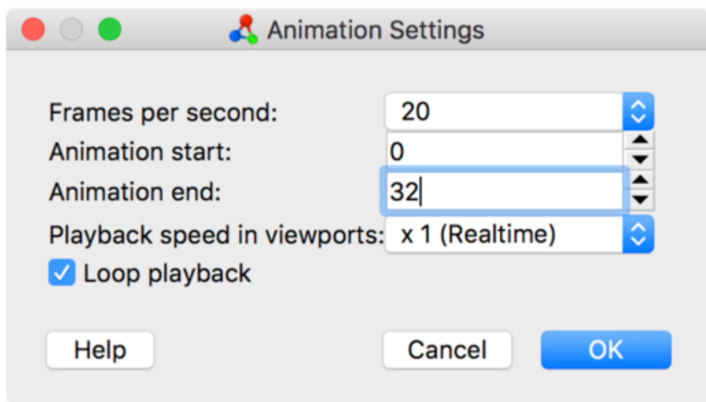




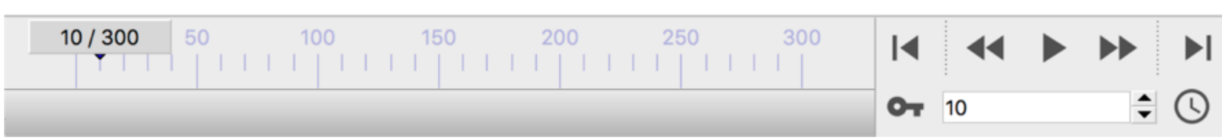
Animation rendering

- Can produce encoded videos in AVI, MP4 or MOV formats (also animated GIFs, but poor quality)
- Alternative approach: Render a series of image files (`img0.png, img1.png, img2.png, ...`) and use an external video encoding tool





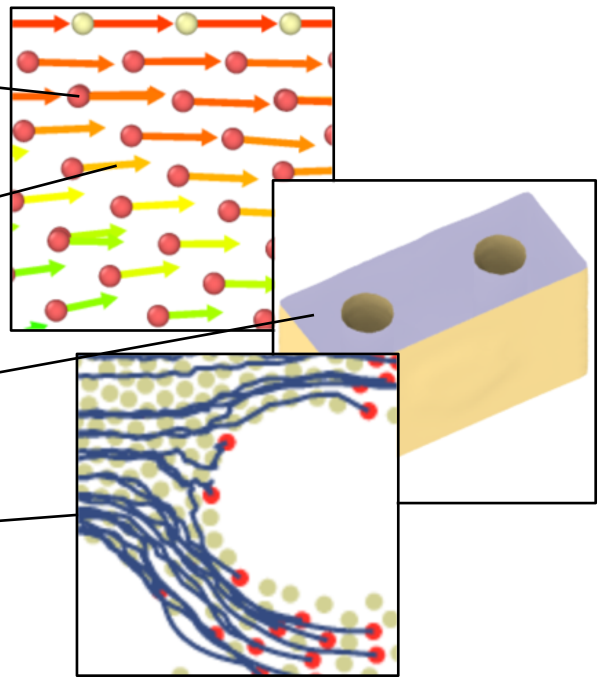
Time slider & timeline:



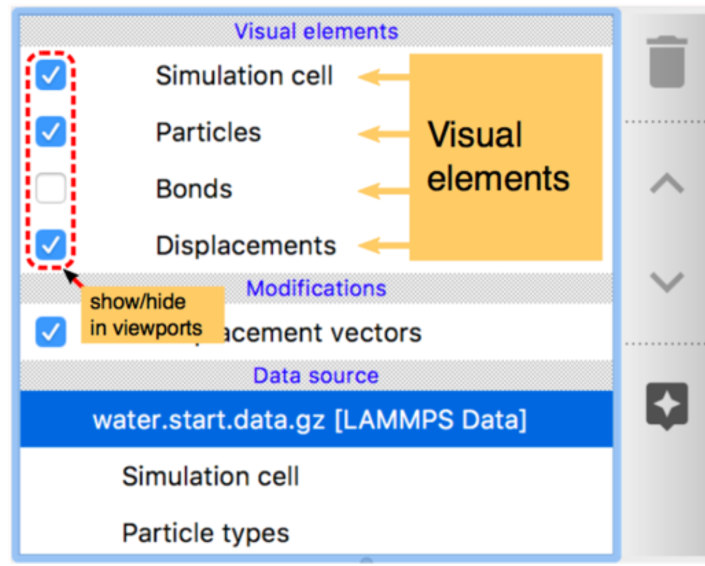
Visualization elements

Visual elements are graphical representations of data.

- **Particles**
- Bonds
- **Vector arrows**
- Simulation cell
- **Surface meshes**
- Polyhedra
- **Trajectory lines**
- Dislocation lines
- …

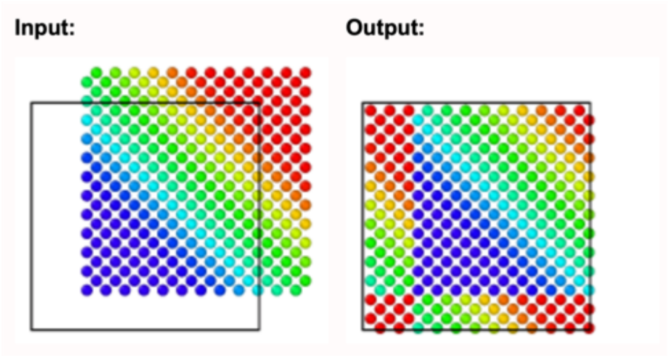


The *Visual elements* section of the pipeline editor:

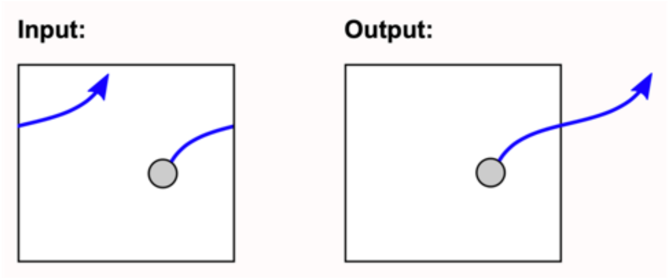


Trajectory wrapping/unwrapping/interpolation

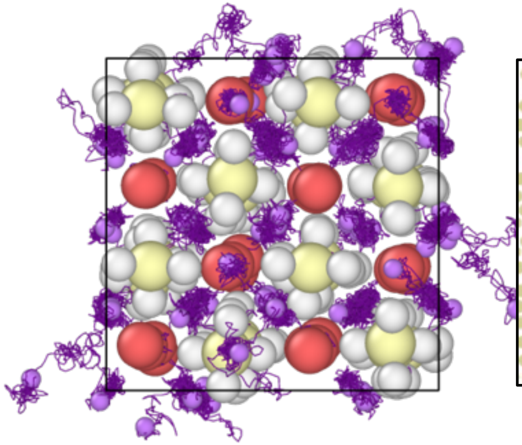
- Fold trajectories:

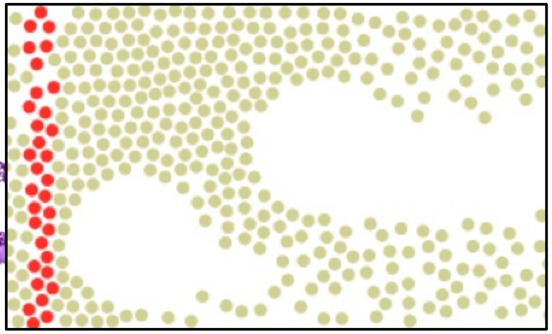


- Unfold trajectories:

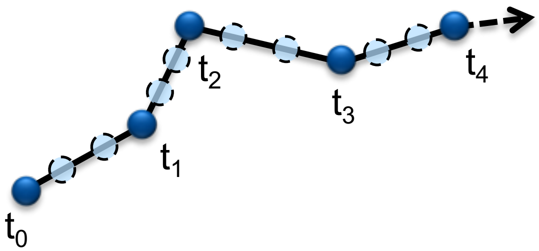


- Visualize trajectory lines:

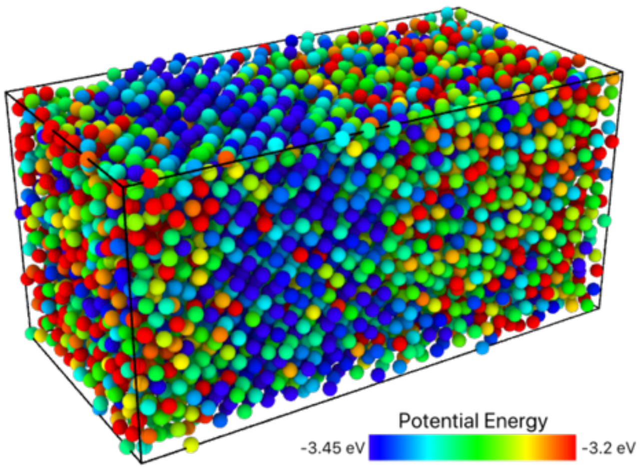


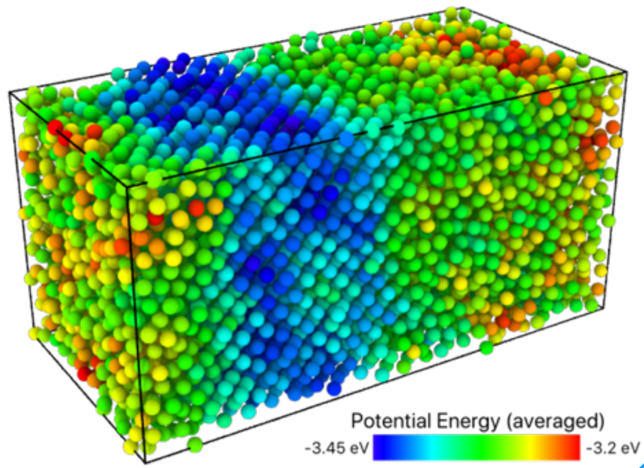


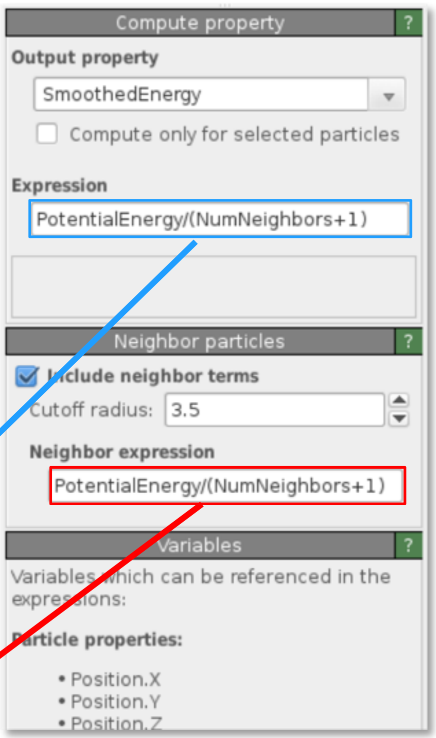
- Interpolate trajectories:

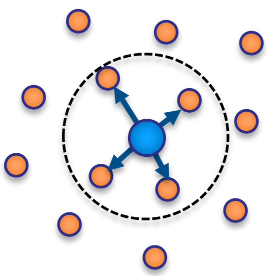


The *Compute Property* modifier



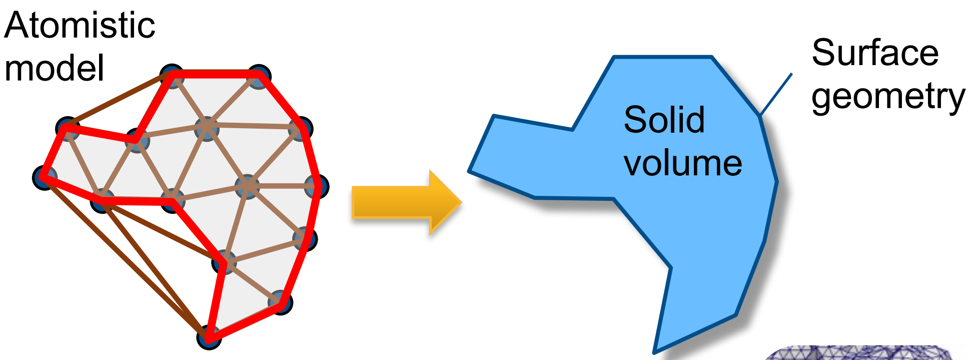


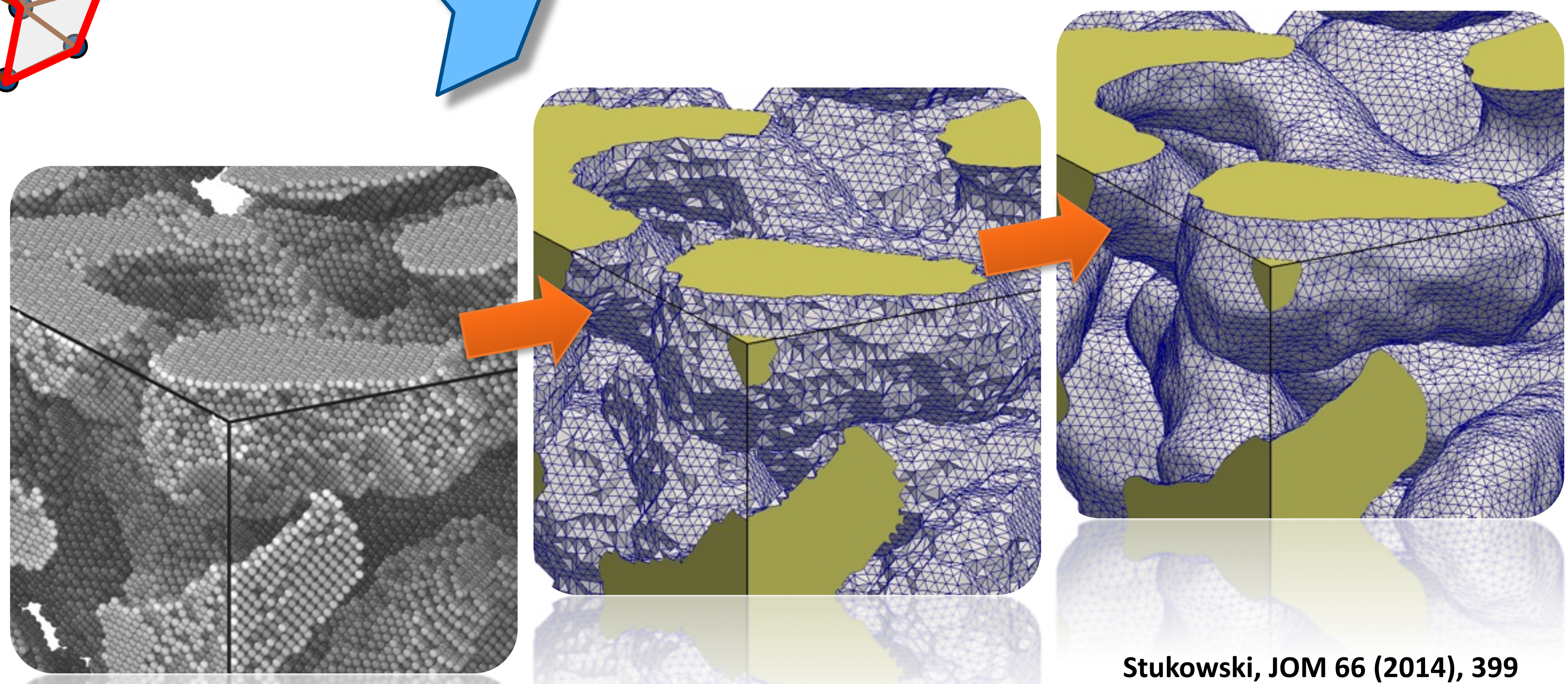




$$P(i) = F(i) + \sum_{j \in \mathcal{N}_i} G(j), \quad \text{with} \quad \mathcal{N}_i = \{j \colon |\mathbf{r}_i - \mathbf{r}_j| < R_c\}$$

Surface mesh construction from atomistic models



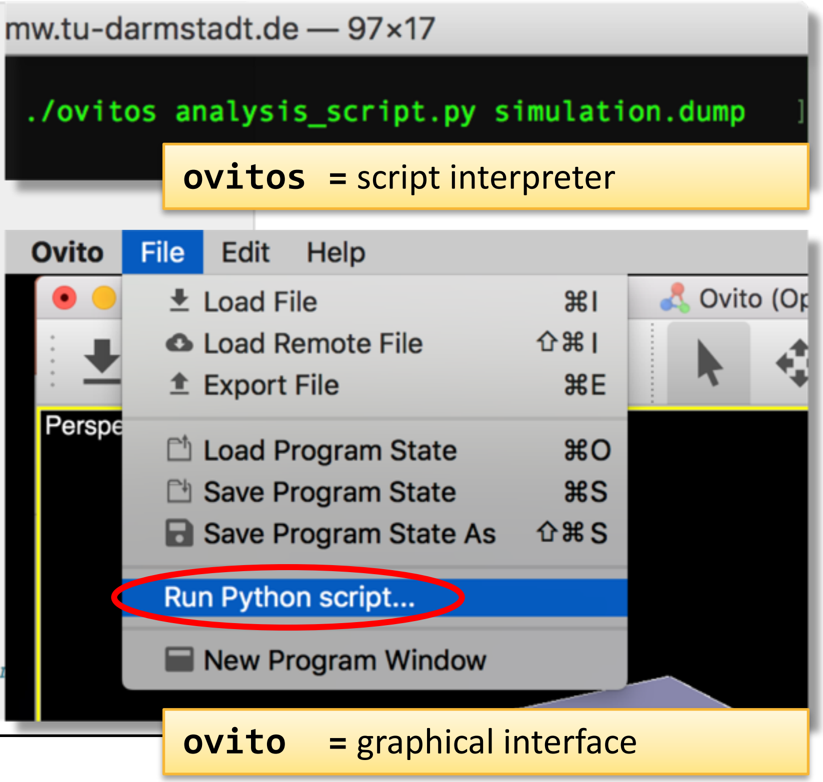


Stukowski, JOM 66 (2014), 399

Python scripting interface

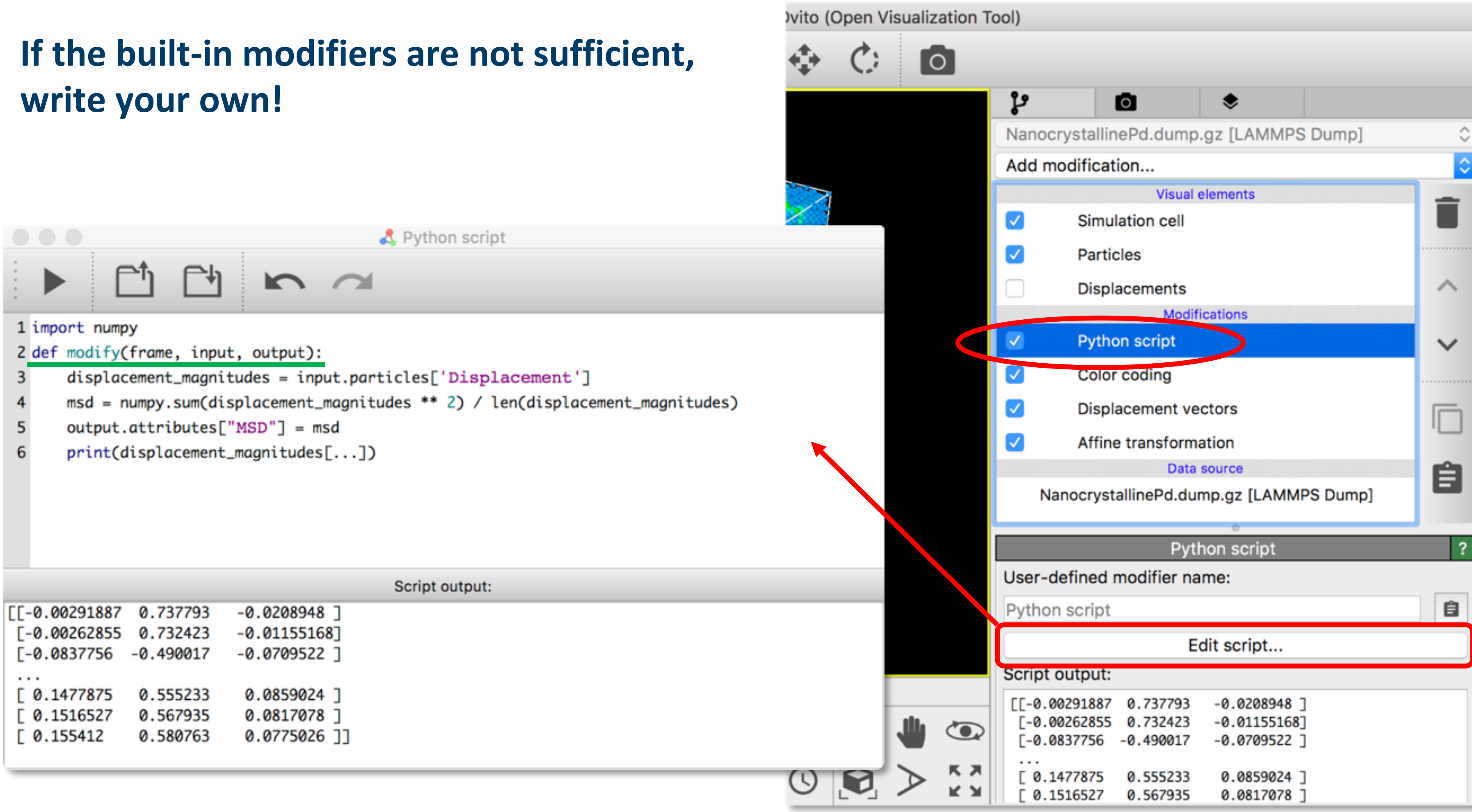
- OVITO‘s Python interface provides access to almost any program function and the data model
- Use it to automate analysis or visualization tasks!
- Use it to integrate OVITO’s capabilities into custom analysis workflows!

```
# Import OVITO modules.
from ovito.io import *
from ovito.modifiers import *
from ovito.data import *

# Import standard Python and NumPy modules.
import sys
import numpy

# Load the simulation dataset to be analyzed.
pipeline = import_file("../data/NanocrystallinePd.dump.gz")

# Create bonds.
pipeline.modifiers.append(CreateBondsModifier(cutoff = 3.5))

# Compute CNA indices on the basis of the created bonds.
pipeline.modifiers.append(CommonNeighborAnalysisModifier(
    mode = CommonNeighborAnalysisModifier.Mode.BondBased))

# Let OVITO's data pipeline do the heavy work.
data = pipeline.compute()

# The 'CNA Indices' bond property is a a two-dimensional array
# containing the three CNA indices computed for each bond in the syste
cna_indices = data.bonds['CNA Indices']
```



User-defined modifier functions

If the built-in modifiers are not sufficient, write your own!



Branched pipelines

Data pipelines can be branched:

1. Visualize the same input dataset in different ways
2. Visualize different datasets in the same way, side by side