Biology simulations with GPUs

Jean Feydy
HeKA team, Inria Paris
Inserm, Université Paris-Cité
PR[AI]RIE-PSAI Institute

9th of April, 2026
Séminaire Modélisation – LJLL, Jussieu

My main motivation

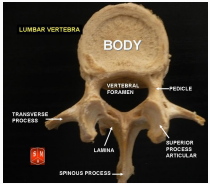
Develop **robust and efficient** software that **stimulates other researchers**:

1. Speed up **geometric machine learning** on GPUs:
   $\Longrightarrow$ **pyKeOps** library for distance and kernel matrices, 1M+ downloads.

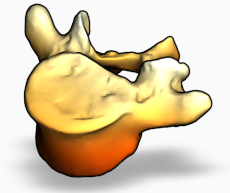
2. Scale up **pharmacovigilance** to the full French population:
   $\Longrightarrow$ **survivalGPU**, a fast re-implementation of the R survival package.

3. Ease access to modern statistical **shape analysis**:
   $\Longrightarrow$ **GeomLoss**, truly scalable optimal transport in Python.
   $\Longrightarrow$ **scikit-shapes**, beta release available soon.

Speed and consistency with diverse data structures

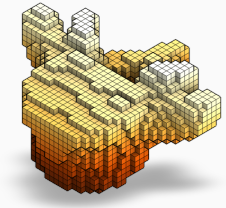


Genuine vertebra.

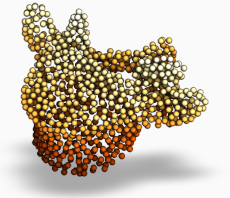


Surface mesh.

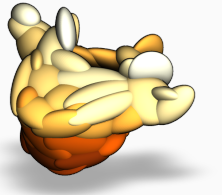
Volumetric mesh.



Segmentation mask.



Point cloud.



Gaussian mixture.

Today's talk

Graphics Processing Units provide a **x1,000 speed-up** for parallel computations.

Bruteforce computing $\neq$ Switch **off** your brain.
Bruteforce computing $=$ Modelling **freedom**!

Two examples, enabled by PyTorch ($\simeq$ Matlab, for GPUs) and my "KeOps" extension:

1. A phase-field model for the **synthesis of biological microstructures**.
2. A simple framework to simulate **incompressible biological cells**.

Biological microstructures

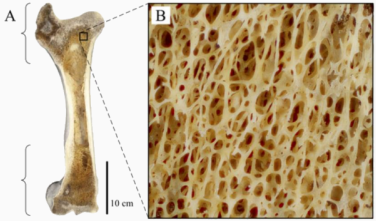
Memories from the Covid lockdown…

Anna Song

Francis Crick Institute,

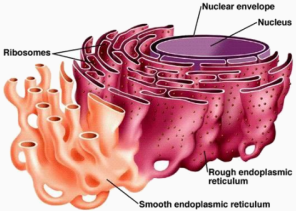
now at Dassault Systèmes.

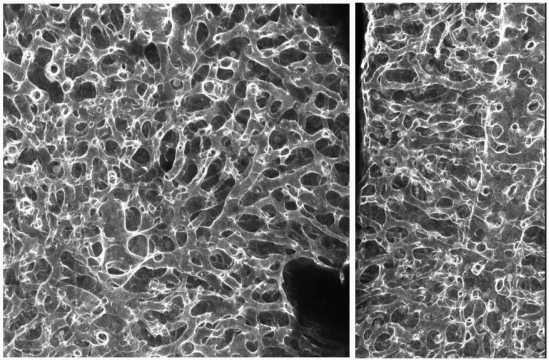
Biological motivation

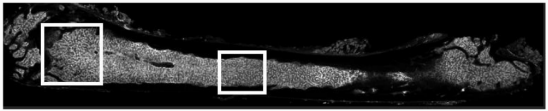


Trabecular bone has plates and rods,
from Bishop et al., PeerJ (2018)

Endoplasmic reticulum has sheets and tubules

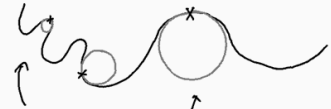






Curvature of a 2D surface

$$\boxed{K = \frac{1}{r}}$$

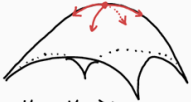


principal curvatures

$$\boxed{K_1 \text{ and } K_2}$$

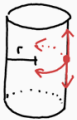
$K_1 = K_2 = 0$

$K_1, K_2 > 0$



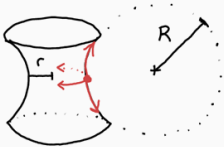
$K_1, K_2 \gg 0$

$K_1 = K_2 = \frac{1}{R}$



$K_1 = \frac{1}{r}$

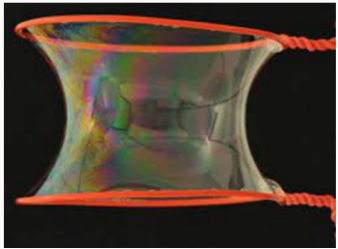
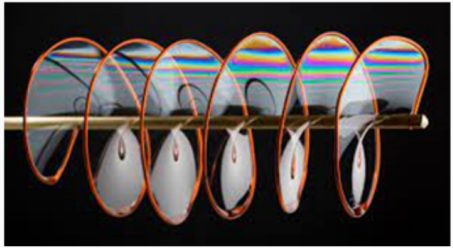
$K_2 = 0$



$K_1 = \frac{1}{r}$

$K_2 = -\frac{1}{R}$

Minimal surfaces in physics

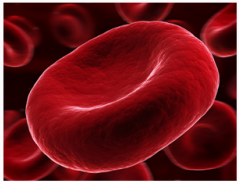
Soap bubbles minimize:

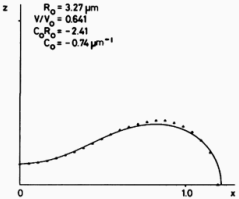
$$\mathrm{area}(\mathcal{S}) = \int_{\mathcal{S}} 1 \, dA$$

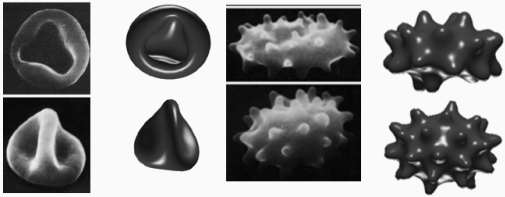
under constant volume, or with boundary conditions.

They correspond to minimal surfaces with $H = \kappa_1 + \kappa_2 = 0$ in cases 2 and 3.

Minimal surfaces in biology





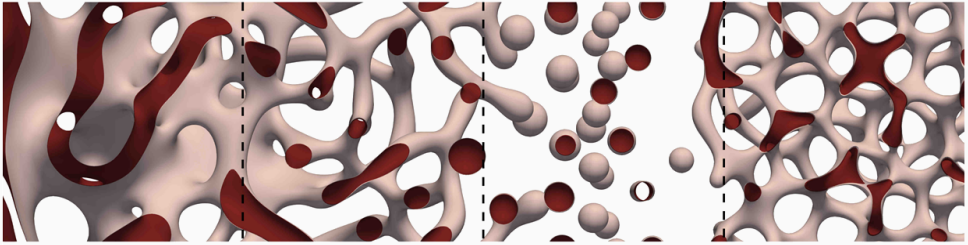


Red blood cells minimize:

$$\mathrm{Helfrich}(\mathcal{S}) = \int_{\mathcal{S}} (H - H_0)^2 \, dA$$

or a variant of this energy, under constant volume.

The curvatubes model [Son22]



$H^2 + \kappa_1^2 + \kappa_2^2$ | $(H-20)^2 + 5\,\kappa_2^2$ | $(H-25)^2 + (\kappa_1 - 12.5)^2 + (\kappa_2 - 12.5)^2$ | $(H-45)^2 + (\kappa_1 - 45)^2 + 10\,\kappa_2^2$

Curvatubes minimize:

$$F(\mathcal{S}) = \int_{\mathcal{S}} p(\kappa_1, \kappa_2)\, dA$$

under constant volume, where:

$$p(\kappa_1, \kappa_2) = a_{2,0}\,\kappa_1^2 + a_{1,1}\,\kappa_1\kappa_2 + a_{0,2}\,\kappa_2^2 + a_{1,0}\,\kappa_1 + a_{0,1}\,\kappa_2 + a_{0,0}\,.$$

Under the hood: a phase-field formulation

$$\mathbf{E}_A(\mathcal{S}) = \int_{\mathcal{S}} 1 \, dA$$

minimal surfaces (1750')

$$\mathbf{E}_{\mathrm{W}}(\mathcal{S}) = \int_{\mathcal{S}} H^2 \, dA$$

Willmore (1960')

$$\mathbf{E}_{\mathrm{H}}(\mathcal{S}) = \int_{\mathcal{S}} \left(\frac{\chi_b}{2} (H - H_0)^2 + \chi_G K \right) dA$$

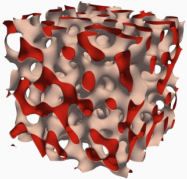
Helfrich (1970')

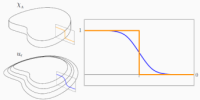
$$\mathbf{F}(\mathcal{S}) = \int_{\mathcal{S}} p(\kappa_1, \kappa_2) \, dA$$

Curvatubes (2021)

$$\mathbf{F}(\mathcal{S}) = \int_{\mathcal{S}} p(\kappa_1, \kappa_2) \, dA$$

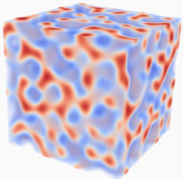
2D surface energy
hard to simulate





$$\mathcal{E}_\epsilon(u) = \int_{\Omega} p(\kappa^{\epsilon}_{1,u}, \kappa^{\epsilon}_{2,u}) \, \epsilon |\nabla u|^2 \, dx$$

3D phase-field energy
easy to simulate on GPUs



https://github.com/annasongmaths/curvatubes

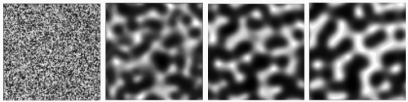
Texture generation via gradient descent on a convolutional energy

curvatubes

parameters inside energy

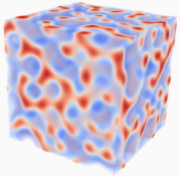
$$\dot{u} = \Delta \frac{\partial \mathcal{F}_\epsilon}{\partial u}$$

random initialization



=

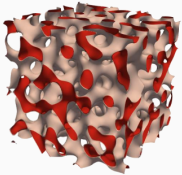
phase-field



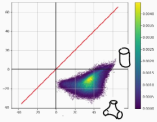
$$\mathcal{F}_\epsilon(u) = \int_\Omega \Bigg[\frac{a_{2,0} + a_{0,2} - a_{1,1}}{2\epsilon} \|\mathcal{M}_u^\epsilon\|^2 + \frac{a_{1,1}}{2\epsilon} (\mathrm{Tr}\mathcal{M}_u^\epsilon)^2 + \frac{a_{2,0} - a_{0,2}}{2\epsilon} \mathrm{Tr}\mathcal{M}_u^\epsilon \sqrt{(2\|\mathcal{M}_u^\epsilon\|^2 - (\mathrm{Tr}\mathcal{M}_u^\epsilon)^2)^+} + \frac{a_{1,0} + a_{0,1}}{2} |\nabla u| \mathrm{Tr}\mathcal{M}_u^\epsilon + \frac{a_{1,0} - a_{0,1}}{2} |\nabla u| \sqrt{(2\|\mathcal{M}_u^\epsilon\|^2 - (\mathrm{Tr}\mathcal{M}_u^\epsilon)^2)^+} + a_{0,0}\, \epsilon |\nabla u|^2 \Bigg] dx.$$

Pytorch
GPU
Adam or L-BFGS

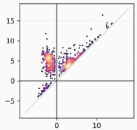
outputs



surface



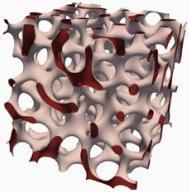
curvature diagram

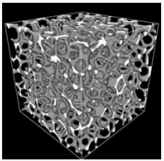


persistence diagram

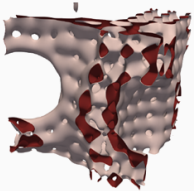
other features…

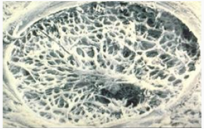
A surprisingly expressive model



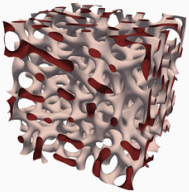


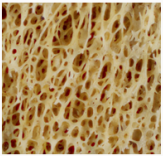
μCT image of open aluminium foam



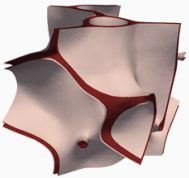


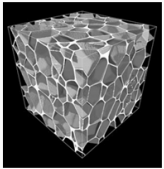
lamina cribrosa behind the eye



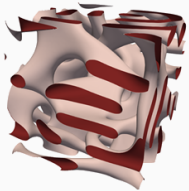


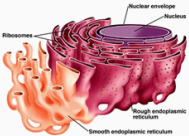
trabecular bone





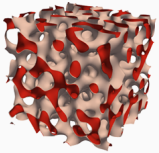
μCT image closed polymer foam



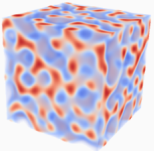


endoplasmic reticulum

PyTorch and GPUs go way beyond deep learning research



$\longrightarrow$



$$\mathbf{F}(S) = \int_S p(\kappa_1, \kappa_2)\, dA$$

2D surface energy

$$\mathcal{E}_\epsilon(u) = \int_\Omega p(\kappa^\epsilon_{1,u}, \kappa^\epsilon_{2,u})\, \epsilon |\nabla u|^2\, dx$$

3D phase-field energy

An **inspiring** model:

- **Surface** energy $\rightarrow$ **convolutional volumetric** loss function (phase-field).
- Start with **white noise** (texture generation) and minimize with **gradient descent**.
- Implemented on GPU with **PyTorch**.

$\Longrightarrow$ Combines **maths** + **GPU** computing + **imaging** data

Incompressible particles

Two very talented colleagues

Maciej Buze

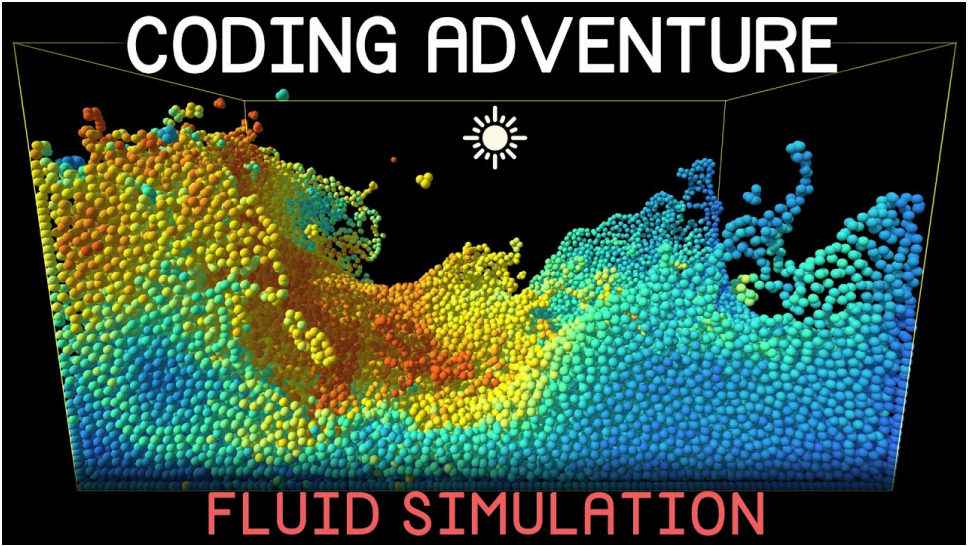
Lancaster University

Antoine Diez

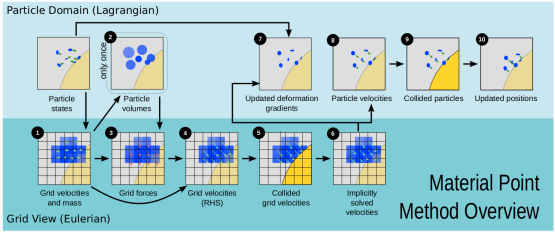
RIKEN iTHEMS, Tokyo

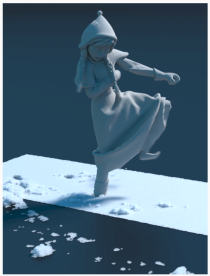
GPU programming 101: the N-body problem [Pri11]

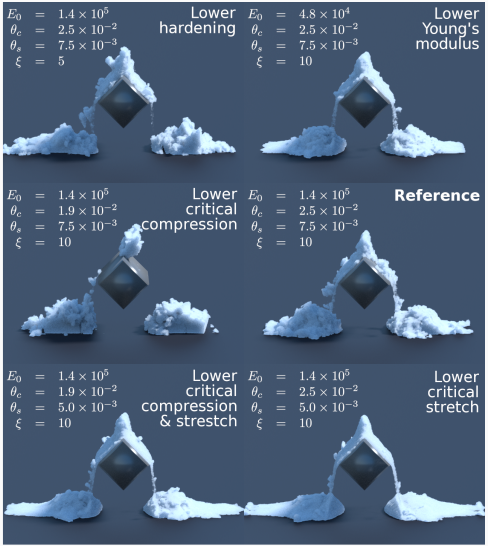
Coding a simple fluid simulation is now a matter of hours [Lag23]



The material point method: Disney's Frozen [SSC+13]



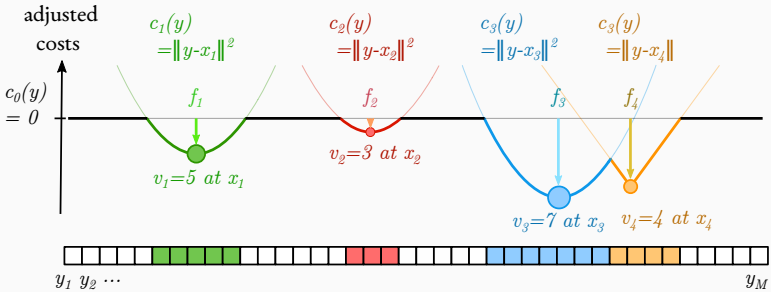


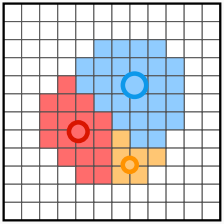


How can we enforce a volume preservation constraint? [QLDGJ22]

2D FLIP Simulation

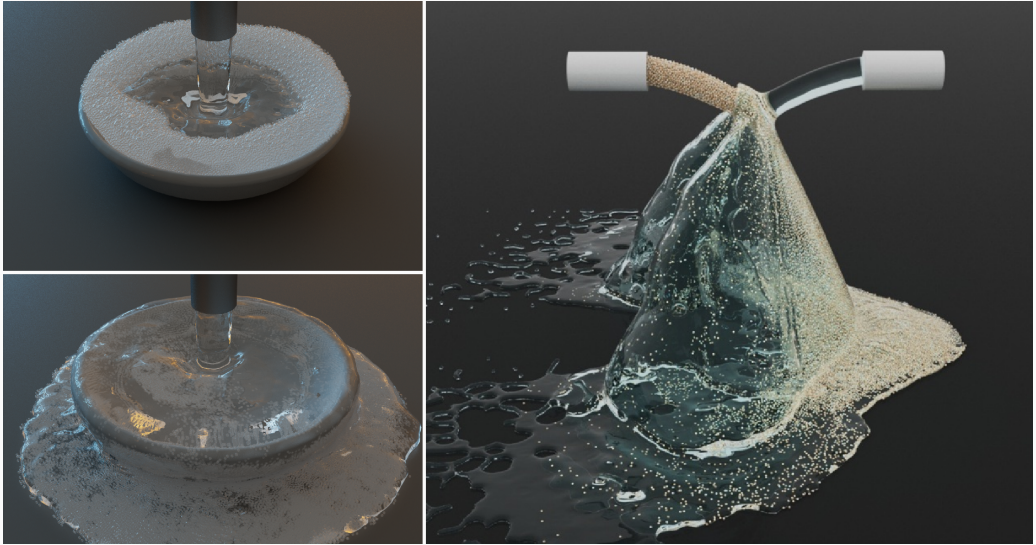
Use power diagrams i.e. semi-discrete optimal transport



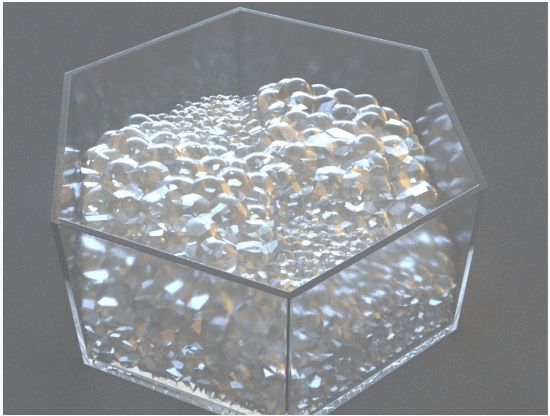


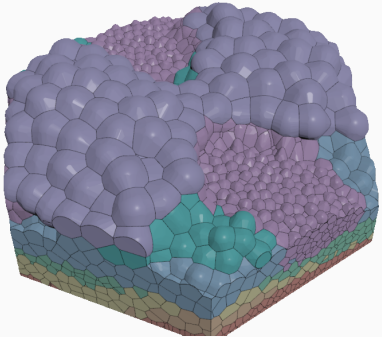
- The f_i's maximize the dual objective $\sum_{i=1}^N v_i f_i + \int_{y\in\Omega} \min_{i=0}^N [\, c_i(y) - f_i \,]\, \mathrm{d}y$.
- **Optimality** conditions $\iff$ $\mathrm{Vol}(\mathrm{Cell}_i) = v_i$.
- To **compute the cells**, the objective and its gradient:
  - If $c_i(y) = \|y - x_i\|^2$ for all cells, use a clever **grid-free** algorithm.
  - Otherwise, just use **KeOps**.

Power plastics [QLY+23]



Power plastics [QLY+23] – without the eye candy





Main numerical ingredients

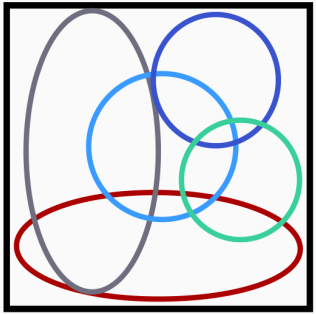
These simulations alternate between:

1. **Moving the particles** according to your favorite N-body model.
2. Computing Laguerre **cells** with the **correct volumes**:
   - (Multiscale) Sinkhorn for tolerance $> 5\%$.
   - (Quasi-)Newton for tolerance $< 1\%$.
3. **Correcting** the particle positions to enforce the volume-preservation constraint:
   - Jump to the centroid of the cell.
   - Or add a spring for smoother trajectories.

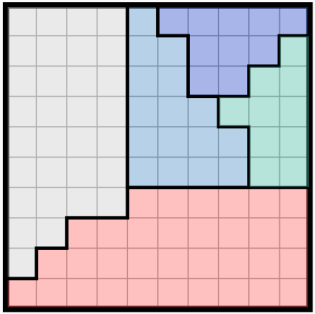
See e.g. Thomas Gallouët for a rigorous analysis with Mérigot, Lévy, etc.

But today: new applications with **custom cost functions** (thanks KeOps).

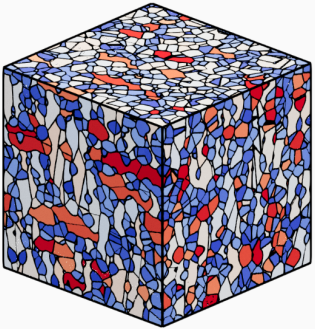
Anisotropic power diagrams let us model polycrystalline metals [BFR+24]



Ellipsoids.

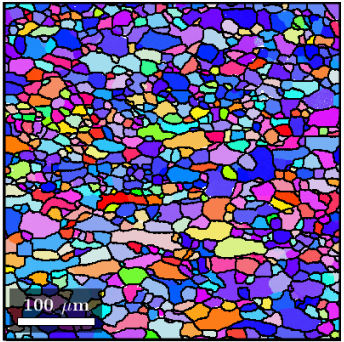


Pixel cells.

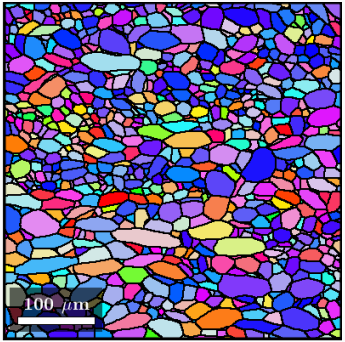


5,000 crystals in 3D.

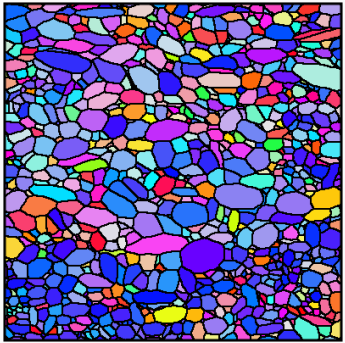
Fit to real EBSD scan of low-carbon steel [BFR+24]



Data from Tata steel.



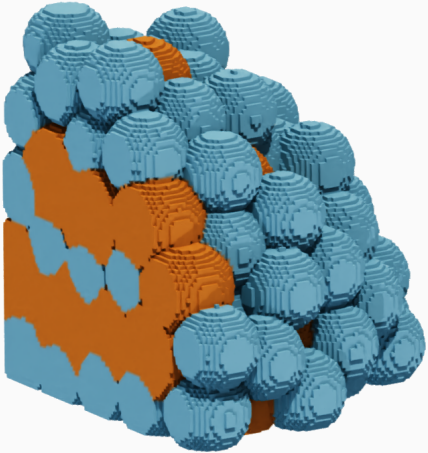
Our APD model.



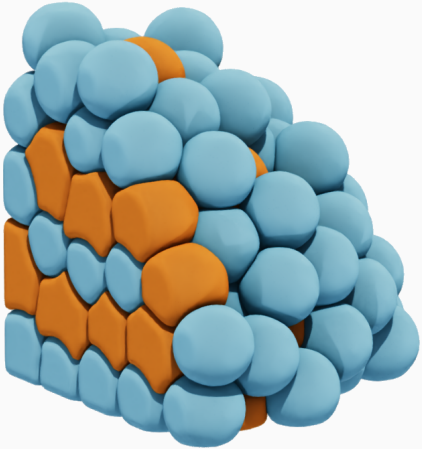
New synthetic image.

We can generate new, realistic 3D images with **prescribed properties** in seconds.

Change the cost function to simulate hard (blue) and soft (orange) cells [DF24]



The **raw** 100x100x100 pixel grid…

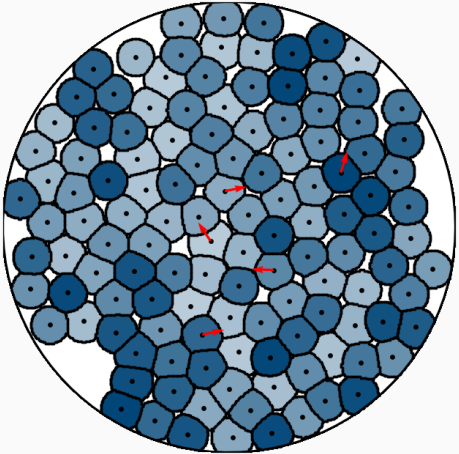


with some Hollywood **makeup**.

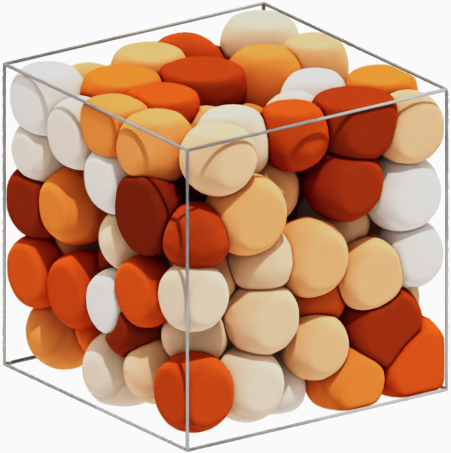
Let’s visit Antoine’s website

$\Longrightarrow$ https://iceshot.readthedocs.io $\Longleftarrow$

Run-and-tumble motion [DF24]

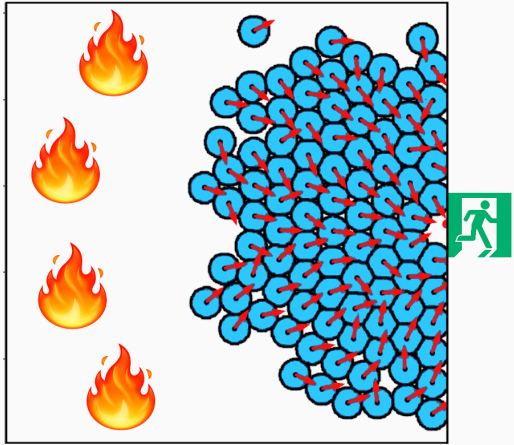


2D disk.

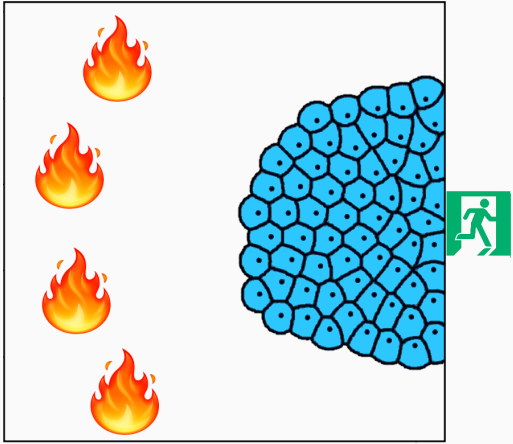


3D cube.

Fire alarm! [DF24]

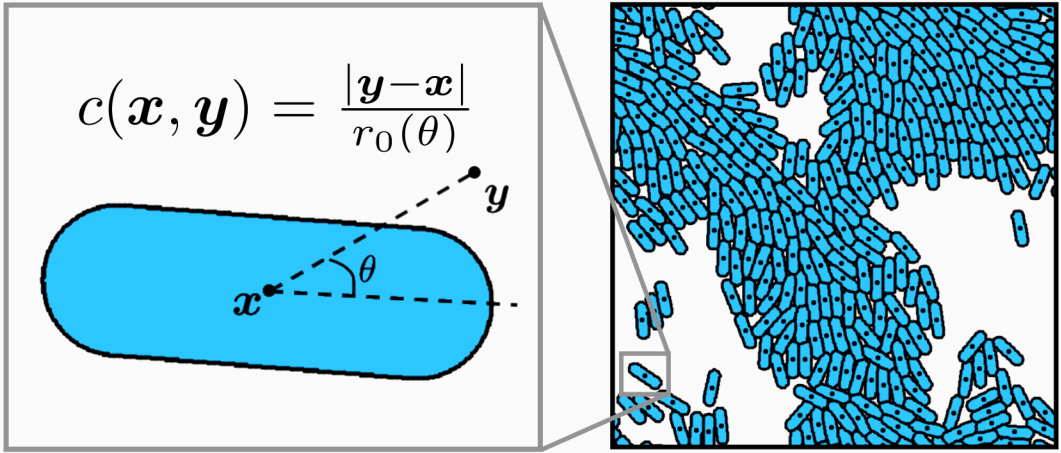


Hard particles **burn**.

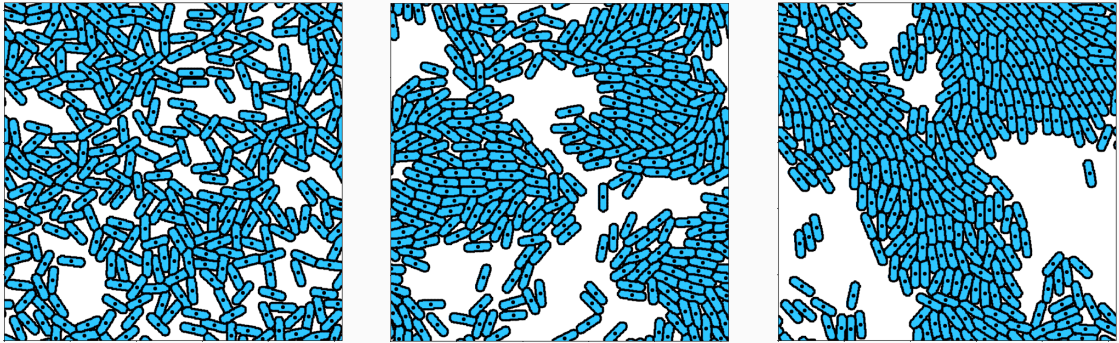


Soft particles **escape**.

Self-organizing swarms of blind, incompressible swimmers [DF24]



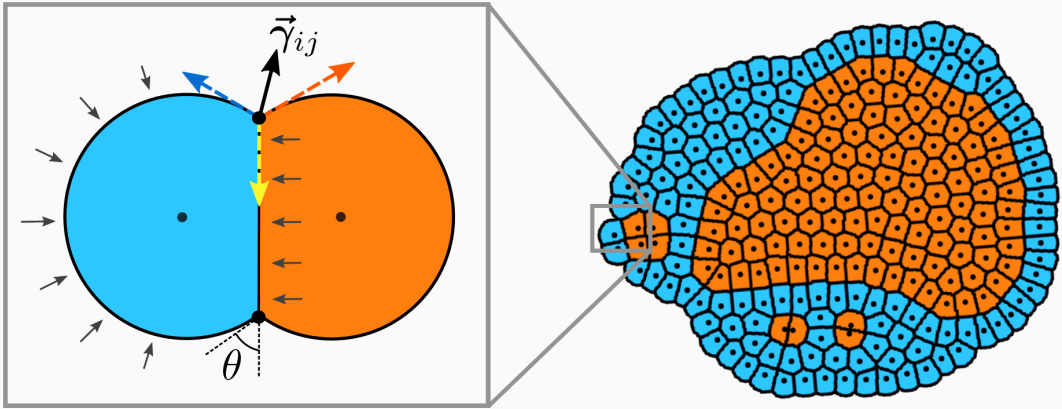
Self-organizing swarms of blind, incompressible swimmers [DF24]



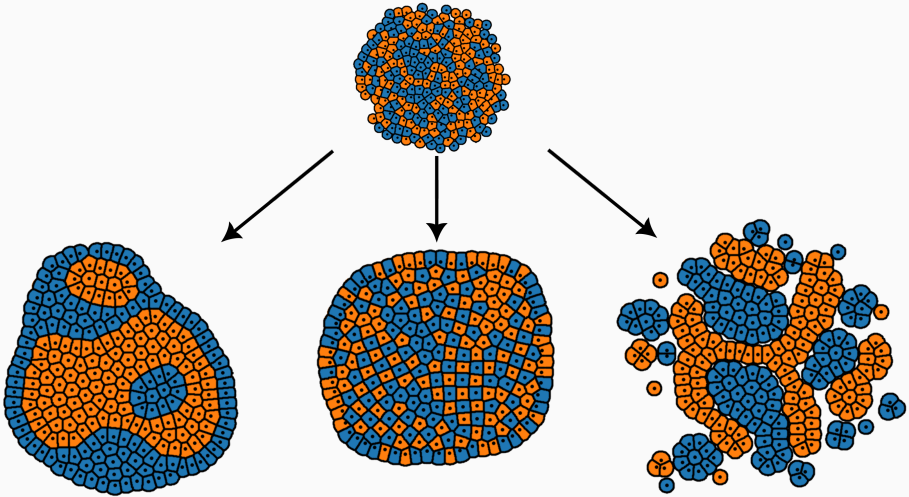
$t = 0$ $t = 4$ $t = 30$

Order emerges out of blind collisions and re-alignments.

Surface tension [DF24]



Surface tension [DF24] – playing with the energy parameters



Conclusion

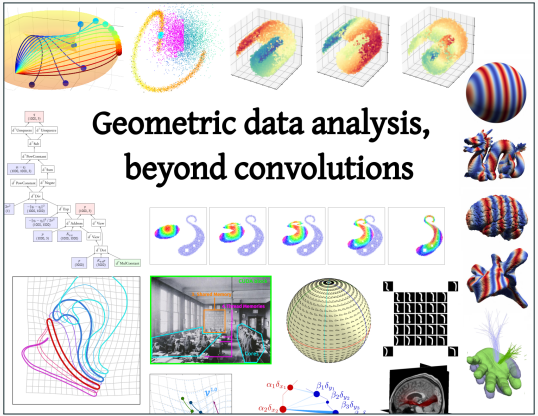
Key points

- **“Video-game physics”** is great for modelling:
  $\longrightarrow$ **Expressive**, real-time simulations that you can implement without being a Finite Elements guru: XPBD, DiffPD, Taichi…
- **GPUs** are more **versatile** than you think.
  $\longrightarrow$ Ongoing work to provide **fast GPU backends** to researchers, going beyond what Google and Facebook are ready to pay for.

Target for scientific Python: **interactive, web-based** simulations à la ShaderToy.

Documentation and tutorials are available online

$\Longrightarrow$ www.kernel-operations.io $\Longleftarrow$



www.jeanfeydy.com/geometric_data_analysis.pdf

Documentation and tutorials are available online

$\Longrightarrow$ shape-analysis.github.io $\Longleftarrow$

Monthly seminar, videos on YouTube.

References

References i

M. Agueh and G. Carlier.

Barycenters in the Wasserstein space.

SIAM Journal on Mathematical Analysis, 43(2):904–924, 2011.

Dimitri P Bertsekas.

A distributed algorithm for the assignment problem.

Lab. for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA, 1979.

References ii

Maciej Buze, Jean Feydy, Steven Roper, Karo Sedighiani, and David P Bourne.

Anisotropic power diagrams for polycrystal modelling: efficient generation of curved grains via optimal transport.

arXiv submission 5452163, 2024.

Haili Chui and Anand Rangarajan.

A new algorithm for non-rigid point matching.

In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 44–51. IEEE, 2000.

References iii

Marco Cuturi.

Sinkhorn distances: Lightspeed computation of optimal transport.

In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.

Antoine Diez and Jean Feydy.

An optimal transport model for dynamical shapes, collective motion and cellular aggregates, 2024.

References iv

Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness.

New algorithms for 2d and 3d point matching: Pose estimation and correspondence.

Pattern recognition, 31(8):1019–1031, 1998.

Leonid V Kantorovich.

On the translocation of masses.

In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.

References v

Harold W Kuhn.

The Hungarian method for the assignment problem.

Naval research logistics quarterly, 2(1-2):83–97, 1955.

Jeffrey J Kosowsky and Alan L Yuille.

The invisible hand algorithm: Solving the assignment problem with statistical physics.

Neural networks, 7(3):477–490, 1994.

References vi

Sebastian Lague.

Coding adventure: Simulating fluids.

https://www.youtube.com/watch?v=rSKMYc1CQHE&t=1s, 2023.

Bruno Lévy.

A numerical algorithm for l2 semi-discrete optimal transport in 3d.

ESAIM: Mathematical Modelling and Numerical Analysis, 49(6):1693–1715, 2015.

References vii

Quentin Mérigot.

A multiscale approach to optimal transport.

In *Computer Graphics Forum*, volume 30, pages 1583–1592. Wiley Online Library, 2011.

Gabriel Peyré and Marco Cuturi.

Computational optimal transport.

arXiv preprint arXiv:1803.00567, 2018.

References viii

Anthony Prieur.

Simulation de la formation des structures de l'univers.

https://github.com/devpack/nbody-cosmos, 2011.

Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang.

The power particle-in-cell method.

ACM Transactions on Graphics, 41(4), 2022.

References ix

Ziyin Qu, Minchen Li, Yin Yang, Chenfanfu Jiang, and Fernando De Goes.

Power plastics: A hybrid Lagrangian/Eulerian solver for mesoscale inelastic flows.

ACM Transactions on Graphics (TOG), 42(6):1–11, 2023.

Bernhard Schmitzer.

Stabilized sparse scaling algorithms for entropy regularized transport problems.

SIAM Journal on Scientific Computing, 41(3):A1443–A1481, 2019.

References x

Anna Song.

Generation of tubular and membranous shape textures with curvature functionals.

Journal of Mathematical Imaging and Vision, 64(1):17–40, 2022.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle.

A material point method for snow simulation.

ACM Transactions on Graphics (TOG), 32(4):1–10, 2013.