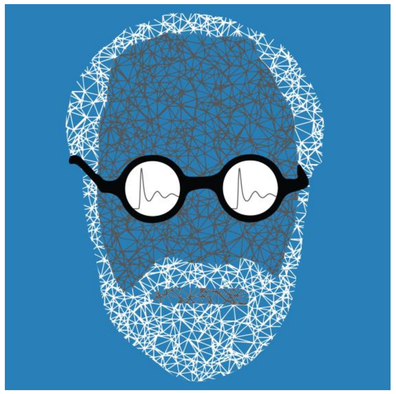


Fast and Efficient Particle Trajectory Analysis with the *freud* Library

Tommy Waltmann[1], Bradley Dice[1], Vyas Ramasubramani[2], Joshua Anderson[2], Sharon C. Glotzer[1234]

[1]*Department of Physics, University of Michigan*
[2]*Department of Chemical Engineering, University of Michigan*
[3]*Department of Materials Science and Engineering, University of Michigan*
[4]*Biointerfaces Institute, University of Michigan*


```
pip install freud-analysis
```

```
conda install -c conda-forge freud
```

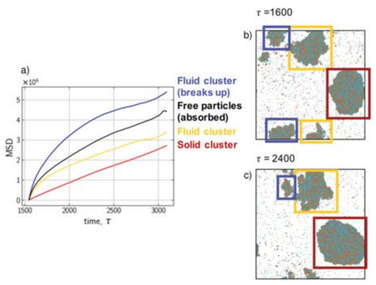
UNIVERSITY OF MICHIGAN

Introduction

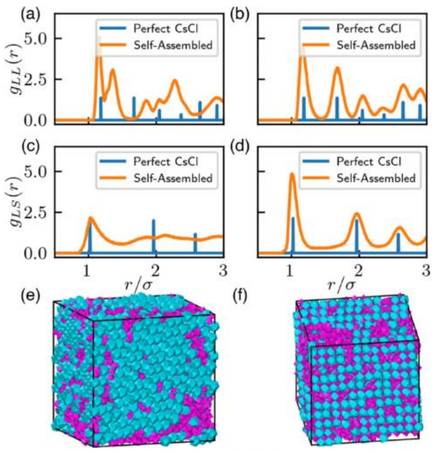
Modern day computational resources have allowed the expansion of research utilizing computer simulation to study larger and larger system sizes. Systems with millions of particles can now be simulated for hundreds of thousands of timesteps in a matter of days due to large amounts of effort into developing software which can run these simulations as fast as possible. However, without tools fit to perform analysis on these large systems, the research bottleneck shifts from the simulation side to the analysis side of the workflow. To this end, we present *freud*: a powerful and efficient particle trajectory analysis library which exposes an easy-to-use python API.

Standard Analysis Methods

The *freud* library computes many common analysis methods, such as radial distribution functions (RDF) and mean-squared displacements (MSD).



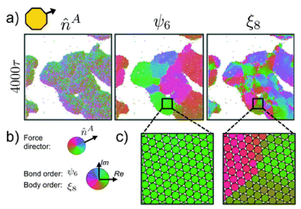
Moran, S. E., Schönhöfer, P. W. A., & Glotzer, S. C. (2022). *New Journal of Physics*, *24*(6), 063007.



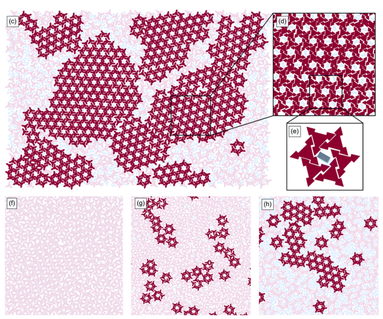
LaCour, R. A., Moore, T. C., & Glotzer, S. C. (2022). *Physical Review Letters*, *128*(18), 188001.

Order Parameters

The *freud* library computes a wide variety of topological order parameters such as the Steinhardt order parameters and their variants, nematic, translational, solid/liquid, and others.



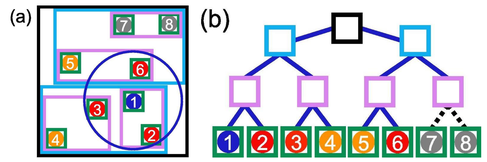
Moran, S. E., Bruss, I. R., Schönhöfer, P. W. A., & Glotzer, S. C. (2022). Soft Matter, 18(5), 1044–1053.



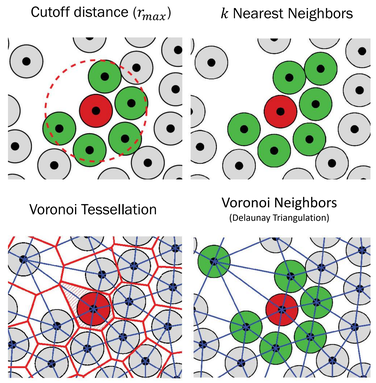
Moore, T. C., Anderson, J. A., & Glotzer, S. C. (2021). *Soft Matter*, *17*(10), 2840–2848.

Neighbor Finding

Whether explicit or implicit, many applications of the *freud* library require neighbor finding, which can be accelerated by using specialized data structures for querying points.



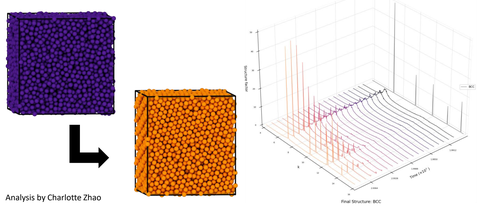
Howard, M. P., Statt, A., Madutsa, F., Truskett, T. M., & Panagiotopoulos, A. Z. (2019). *Computational Materials Science*, *164*, 139–146.



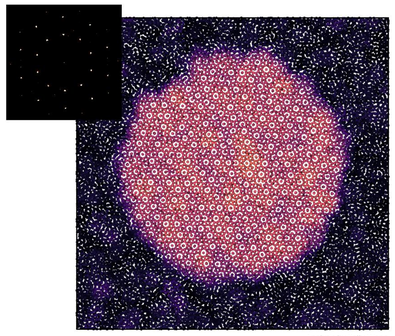
Mickel, W., Kapfer, S. C., Schröder-Turk, G. E., & Mecke, K. (2013). *The Journal of Chemical Physics*, *138*(4), 044501.

Scattering Analysis

In more recent *freud* developments, many new scattering analysis features, such as diffraction patterns and static structure factors, have been added to the codebase.



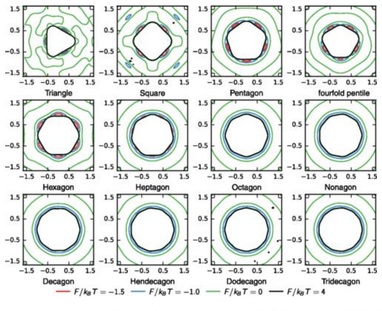
Analysis by Charlotte Zhao



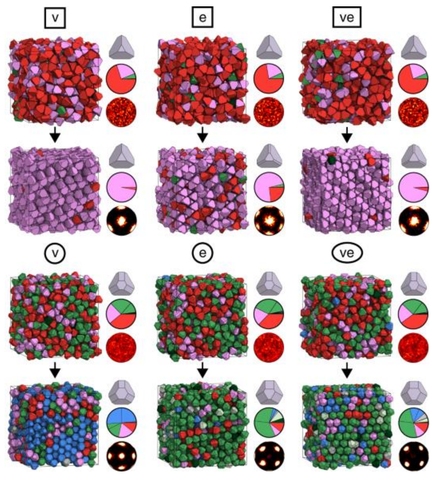
Analysis by Kelly Wang

Other Analysis Methods

The *freud* library also contains other analysis methods rarely or not seen elsewhere such as PMFTs, environment matching, and interface detection.



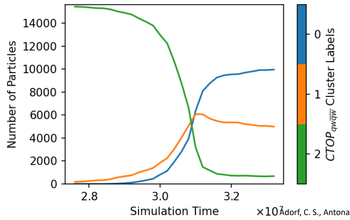
Anderson, J. A., Antonaglia, J., Millan, J. A., Engel, M., & Glotzer, S. C. (2017). *Physical Review X*, *7*(2), 021001.

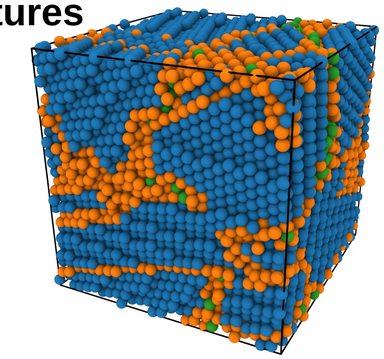


Teich, E. G., van Anders, G., & Glotzer, S. C. (2019). *Nature Communications*, *10*(1), 64.

Machine Learning Features

In more sophisticated applications, order parameters from the *freud* library have been used as features for machine learning applications.



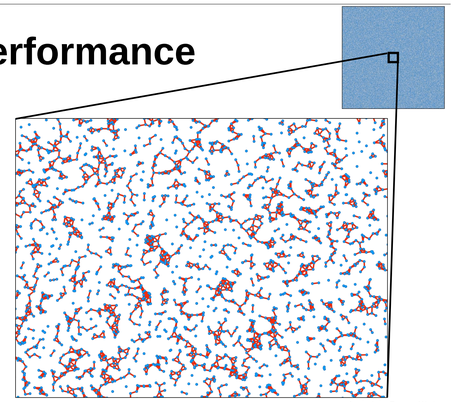


cF4-Cu self-assembly with FF-REM potential

×10Ådorf, C. S., Antonaglia, J., Dshemuchadse, J., & Glotzer, S. C. (2018). *The Journal of Chemical Physics*, *149*(20), 204102. Analysis by Bradley Dice

Performance

The *freud* library was built to compute results quickly. The figure to the right shows a system of 1,000,000 particles (blue) randomly distributed in a 2D box at density 0.01. Freud computed all the neighbors (red) for this system in 3.18 seconds using 12 threads.



Analysis by Tommy Waltmann

Python Ecosystem

The *freud* library's NumPy array interfaces allow it to integrate tightly within the scientific python ecosystem.

```python
@FlowProject.operation
@FlowProject.pre.isfile('assembly_sim.gsd')
def compute_rdf(job):
    import freud
    import gsd.hoomd
    import matplotlib.pyplot as plt

    traj = gsd.hoomd.open(job.fn('assembly_sim.gsd'))

    rdf = freud.density.RDF(bins=100, r_max=3.6)
    for frame in traj:
        pos = frame.particles.position[frame.particles.typeid == 0]
        rdf.compute((frame.configuration.box, pos), reset=False)

    # plot
    plt.plot(rdf.bin_centers, rdf.rdf)
    plt.savefig(job.fn("rdf.png"))
    plt.close()
```

Code: github.com/glotzerlab/freud
Documentation: freud.readthedocs.io