Advances in the Material Point Method:
Moving Least Squares Discretization, Compatible Particle-in-Cell Transfer, and Asynchronous Time Integration

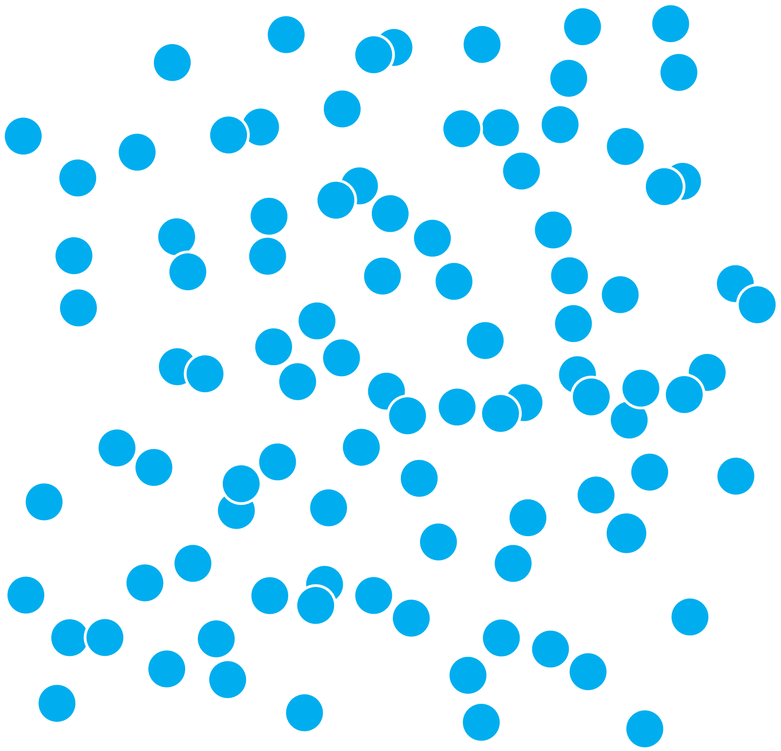
Presented by
Yuanming Hu 胡渊鸣, MIT CSAIL

Overview

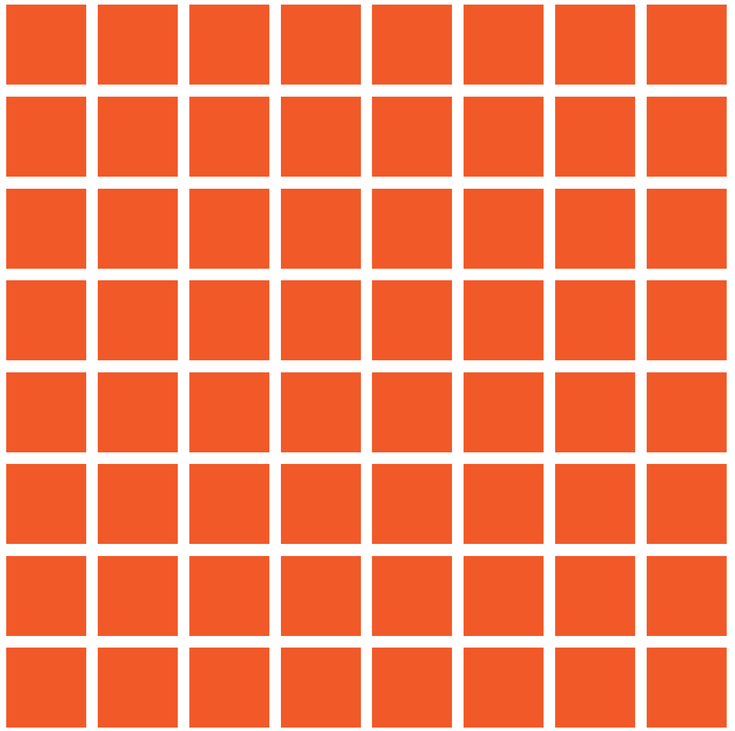
- **A Gentle Introduction to the Material Point Method (9 min)**
  - What is MPM and what are the current limitations?
- **Moving Least Squares Discretisation (MLS-MPM, 10 min)**
  - An easier to implement and optimise scheme
- **Compatible Particle-in-Cell (CPIC, 10 min)**
  - Velocity field discontinuity
  - Enables cutting and rigid body coupling
- **High-Performance Implementation (10 min)**
  - Parallelism and locality
  - Asynchronous time integration (AsyncMPM)

Part I:
A Gentle Introduction to the Material Point Method

Deformable Body Simulation

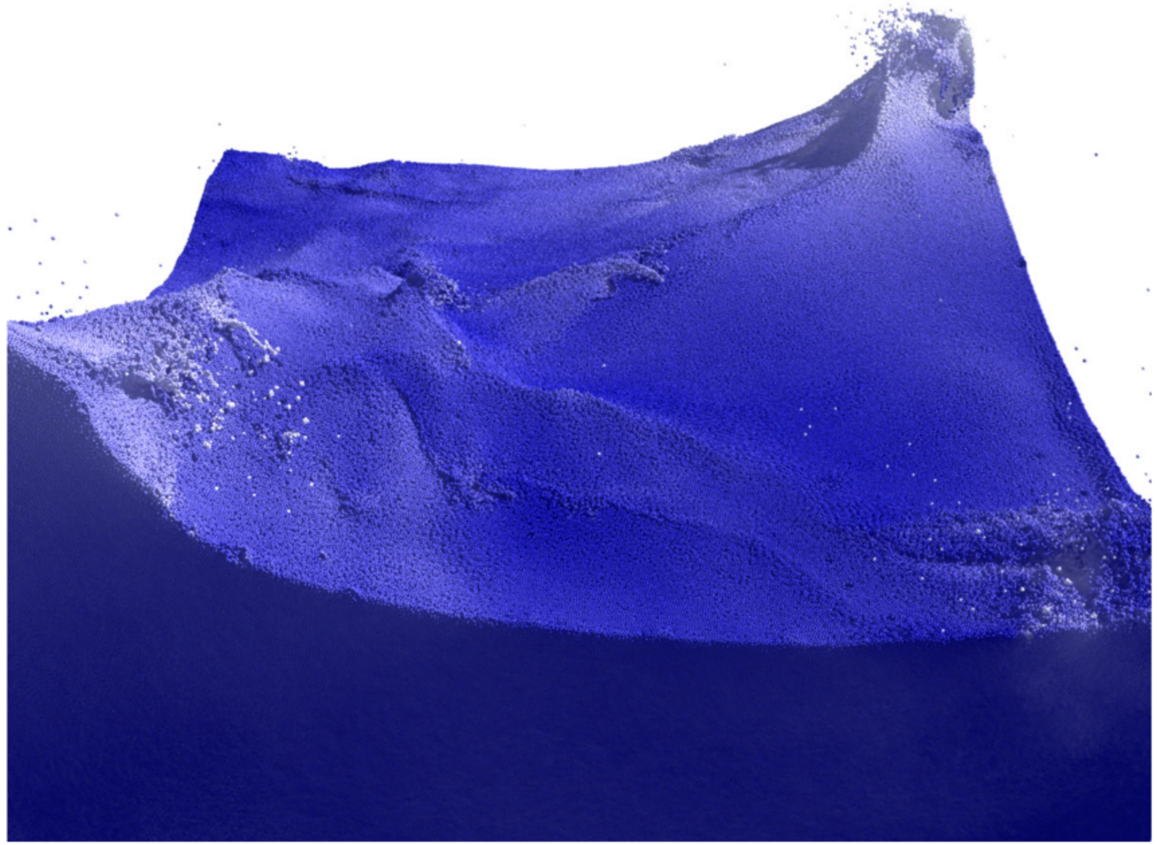


Lagrangian representation



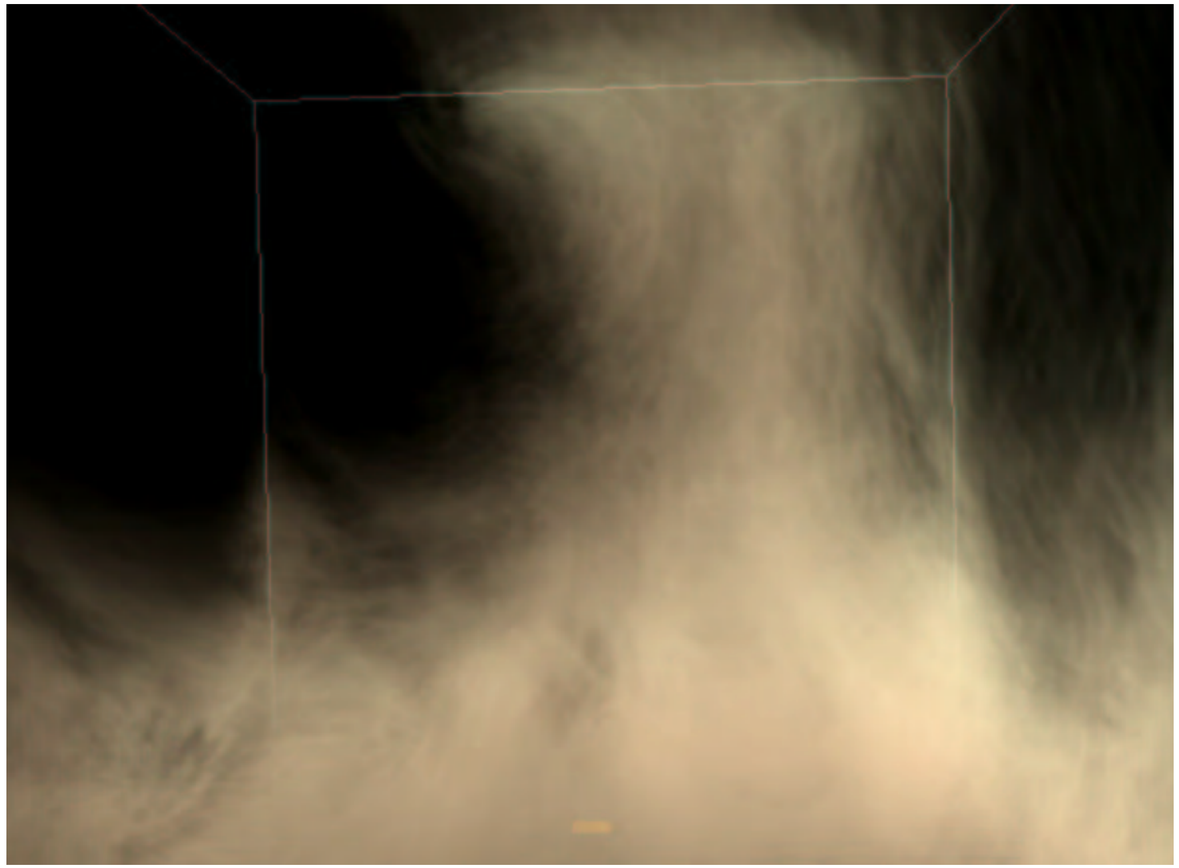
Eulerian representation

Deformable Body Simulation



Lagrangian representation

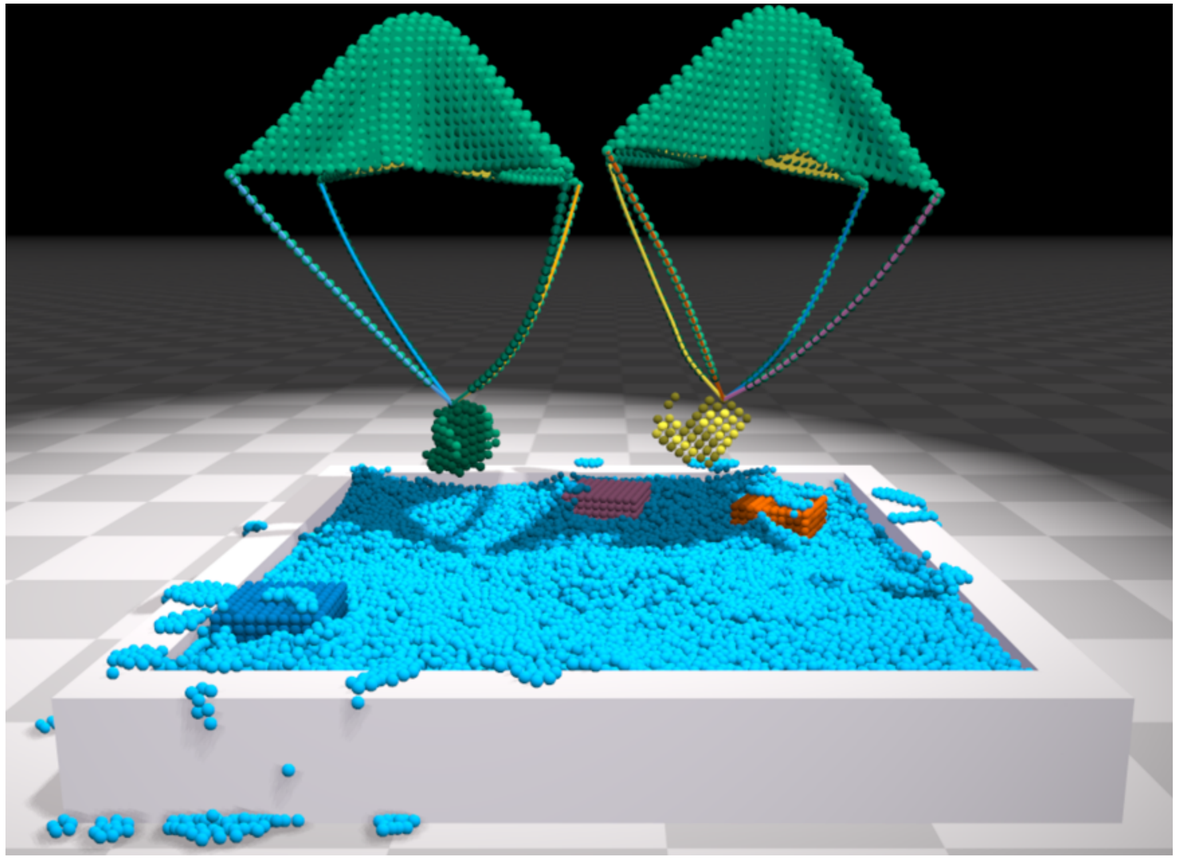
[Markus et al. 2014,
SPH Fluids in Computer Graphics]



Eulerian representation

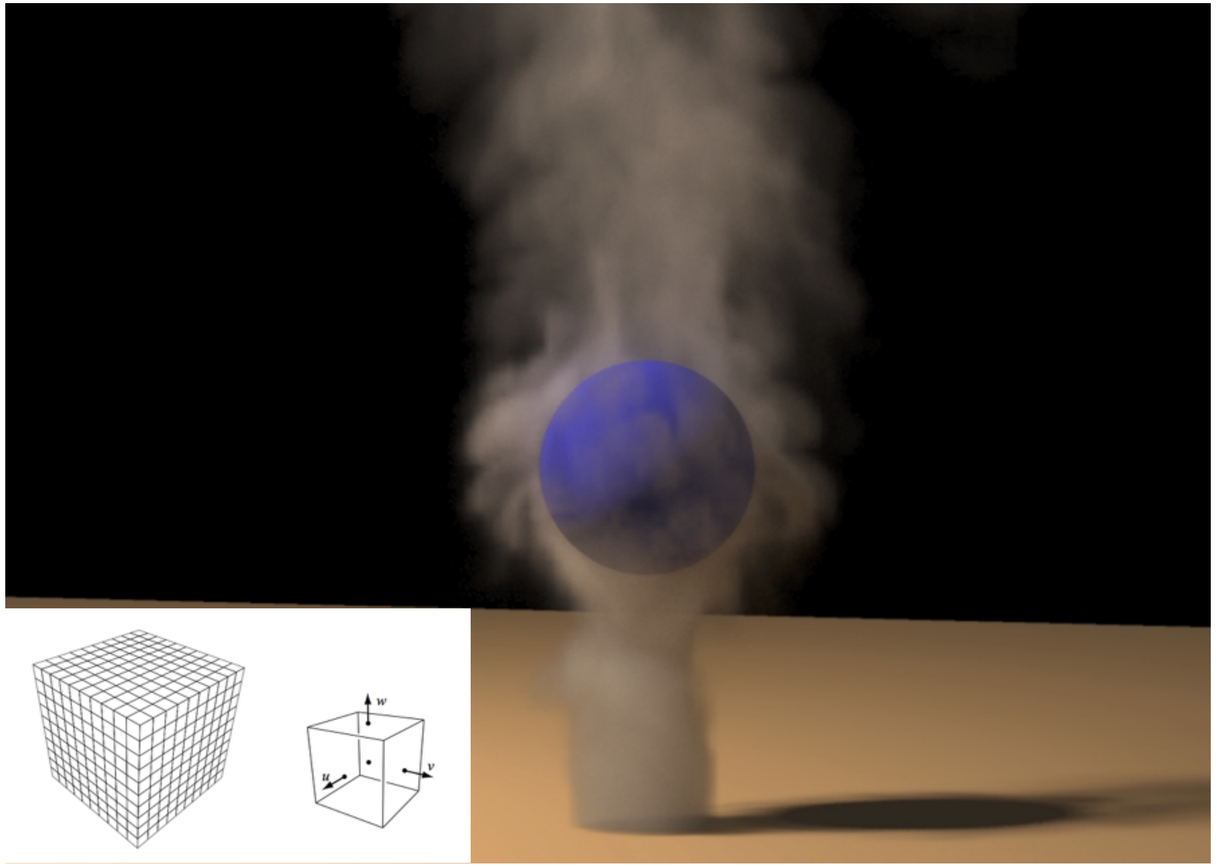
[Stam 1999, Stable Fluids]

Deformable Body Simulation



Lagrangian representation

[Macklin et al. 2014,
Unified Particle Physics for Real-Time Applications]



Eulerian representation

[Fedkiw 2001, Visual Simulation of Smoke]

The Material Point Method (MPM)

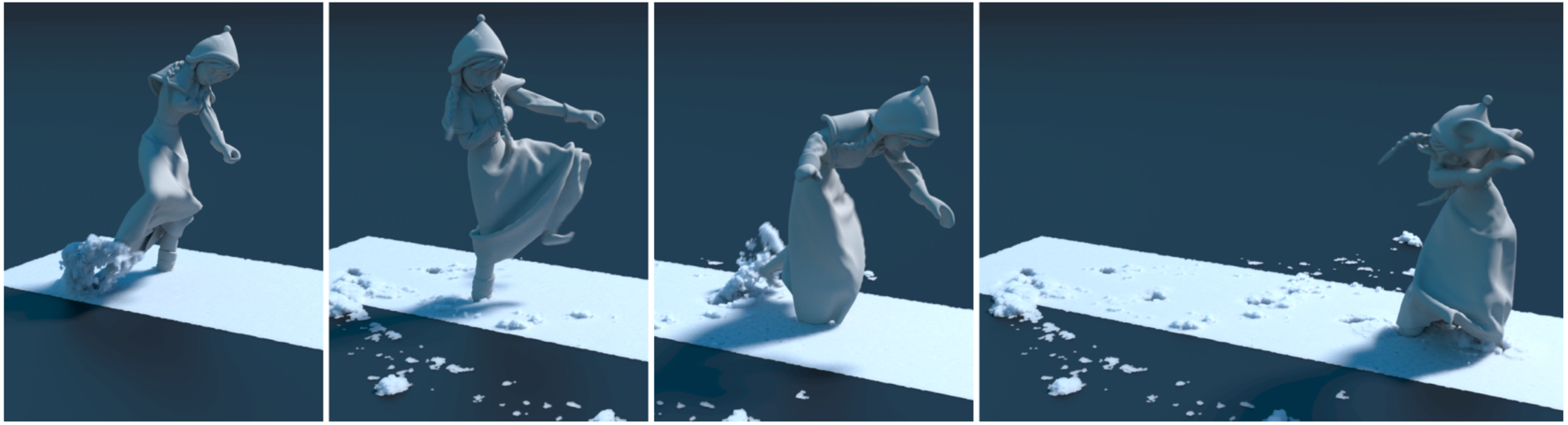
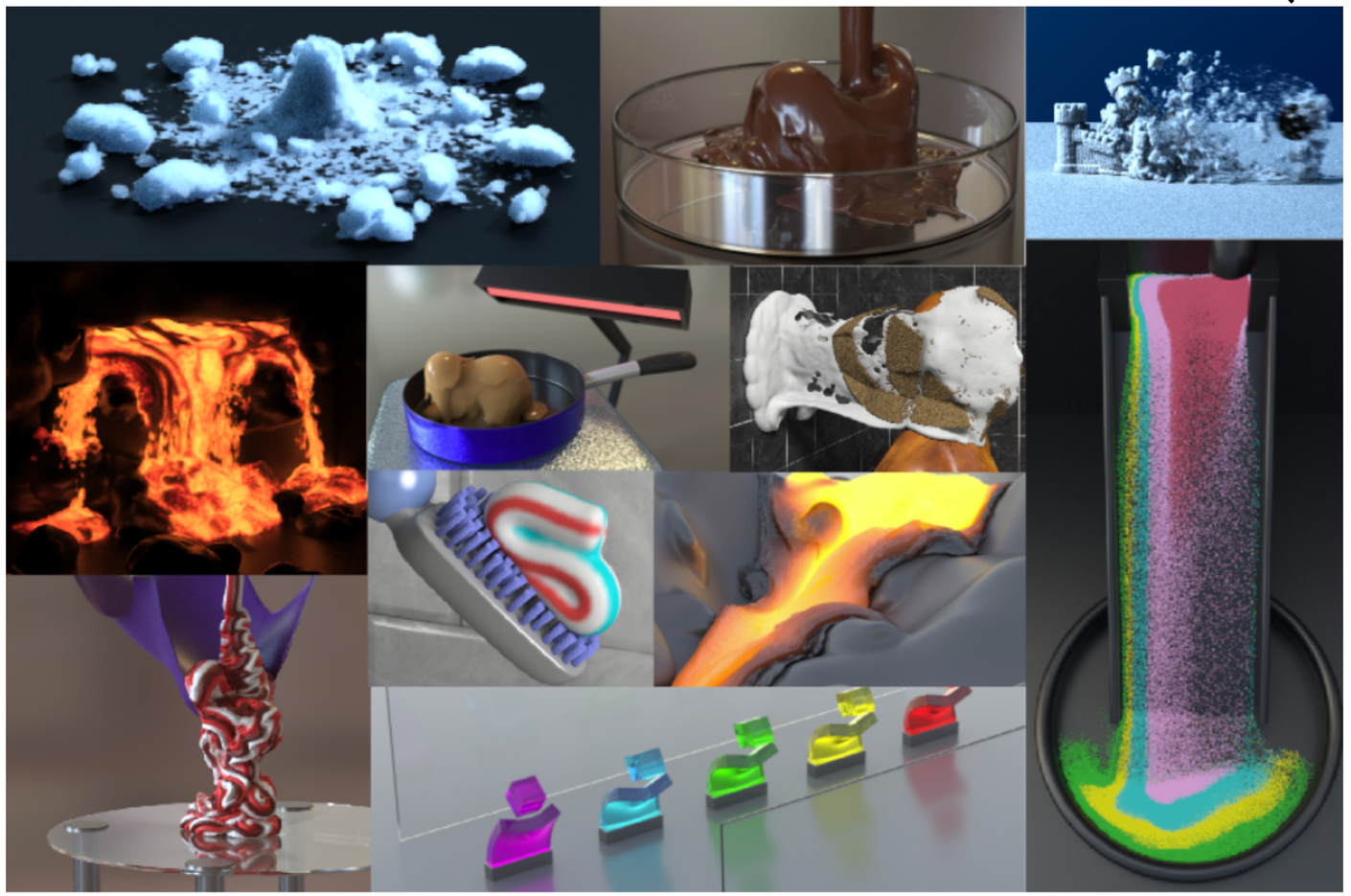


Figure 9: Walking character. *A character stepping through snow produces an interesting structure.* ©*Disney.*

Hybrid Eulerian/Lagrangian

[Stomakhin et al. 2013, A material point method for snow simulation]

The Material Point Method (MPM)



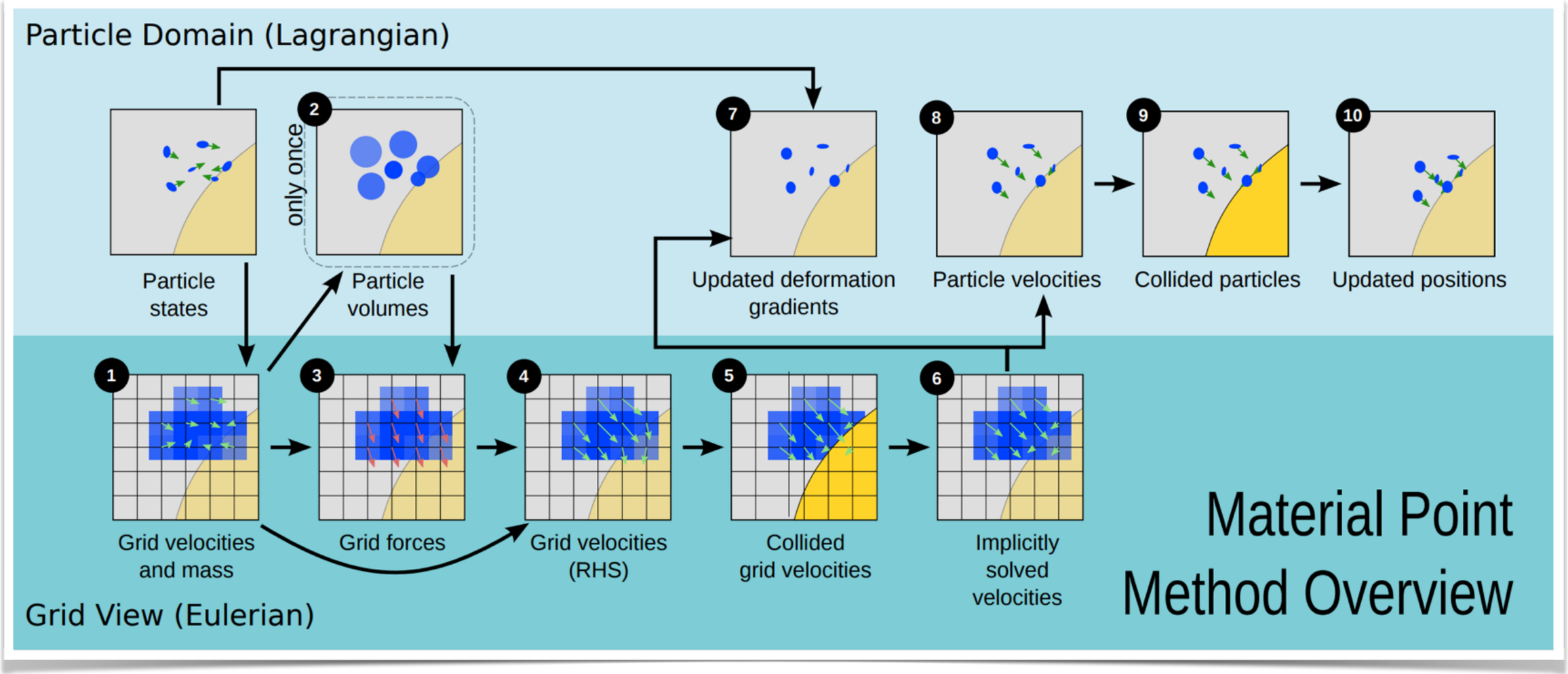
Hybrid Eulerian/Lagrangian

[Jiang et al. 2016,
The Material Point Method for Simulating Continuum Materials]

The Material Point Method (MPM)

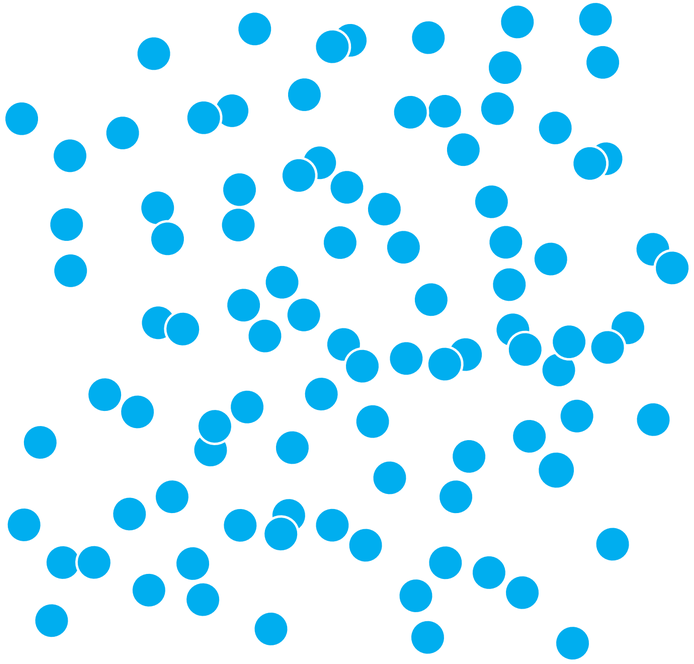
- MPM is a recently popular method for physical simulation in graphics
- It is popular because of …
  - Automatic coupling of different materials (liquids, solids, granular materials etc.)
  - Automatic (self-)collision handling
  - Automatic fracture
  - Capable of simulating large deformations
- Hybrid Lagrangian-Eulerian: both a **grid** and **particles** are used
  - An Eulerian grid is used for collision handling and momentum update
  - Lagrangian particles are used for state tracking such as advection and deformation

The Material Point Method (MPM)



Stomaching et al., A material point method for snow simulation, SIGGRAPH 2013

The Material Point Method (MPM)



Particle to Grid (P2G)

Grid to Particle (G2P)

Transfer (Particle in Cell, PIC)

Affine PIC, APIC [Jiang et al. 2016]

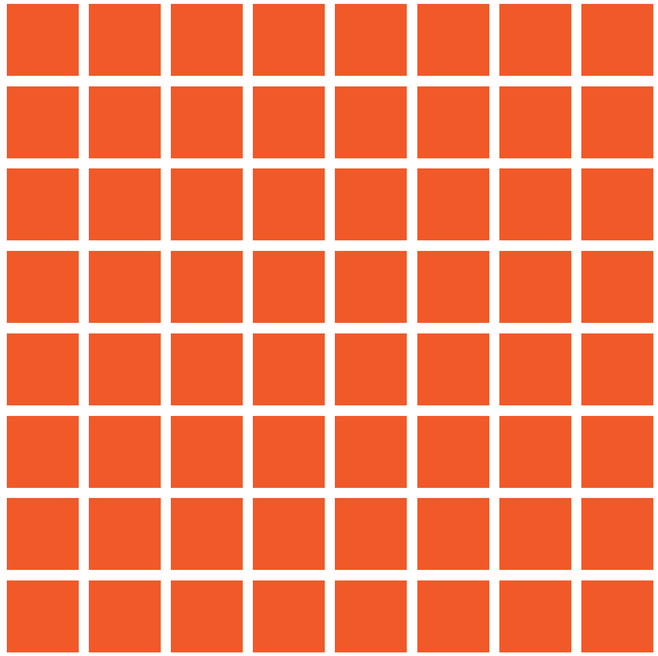
Polynomial PIC, PolyPIC [Fu et al. 2017]

High-performance GIMP [Gao et al. 2017]

Moving Least Squares [Hu et al. 2018]

Compatible PIC [Hu et al. 2018]

...



Particles (Constitutive models)

Snow [Stomakhin et al. 2013],

Foam [Ram et al. 2015, Yue et al 2015.]

Sand [Klar et al. 2015, Pradhana et al 2017]

Grid

SPGrid [Setaluri et al. 2014],

OpenVDB [Museth 2013]

Multiple Grids

[Pradhana et al. 2015]

Part II & III:

A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling

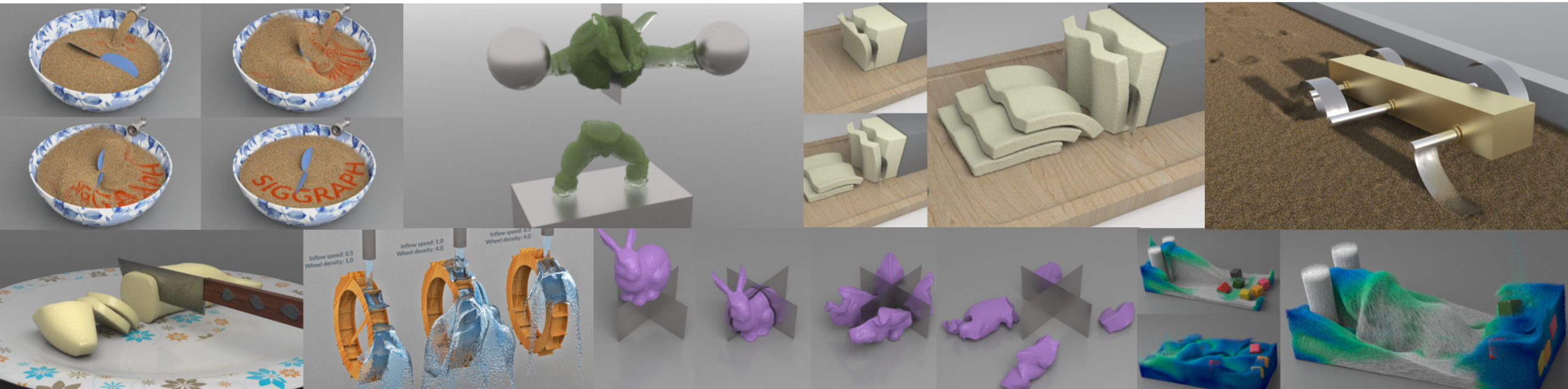
SIGGRAPH 2018

Yuanming Hu[1] Yu Fang[2] Ziheng Ge[3] Ziyin Qu[4] Yixin Zhu[5]

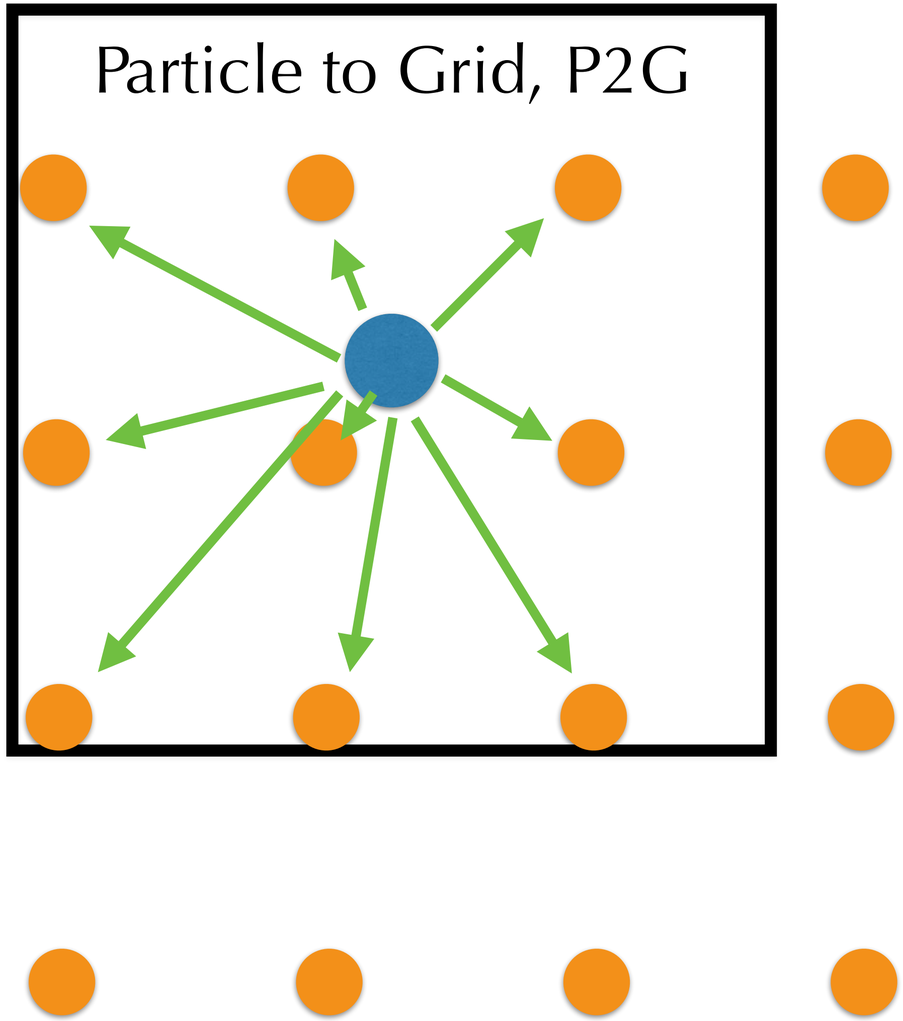
Andre Pradhana[4] Chenfanfu Jiang[4]

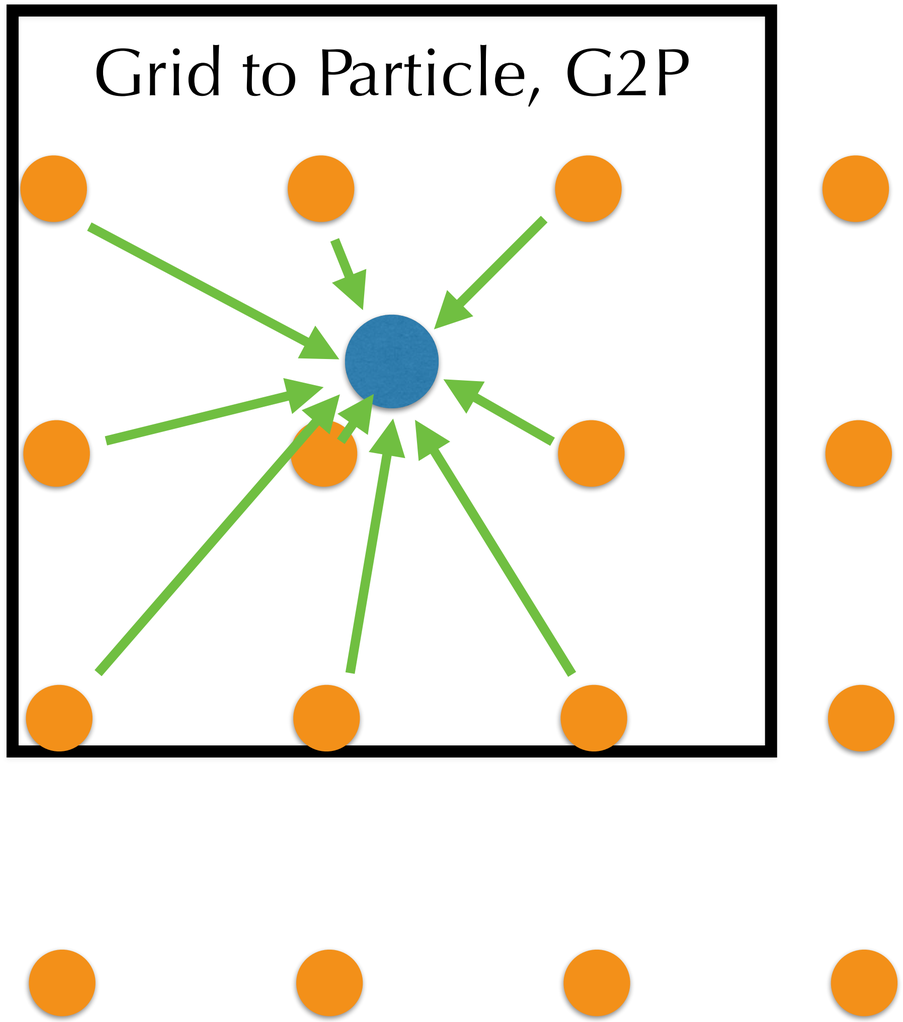
[1]MIT CSAIL [2]Tsinghua University [3]University of Science and Technology of China

[4]University of Pennsylvania [5]UCLA



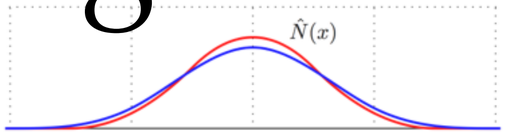
The “Transfer” Problem

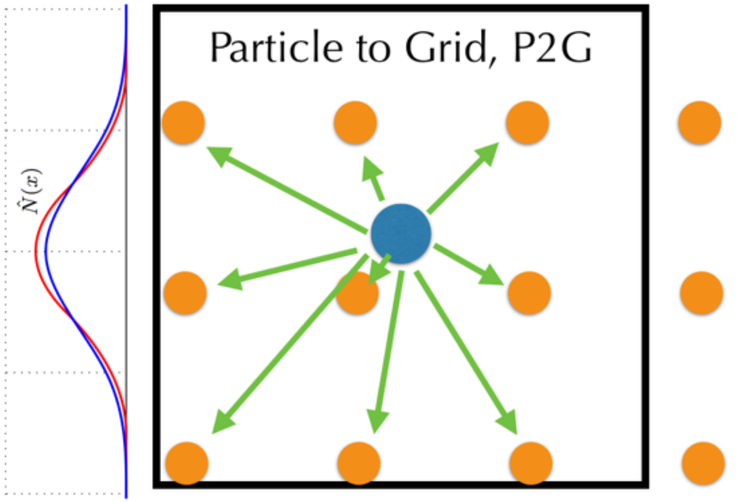


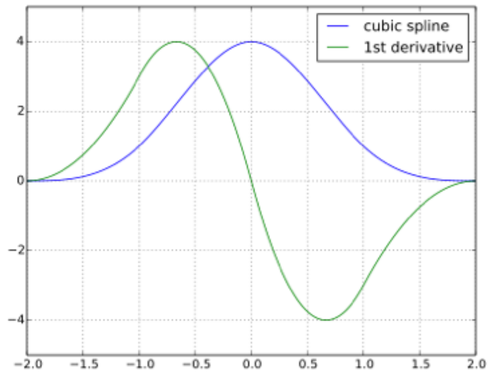


The “Transfer” Problem - Insights

- **Information flow**
  - Particle->Grid->Particle, information loss
  - Dissipation (damping)
  - Mitigated by FLIP, **APIC, PolyPIC** (will introduce later)
- **Performance & Complexity: each particle touches 27 nodes in 3D and 9 nodes in 2D**
  - Accelerated by MLS-MPM (will introduce later)
  - Needs kernel gradient, slow and complex (solved by MLS)
  - P2G has data race when parallelized
- **“Fuzzy” kernel leads to blurring**
  - Smooths particle velocity field
  - Fixed by CPIC [Hu et al. 2018] (will introduce later)

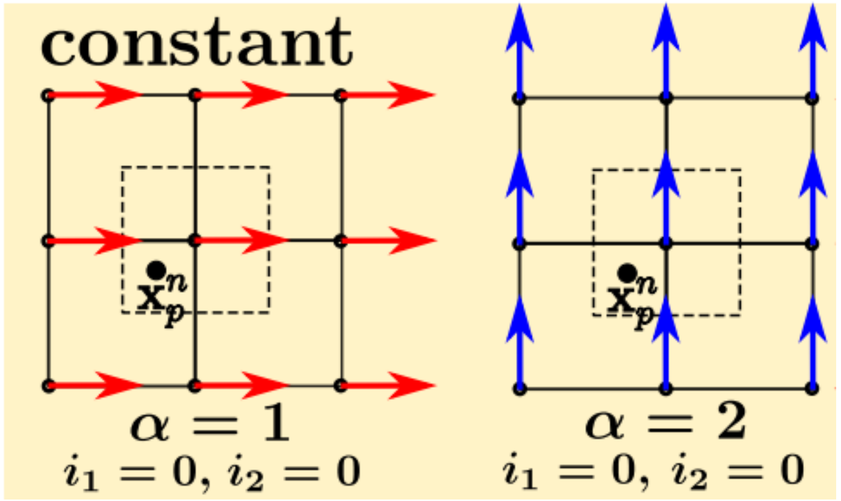






The Dissipation Problem, G2P

Traditional Particle-in-Cell



Problem: 18 DoFs on grid, 2 DoFs on particle

Figure from Fu et al 2017, A Polynomial Particle-In-Cell Method

The Dissipation Problem

The Affine Particle-in-Cell, Jiang et al. 2016

Gather more information during G2P

Preserves angular momentum

Locally preserves affine velocity fields

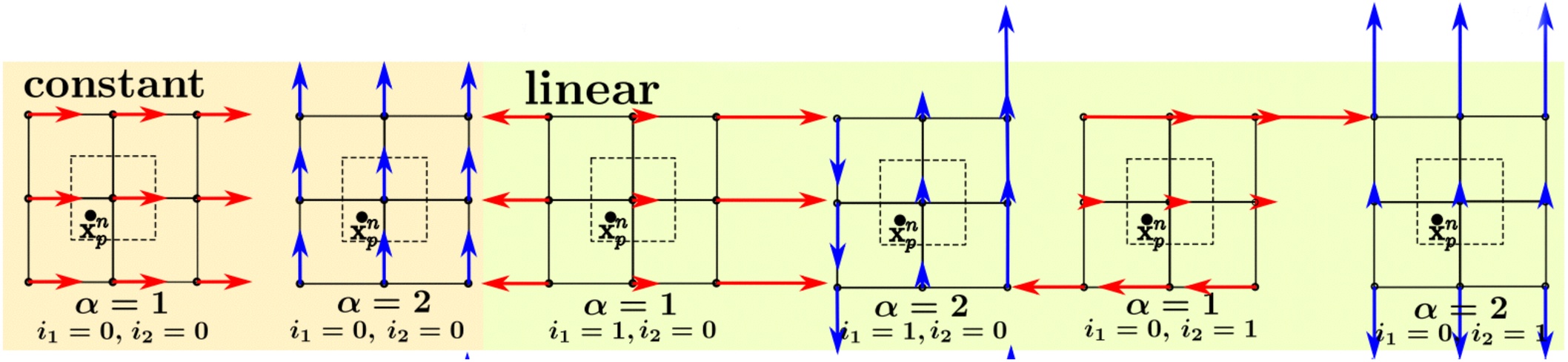
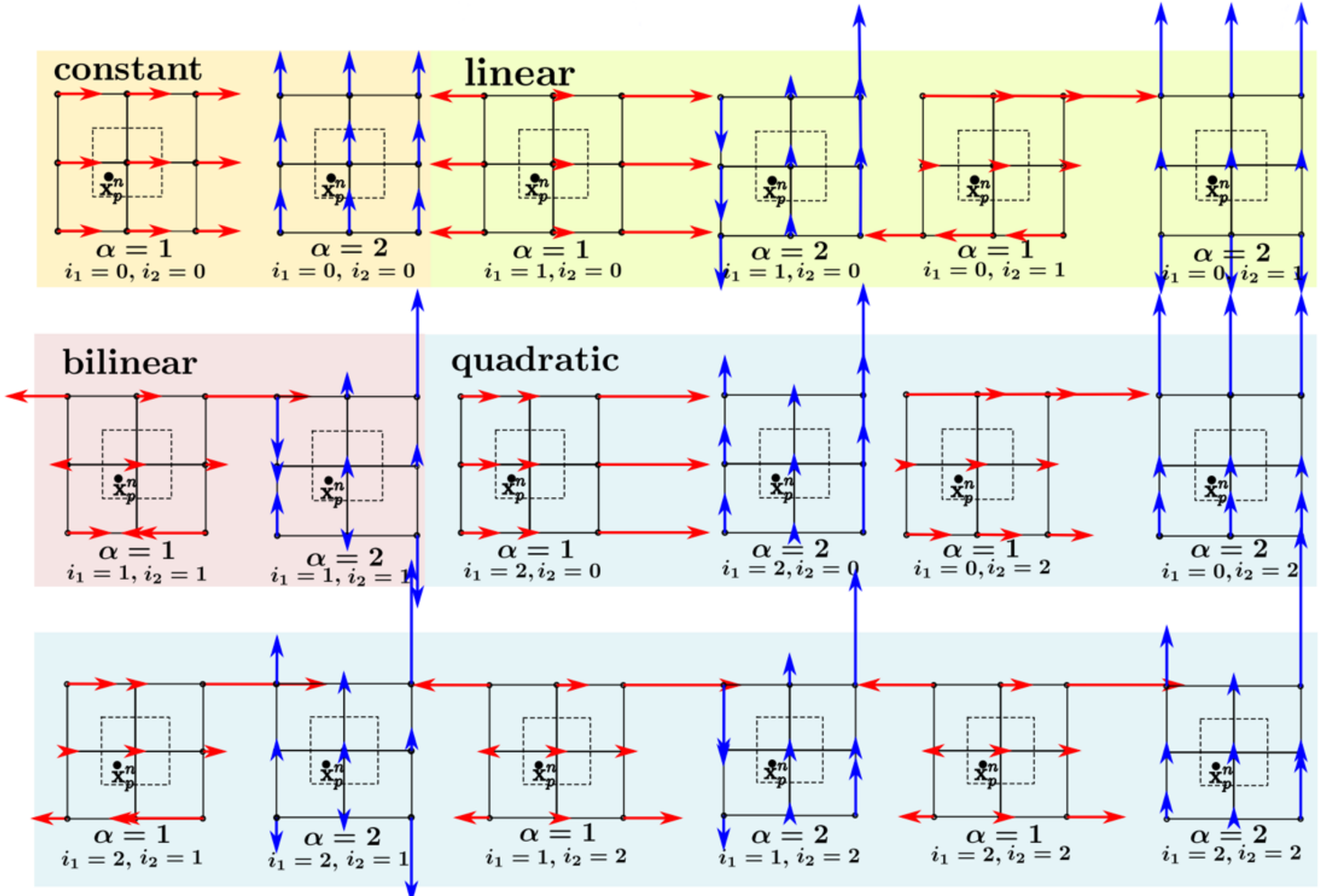


Figure from Fu et al 2017, A Polynomial Particle-In-Cell Method

The Dissipation Problem

A Polynomial Particle-In-Cell Method, Fu et al. 2017

18 modes=9 nodes X 2 DoFs per node: Lossless transfer!



The Dissipation Problem

- **In graphics, APIC has almost been one of the most popular MPM transfer schemes**
  - APIC is easy to implement
  - APIC is storage friendly (no backup velocity field compared with FLIP)
  - APIC is stable
  - APIC leads to good visual results

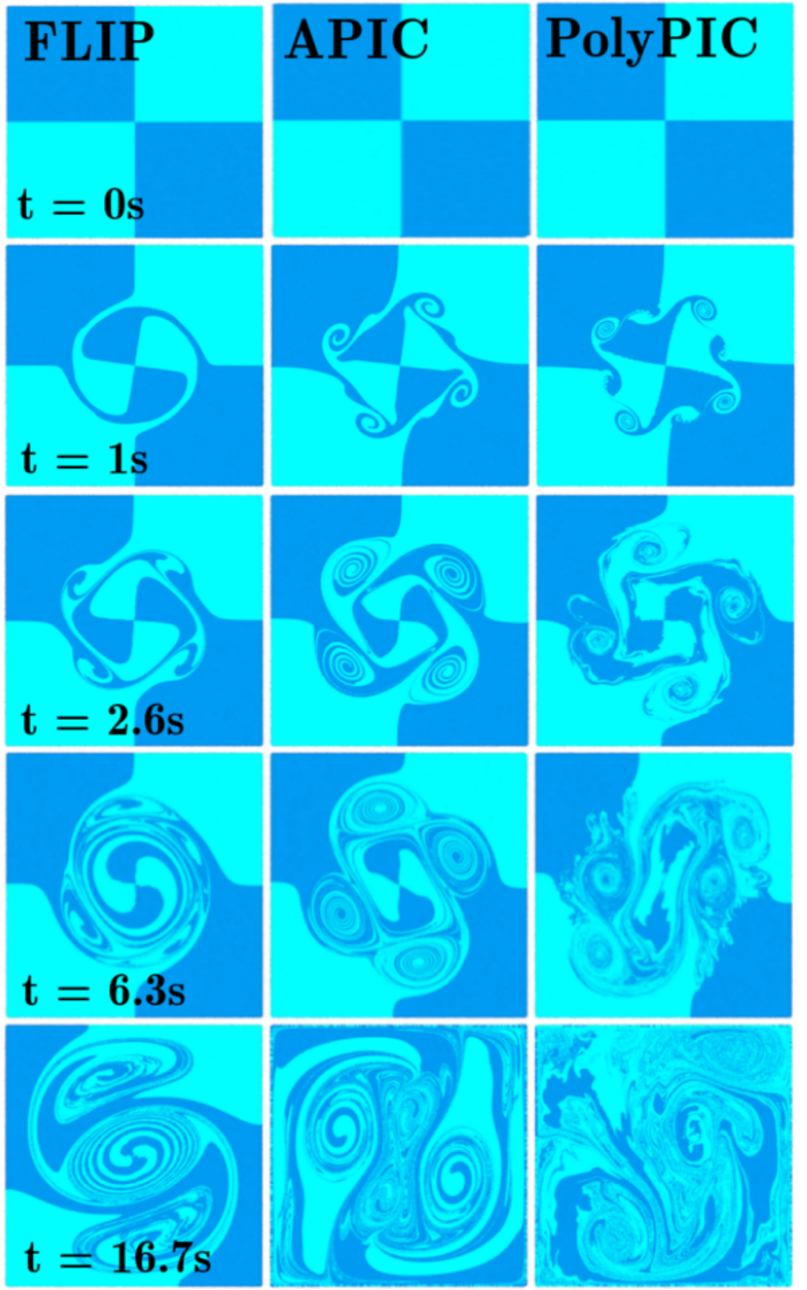


Figure from Fu et al 2017, A Polynomial Particle-In-Cell Method

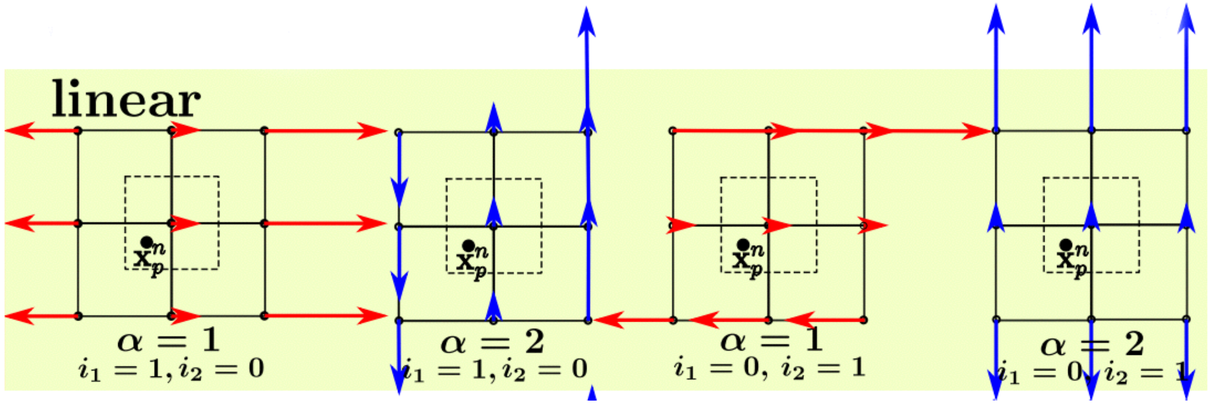
Deformation Gradient Update

- **Deformation (displacement) gradients**
  - Local description of accumulated deformation in history
  - I.e. the relationship between the rest state and deformed state, as a 3x3 matrix (in 3D)
- **Deformation gradient updates**
  - Use **velocity gradient** $\nabla \boldsymbol{v}_p^{n+1} = \sum_i \boldsymbol{v}_i^{n+1} (\nabla w_{ip}^n)^T$

$$\boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1}) \boldsymbol{F}_p^n$$

- **"Fuzzy" kernel leads to blurring**
  - **Smooths** particle velocity field
  - Fixed by CPIC [Hu et al. 2018] (will introduce later)

APIC:
Local affine velocity field
(Essentially moving least squares reconstruction)



Deformation Gradient Update:
Velocity gradient (finite difference)

Question:
Can we use the local velocity field as the velocity gradient?

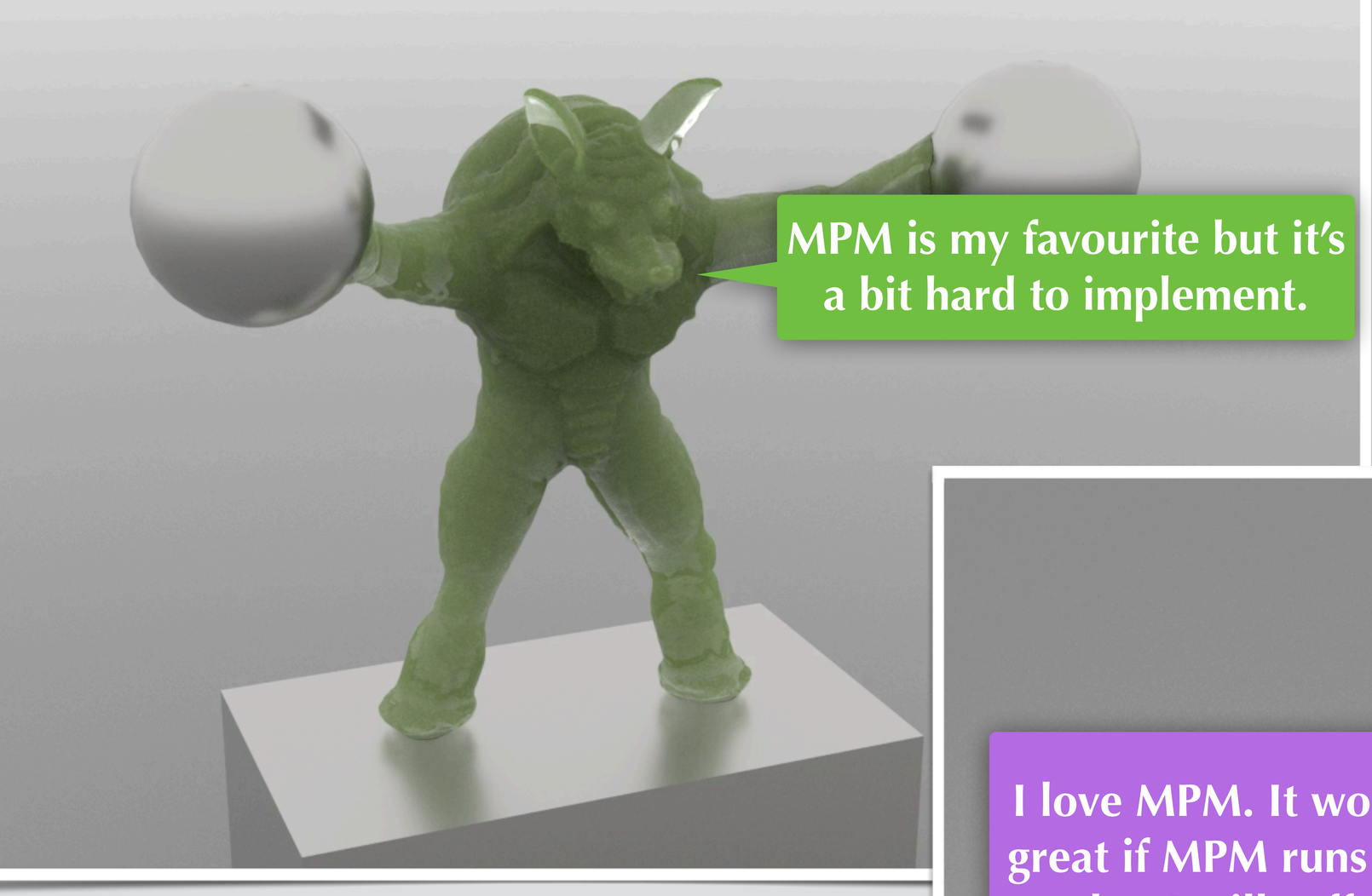
Answer:
Yes!
But we need a bit more modification to MPM.
Fortunately, these modification actually simplifies and accelerates MPM.

Intuitive Changes

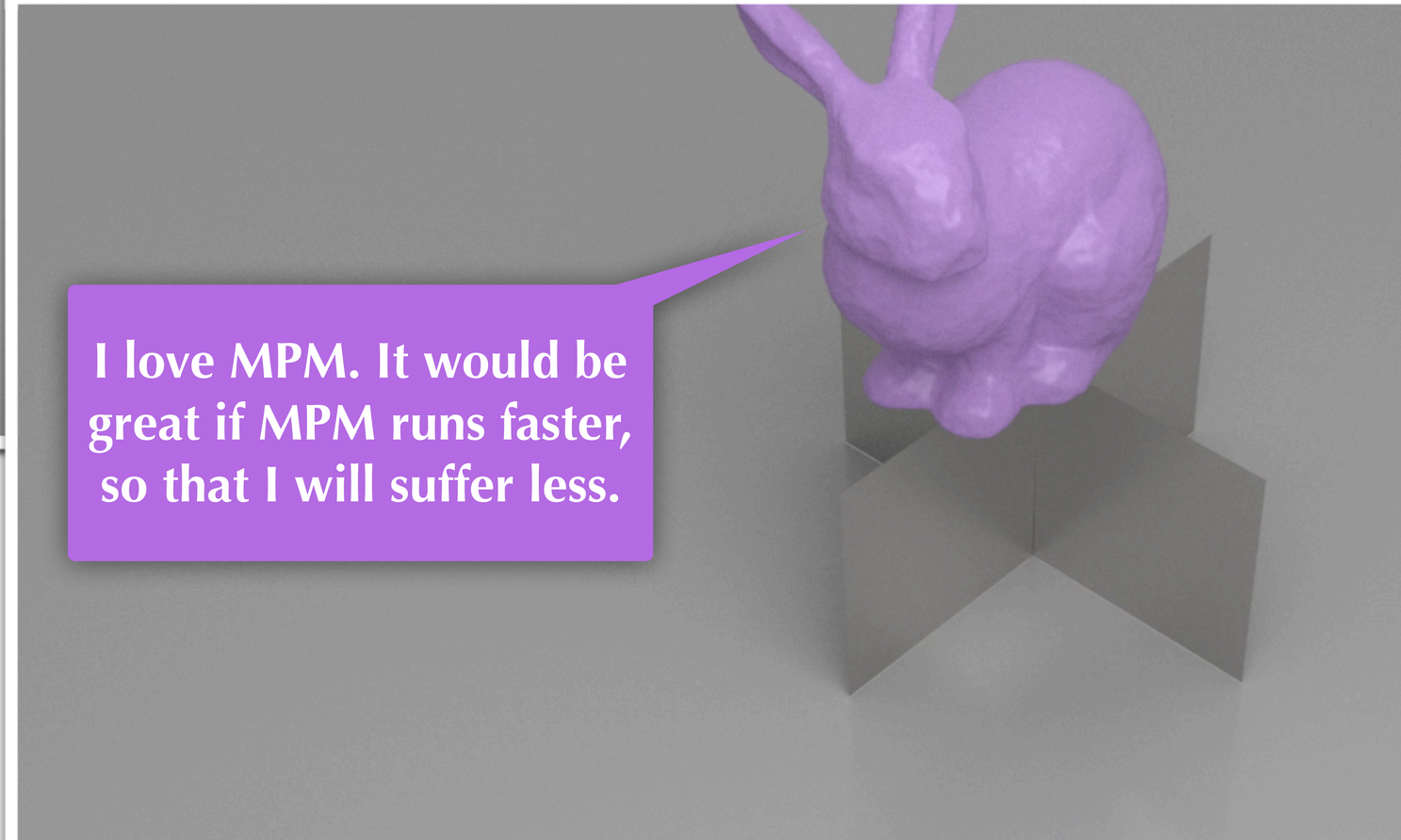
- The velocity field is defined using moving least squares interpolation
  - 0-th derivative (i.e. the velocity) stays the same as the original MPM
  - 1-th derivative (i.e. the velocity gradient) changed
    - Consistent with the APIC local affine velocity field!

$$\nabla \boldsymbol{v}_p^{n+1} = \sum_i \boldsymbol{v}_i^{n+1} (\nabla w_{ip}^n)^T \qquad \nabla \boldsymbol{v}_p^{n+1} = \frac{\partial \hat{\mathbf{v}}^{n+1}}{\partial \mathbf{x}} = \mathbf{C}_p^{n+1} \quad \text{and} \quad \mathbf{F}_p^{n+1} = \left(\mathbf{I} + \Delta t \mathbf{C}_p^{n+1}\right) \mathbf{F}_p^n$$

- Actually, MLS-MPM is a special **Element-Free Galerkin Method** [Belytschko et al. 1994]
- More details in the paper



Question: Do you like the Material Point Method?



`#include “taichi.h”`

Affine Particle-in-Cell

Material Point Method

Moving Least Squares

MLS-MPM
faster & easier

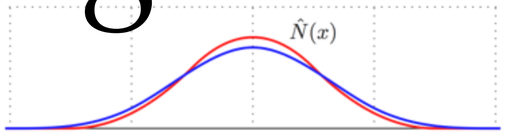
Implement Interactive MLS-MPM within

88 lines of code (comments included)!

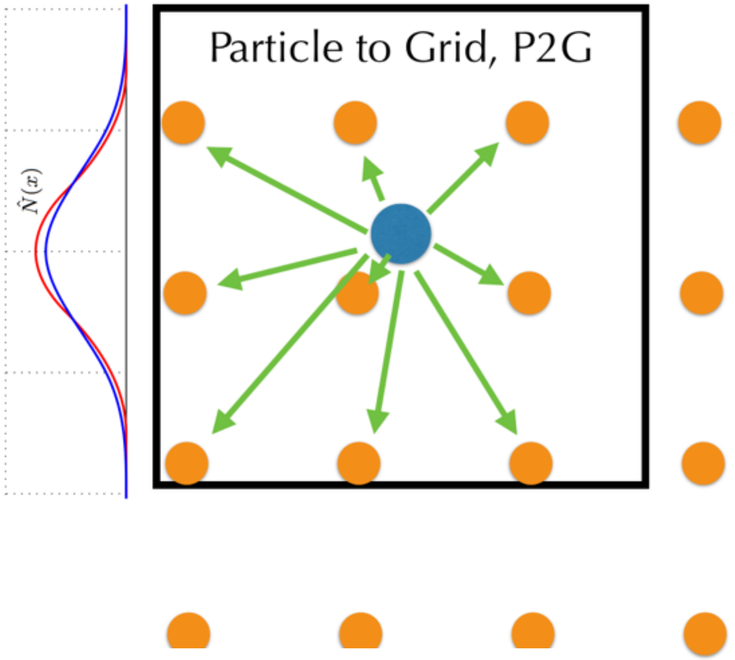
```cpp
// The Moving Least Squares Material Point Method in 88 LoC (with comments)
//   To compile:   g++ mpm.cpp -std=c++14 -g -lX11 -lpthread -O2 -o mpm
#include "taichi.h" // Single header version of (a small part of) taichi
using namespace taichi;
const int n = 64 /*grid resolution (cells)*/, window_size = 500;
const real dt = 1e-4_f, frame_dt = 1e-3_f, dx = 1.0_f / n, inv_dx = 1.0_f / dx;
real mass = 1.0_f, vol = 1.0_f; // Particle mass and volume
real hardening = 10, E = 1e4 /* Young's Modulus*/, nu = 0.2 /*Poisson's Ratio*/;
real mu_0 = E/(2*(1+nu)), lambda_0=E*nu/((1+nu)*(1-2*nu));    // Lame parameters
using Vec = Vector2; using Mat = Matrix2; //Handy abbriviations for lin. algebra
struct Particle {Vec x/*position*/, v/*velocity*/; Mat B/*affine momentum*/;
  Mat F/*elastic deformation grad.*/;   real Jp /*det(plastic def. grad.)*/;
  Particle(Vec x, Vec v=Vec(0)) : x(x), v(v), B(0), F(1), Jp(1) {} };
std::vector<Particle> particles; // Particle states
Vector3 grid[n + 1][n + 1];// velocity with mass, note that node res=cell res+1

void advance(real dt) {                  // Simulation
  std::memset(grid, 0, sizeof(grid));  // Reset grid
  for (auto &p : particles) {            // P2G
    Vector2i base_coord = (p.x*inv_dx-Vec(0.5_f)).cast<int>();
    Vec fx = p.x * inv_dx - base_coord.cast<real>();
    // Quadratic kernels, see http://mpm.graphics Formula (123)
    Vec w[3]{Vec(0.5) * sqr(Vec(1.5) - fx), Vec(0.75) - sqr(fx - Vec(1.0)),
             Vec(0.5) * sqr(fx - Vec(0.5))};
    auto e = std::exp(hardening * (1.0_f - p.Jp)), mu=mu_0*e, lambda=lambda_0*e;
    real J = determinant(p.F);             //Current volume
    Mat r, s; polar_decomp(p.F, r, s); //Polor decomp. for Fixed Corotated Model
    auto force =                           // Negative Cauchy stress times dt and inv_dx
        inv_dx*dt*vol*(2*mu * (p.F-r) * transposed(p.F) + lambda * (J-1) * J);
    for (int i = 0; i < 3; i++) for (int j = 0; j < 3; j++) { // Scatter to grid
      auto dpos = fx - Vec(i, j);
      Vector3 contrib(p.v * mass, mass);
      grid[base_coord.x + i][base_coord.y + j] +=
          w[i].x*w[j].y*(contrib+Vector3(4.0_f*(force+p.B*mass)*dpos));
    }
  }
  for(int i = 0; i <= n; i++) for(int j = 0; j <= n; j++) { //For all grid nodes
    auto &g = grid[i][j];
    if (g[2] > 0) {                                  // No need for epsilon here
      g /= g[2];                                     // Normalize by mass
      g += dt * Vector3(0, -100, 0);                 // Apply gravity
      real boundary=0.05,x=(real)i/n,y=real(j)/n;//boundary thickness,node coord
      if (x < boundary||x > 1-boundary||y > 1-boundary) g=Vector3(0);//Sticky BC
      if (y < boundary) g[1]=std::max(0.0_f, g[1]);              //"Separate" BC
    }   // "BC" stands for "boundary condition", which is applied to grid nodes
  }
  for (auto &p : particles) { // Grid to particle
    Vector2i base_coord = (p.x * inv_dx - Vec(0.5_f)).cast<int>();
    Vec fx = p.x * inv_dx - base_coord.cast<real>();
    Vec w[3]{Vec(0.5) * sqr(Vec(1.5) - fx), Vec(0.75) - sqr(fx - Vec(1.0)),
             Vec(0.5) * sqr(fx - Vec(0.5))};
    p.B = Mat(0); p.v = Vec(0);
    for (int i = 0; i < 3; i++) for (int j = 0; j < 3; j++) {
      auto dpos = fx - Vec(i, j),
           grid_v = Vec(grid[base_coord.x + i][base_coord.y + j]);
      auto weight = w[i].x * w[j].y;
      p.v += weight * grid_v;                                  //  Velocity
      p.B += Mat::outer_product(weight * grid_v, dpos);        //    APIC B
    }
    p.x += dt * p.v;                                           // Advection
    auto F = (Mat(1) - (4 * inv_dx * dt) * p.B) * p.F;       // MLS-MPM F-update
    Mat svd_u, sig, svd_v; svd(F, svd_u, sig, svd_v); // SVD for snow Plasticity
    for (int i = 0; i < 2; i++)    // See SIGGRAPH 2013: MPM for Snow Simulation
      sig[i][i] = clamp(sig[i][i], 1.0_f - 2.5e-2_f, 1.0_f + 7.5e-3_f);
    real oldJ = determinant(F); F = svd_u * sig * transposed(svd_v);
    real Jp_new = clamp(p.Jp * oldJ / determinant(F), 0.6_f, 20.0_f);
    p.Jp = Jp_new; p.F = F;
  }
}

void add_object(Vec center) {    // Seed particles
  for (int i = 0; i < 1000; i++) // Randomly sample 1000 particles in the square
    particles.push_back(Particle((Vec::rand()*2.0_f-Vec(1))*0.08_f+center)); }

int main() {
  GUI gui("Taichi Demo: Real-time MLS-MPM 2D ", window_size, window_size);
  add_object(Vec(0.5,0.4));add_object(Vec(0.45,0.6));add_object(Vec(0.55,0.8));
  for (int i = 0;; i++) {                                  // Main Loop
    advance(dt);                                           // Advance simulation
    if (i % int(frame_dt / dt) == 0) {                     // Redraw frame
      gui.canvas->clear(Vector4(0.7, 0.4, 0.2, 1.0_f)); // Clear background
      for (auto p : particles)                             // Draw particles
        gui.buffer[(p.x * (inv_dx*window_size/n)).cast<int>()] = Vector4(0.8);
      gui.update();                                        // Update GUI
    }//Reference: A Moving Least Squares Material Point Method with Displacement
  } //            Discontinuity and Two-Way Rigid Body Coupling (SIGGRAPH 2018)
}  //  tDy Yuanming Hu (who also wrote this 88-line version), Yu Fang, Ziheng Ge,
//                      Ziyin Qu, Yixin Zhu, Andre Pradhana, Chenfanfu Jiang
```

Part III. The Compatible Particle-in-Cell Method

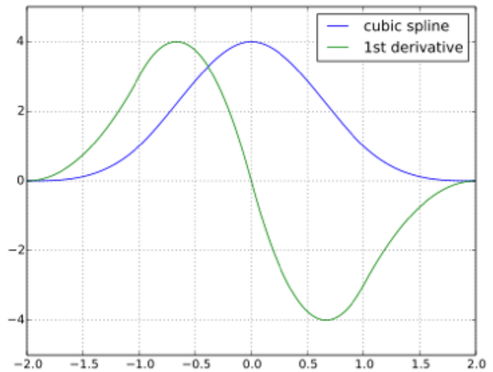
The “Transfer” Problem - Insights



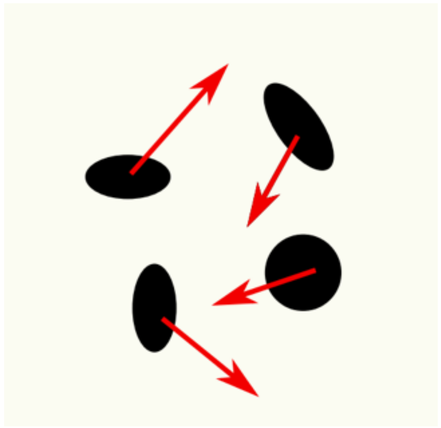
- **Information flow**
  - Particle->Grid->Particle, information loss
  - Dissipation (damping)
  - Mitigated by FLIP, **APIC, PolyPIC** (will introduce later)
- **Performance & Complexity: each particle touches 27 nodes in 3D and 9 nodes in 2D**
  - Accelerated by MLS-MPM (will introduce later)
  - Needs kernel gradient, slow and complex (solved by MLS)
  - P2G has data race when parallelized



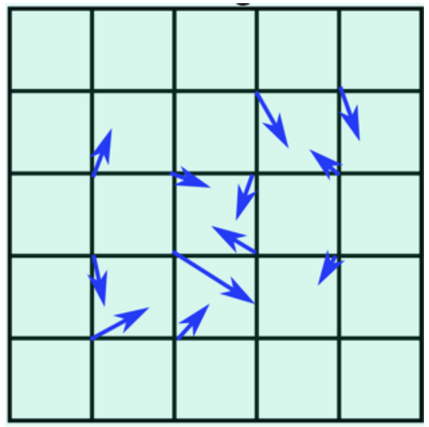
- **“Fuzzy” kernel leads to blurring**
  - Smooths particle velocity field
  - Fixed by CPIC [Hu et al. 2018] (will introduce later)



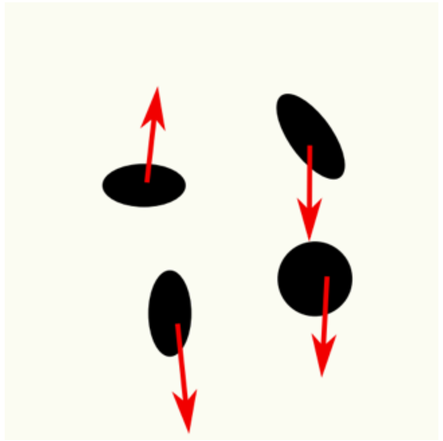
Advection



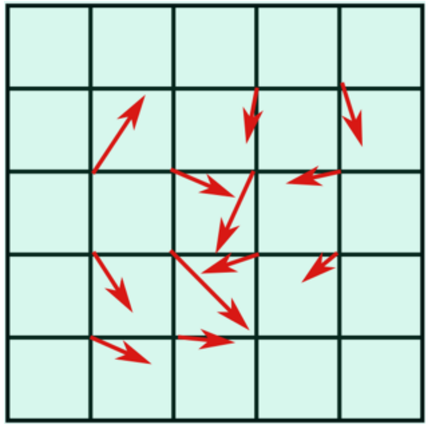
Rasterize



Grid Momentum Update

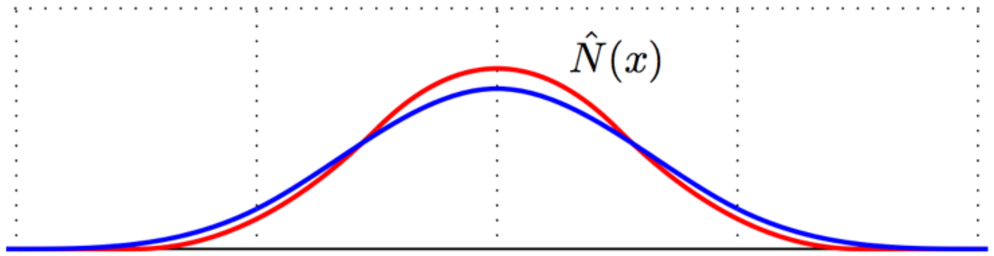


Resample



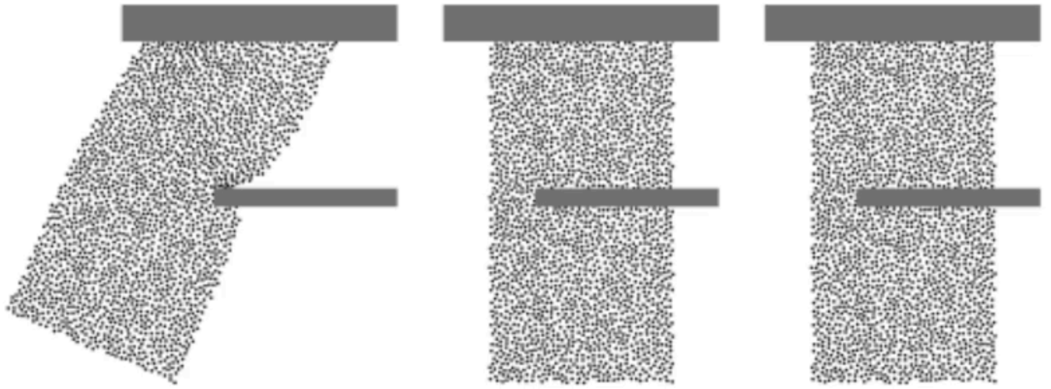
Advection

Traditional MPM Simulation Cycle



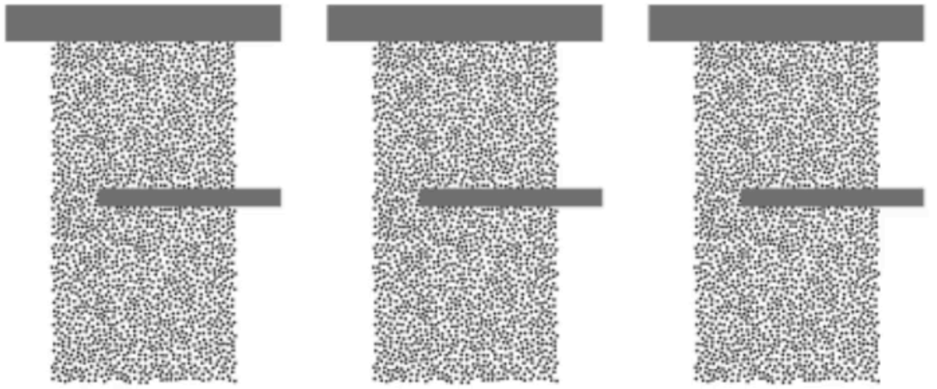
Particle-grid
transfer contribution weight

Level Set Cut



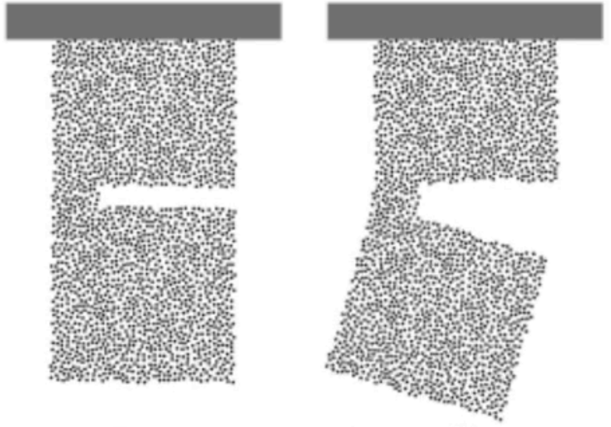
Sticky Slip Separate

Level Set Appear

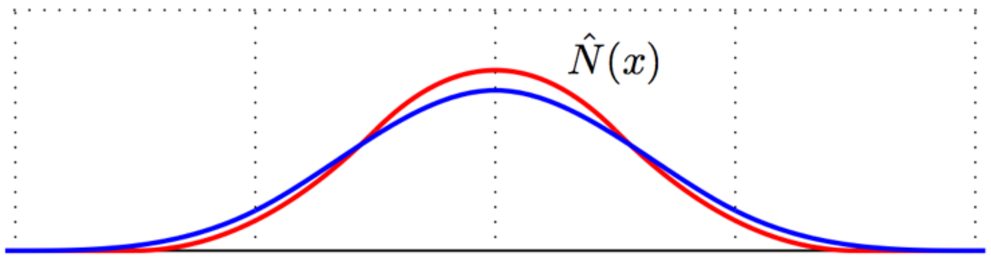


Sticky Slip Separate

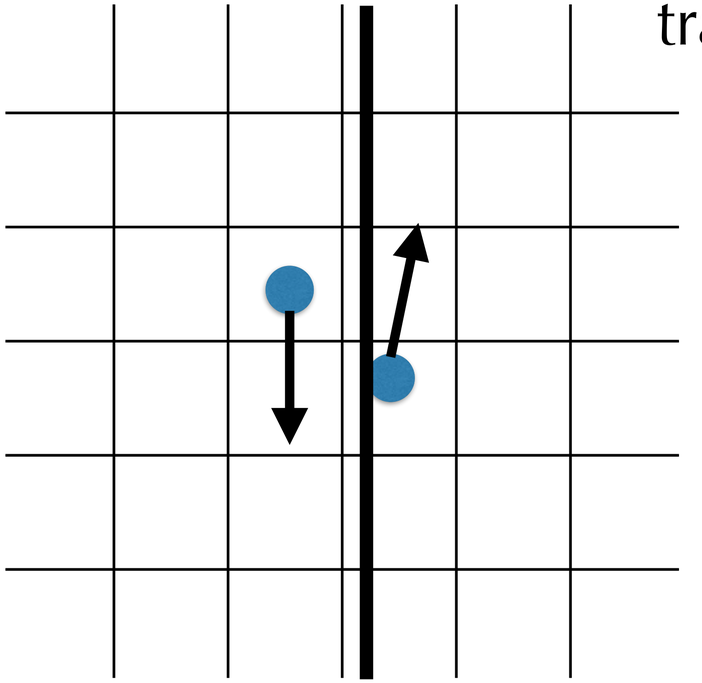
Traditional Method



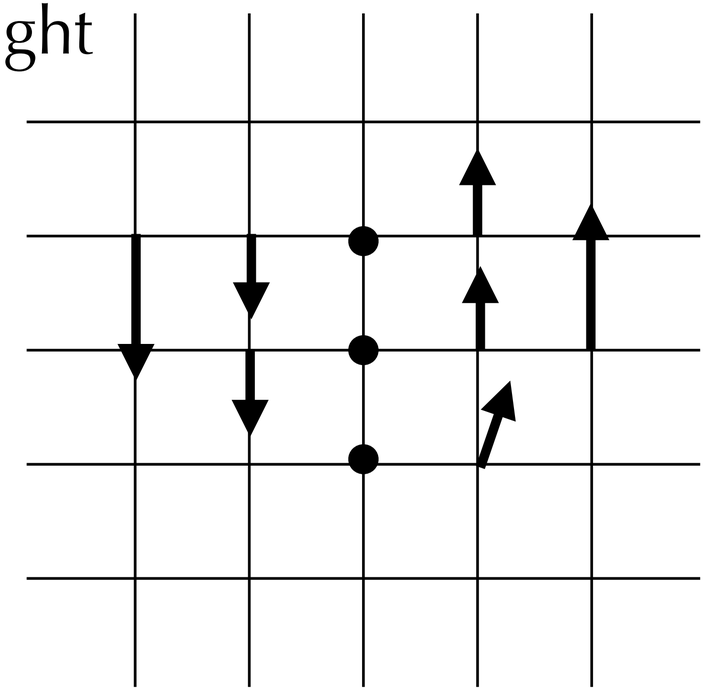
1dx Remove 1.5dx Remove

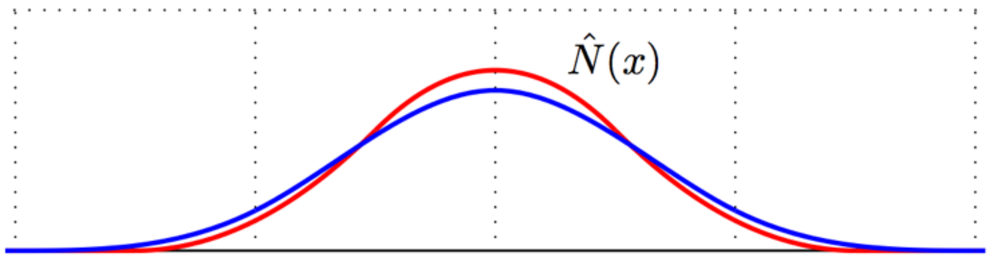


Particle-grid
transfer contribution weight

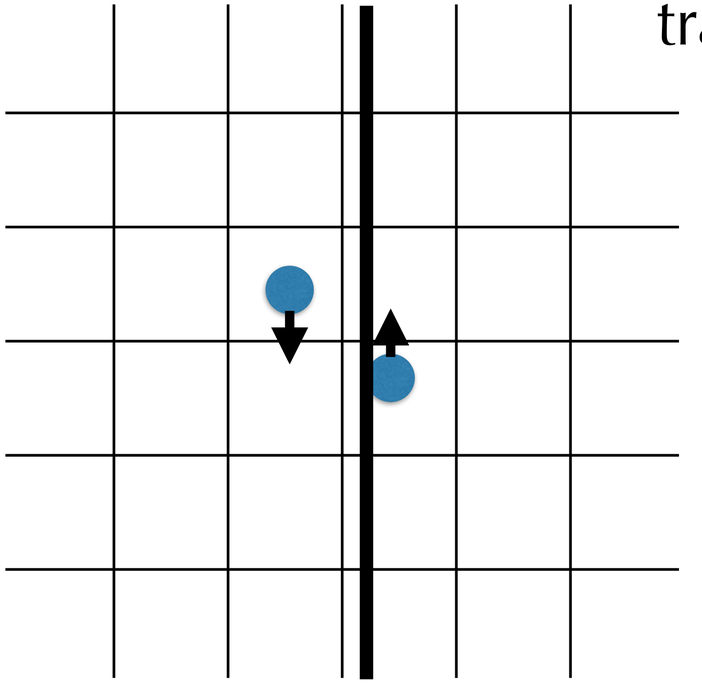


Rasterize

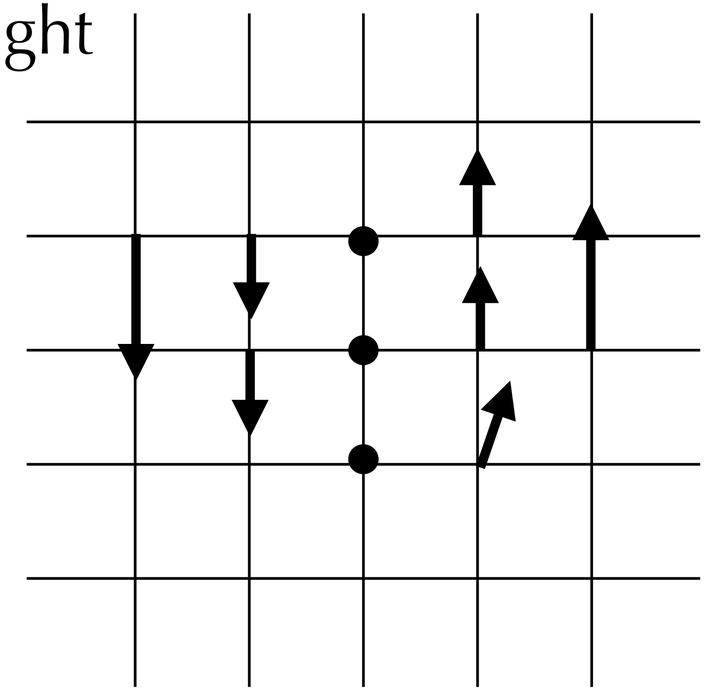




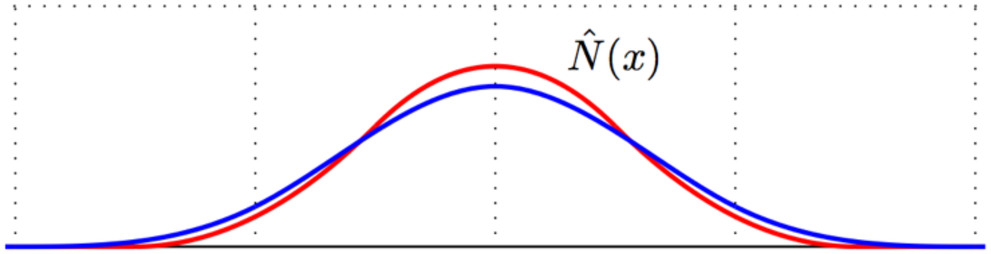
Particle-grid
transfer contribution weight



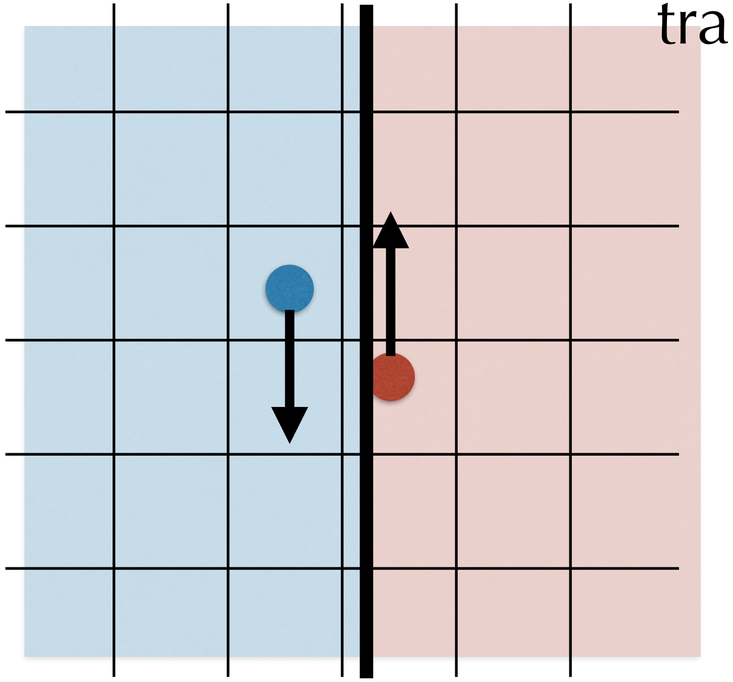
Resample



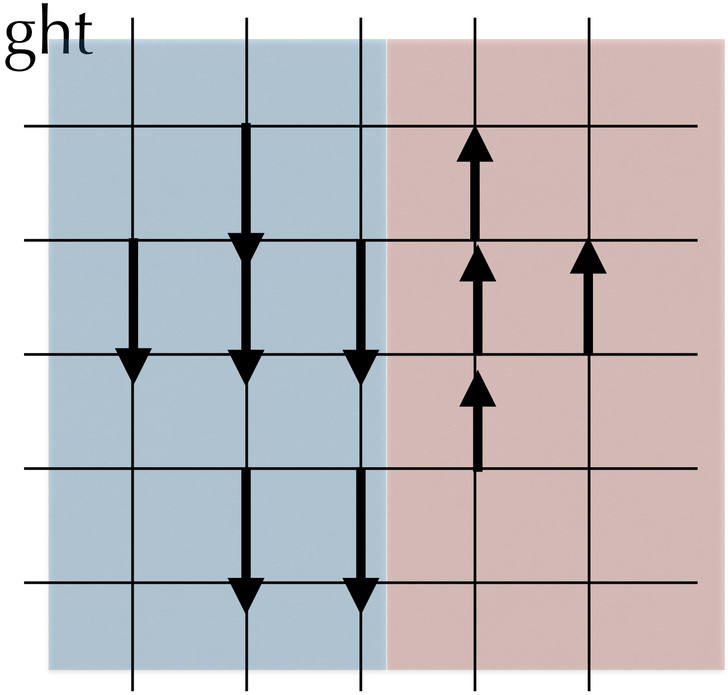
Velocity Discontinuity (Compatible Particle-in-Cell, CPIC)



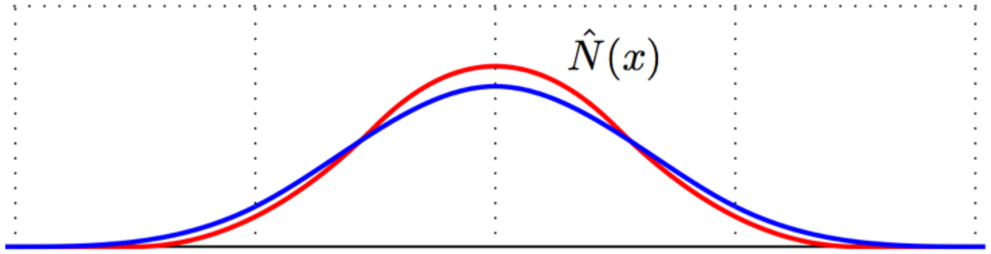
Particle-grid
transfer contribution weight



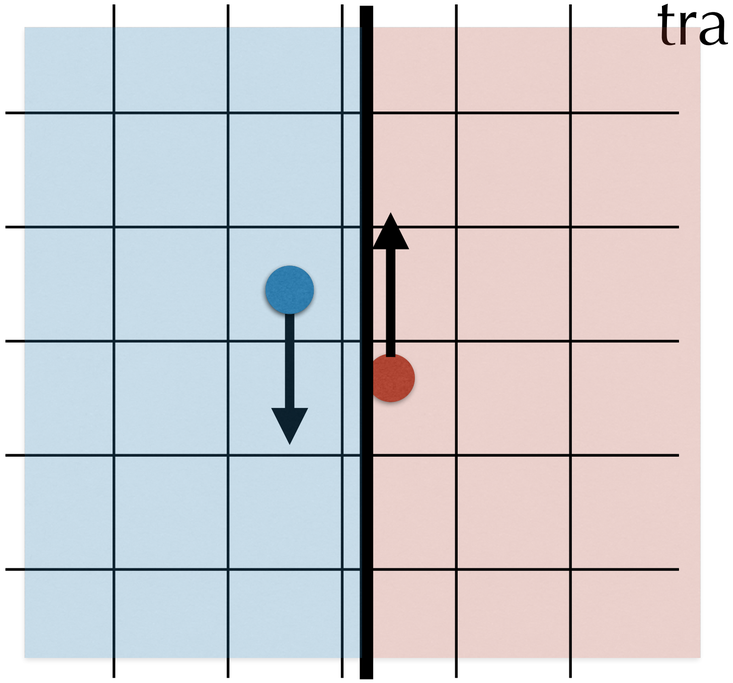
Rasterize



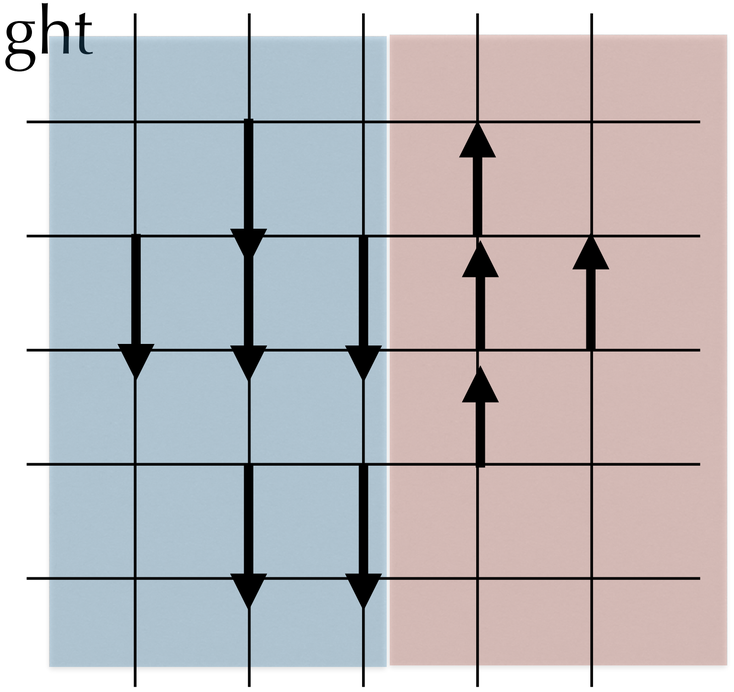
Velocity Discontinuity (Compatible Particle-in-Cell, CPIC)



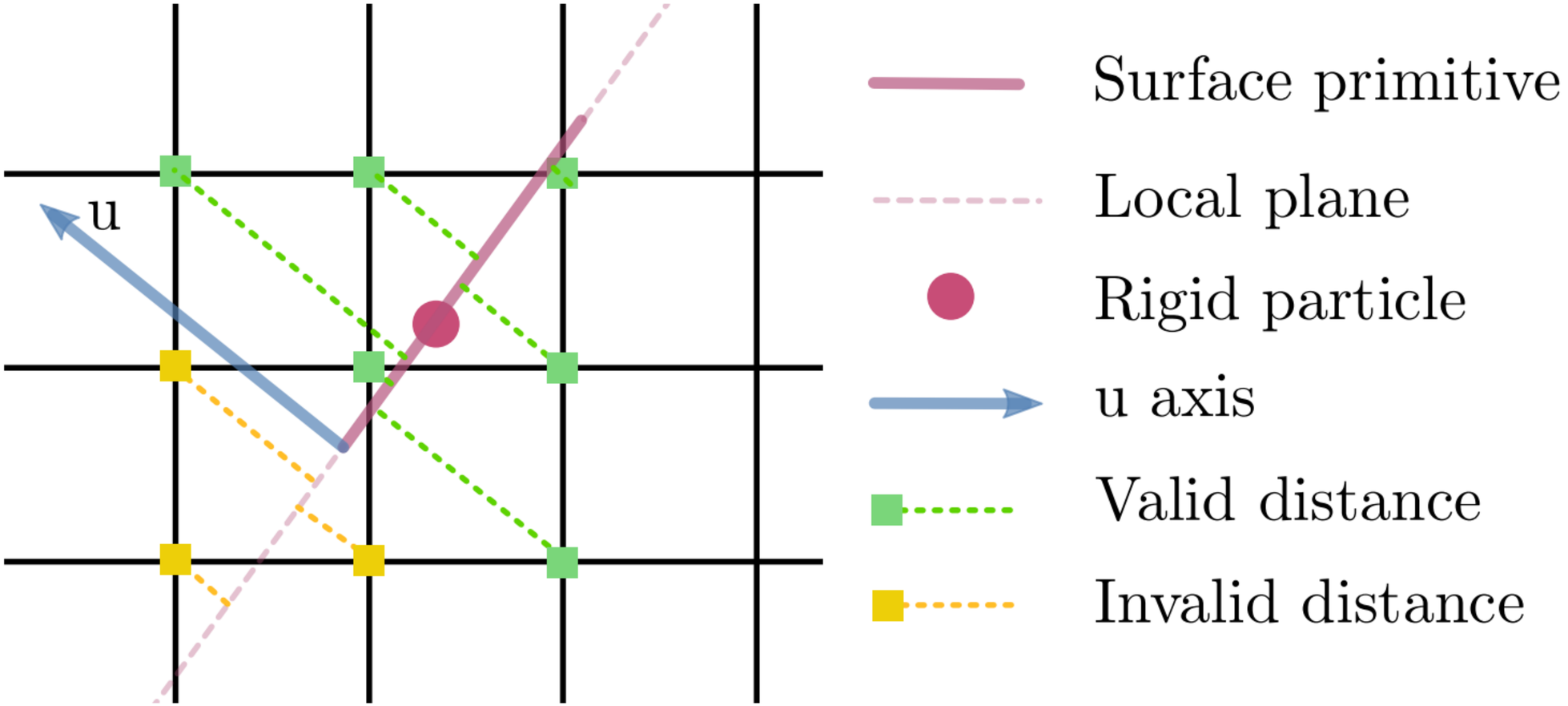
Particle-grid
transfer contribution weight



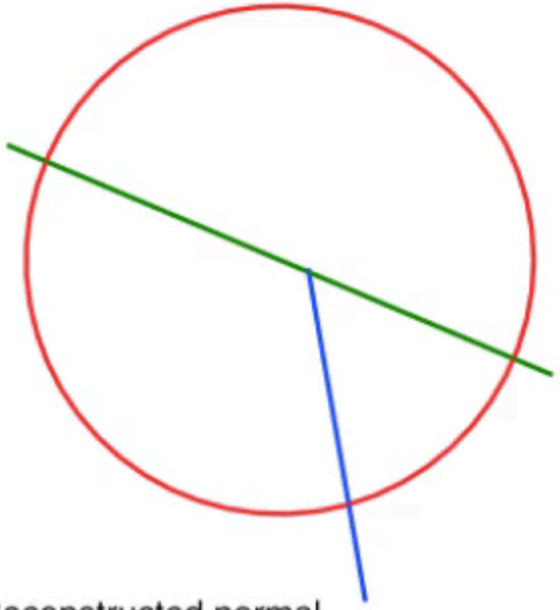
Resample



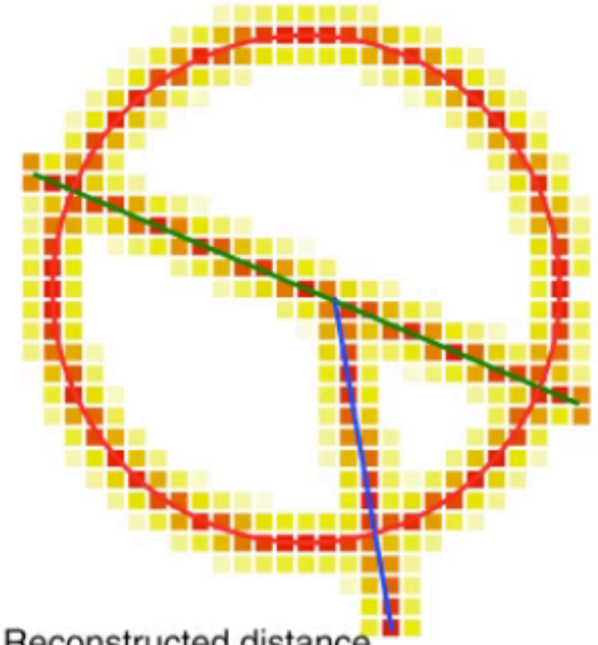
Representing boundaries



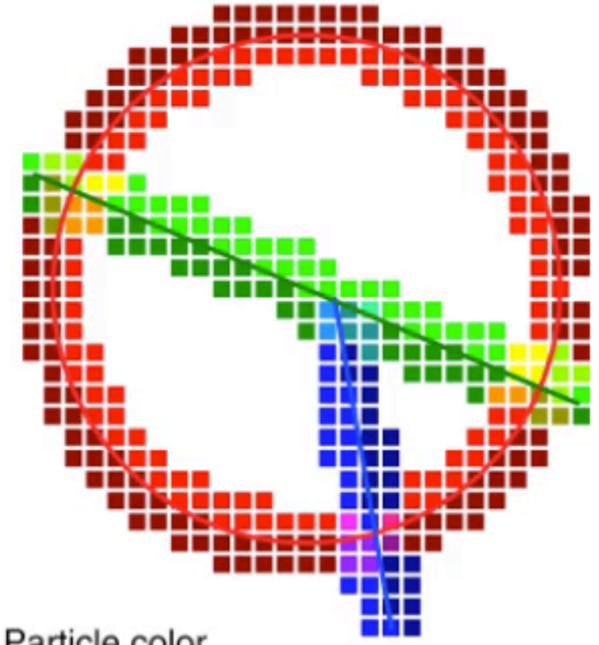
Boundary mesh



Grid distance

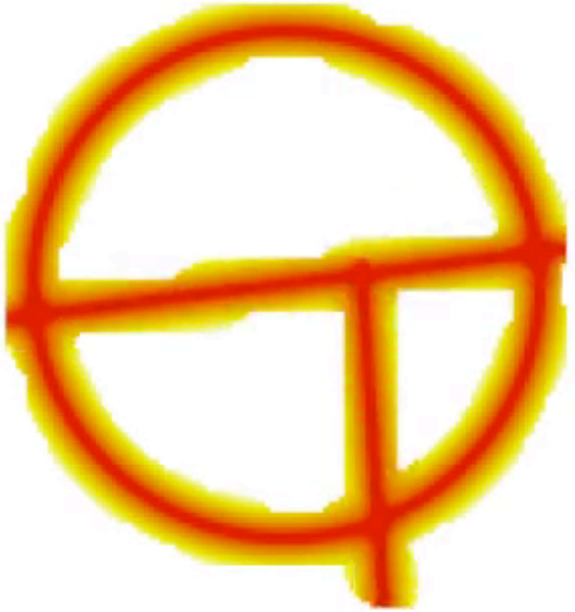


Grid color

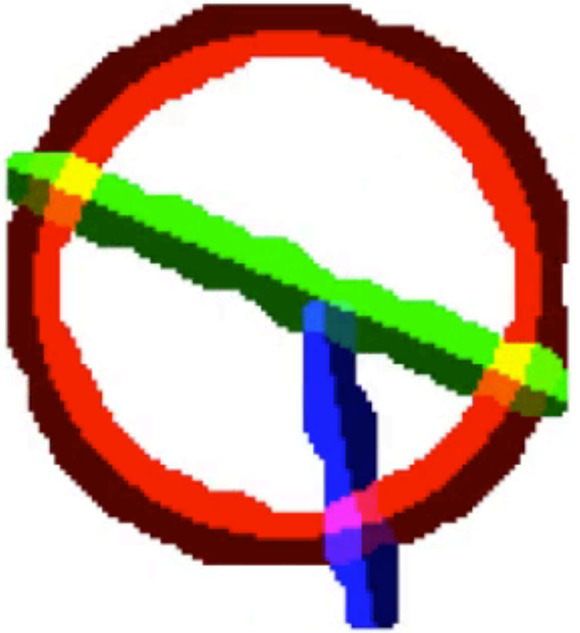


Reconstructed normal

Reconstructed distance



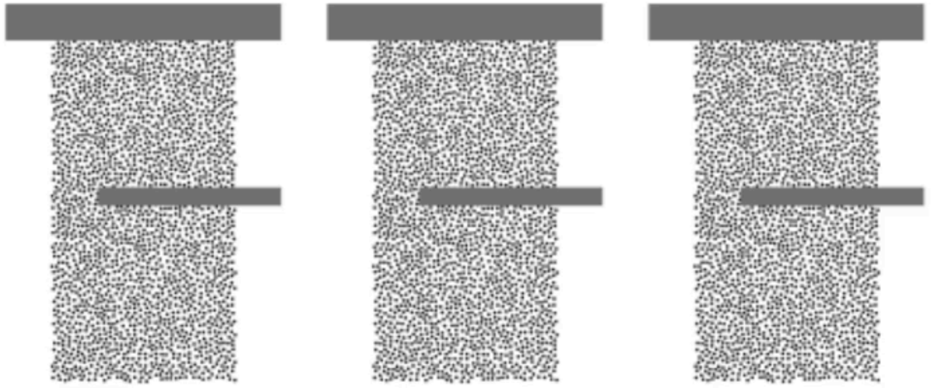
Particle color



Level Set Cut

Sticky Slip Separate

Level Set Appear

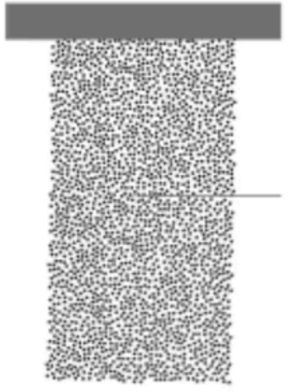


Sticky Slip Separate

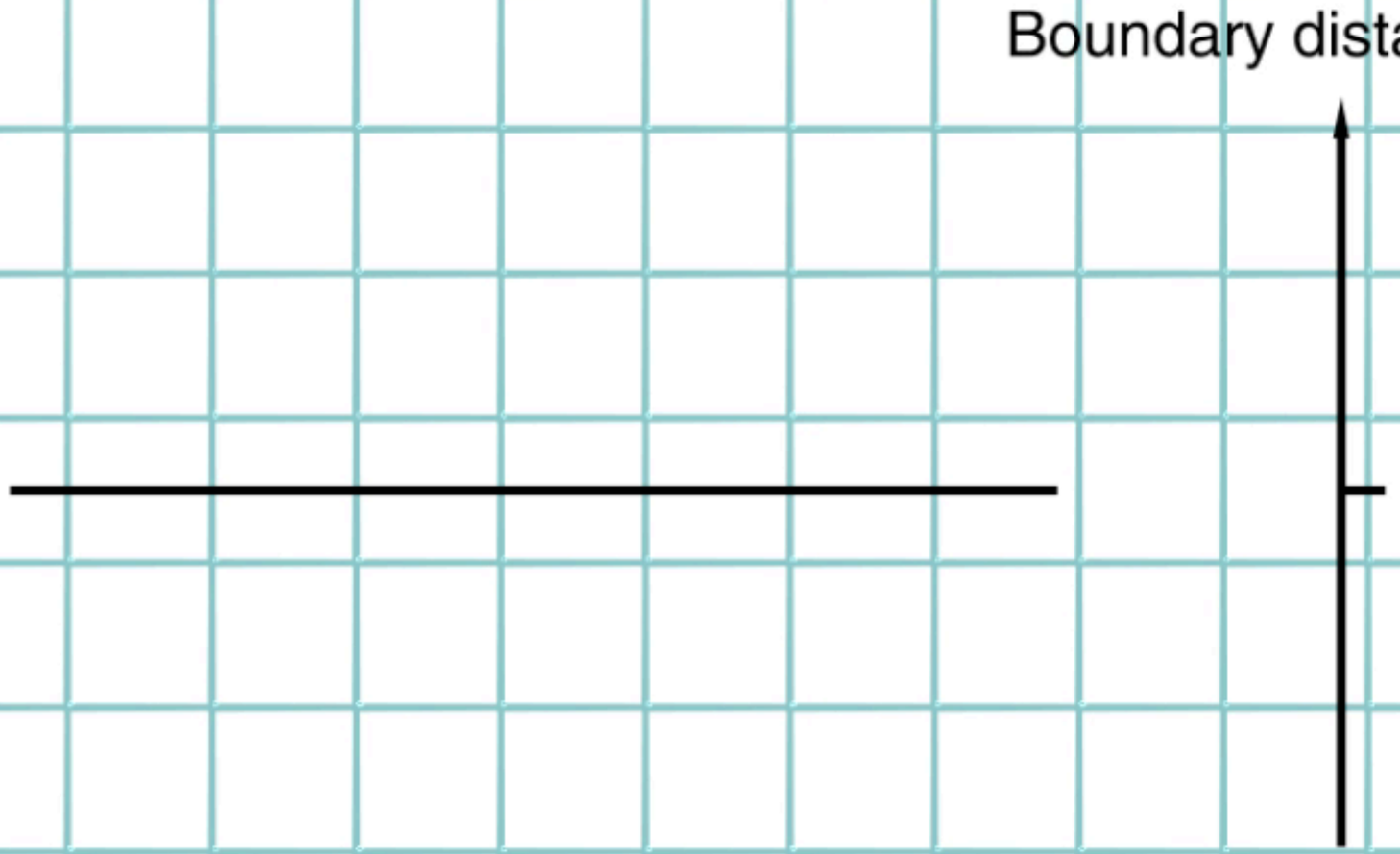
Traditional Method

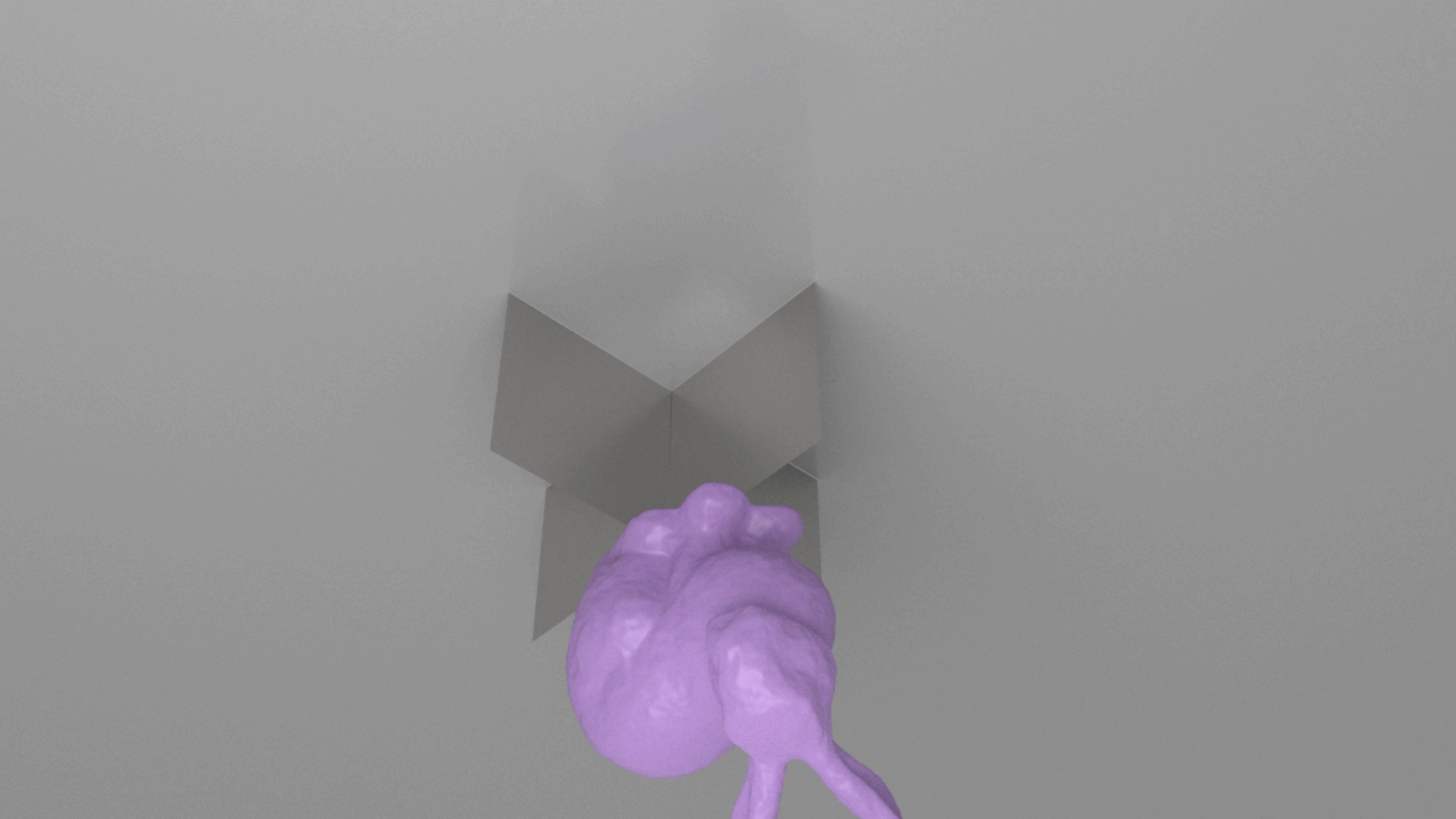
1dx Remove 1.5dx Remove 2dx Softening

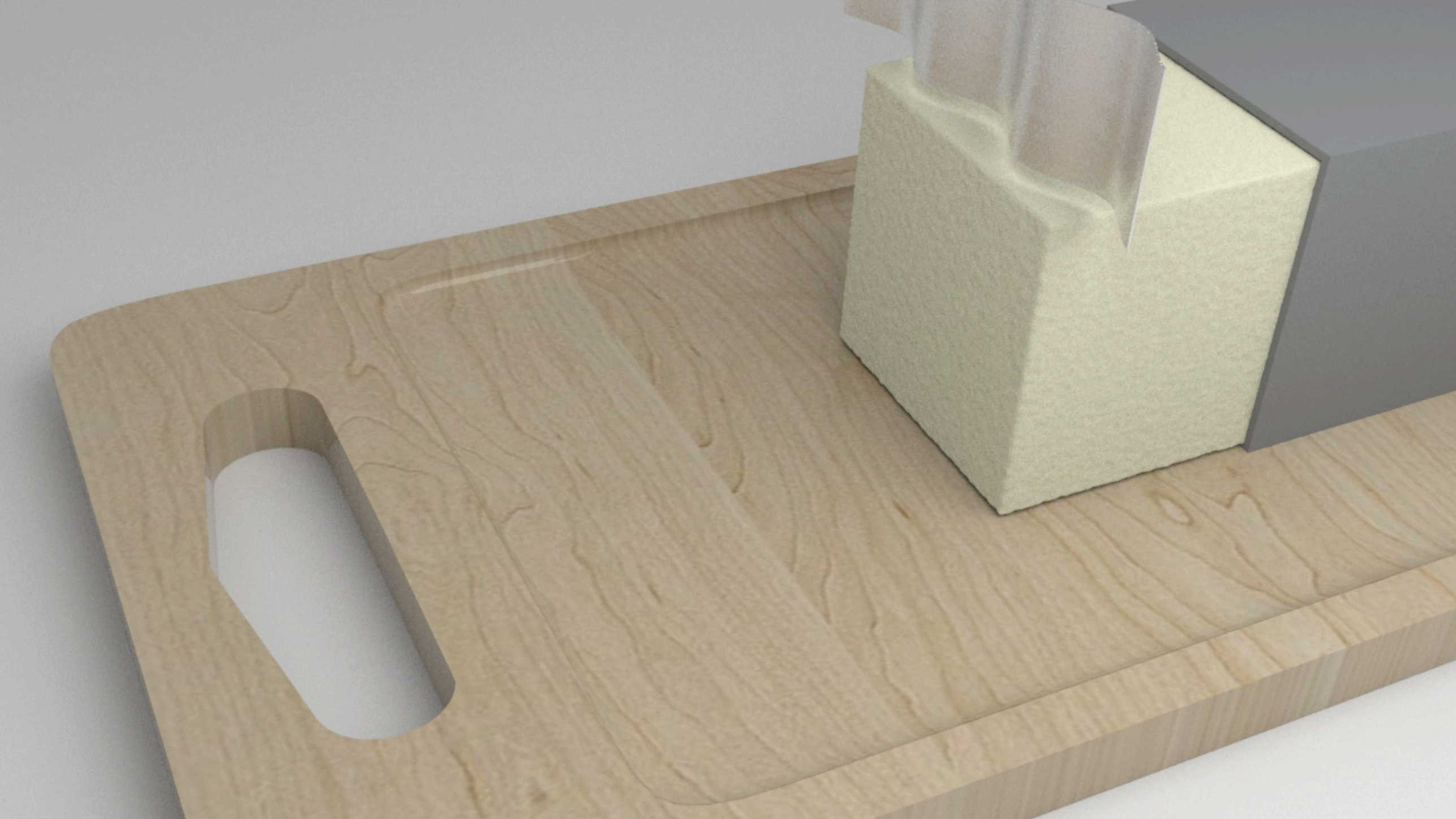
Our Method

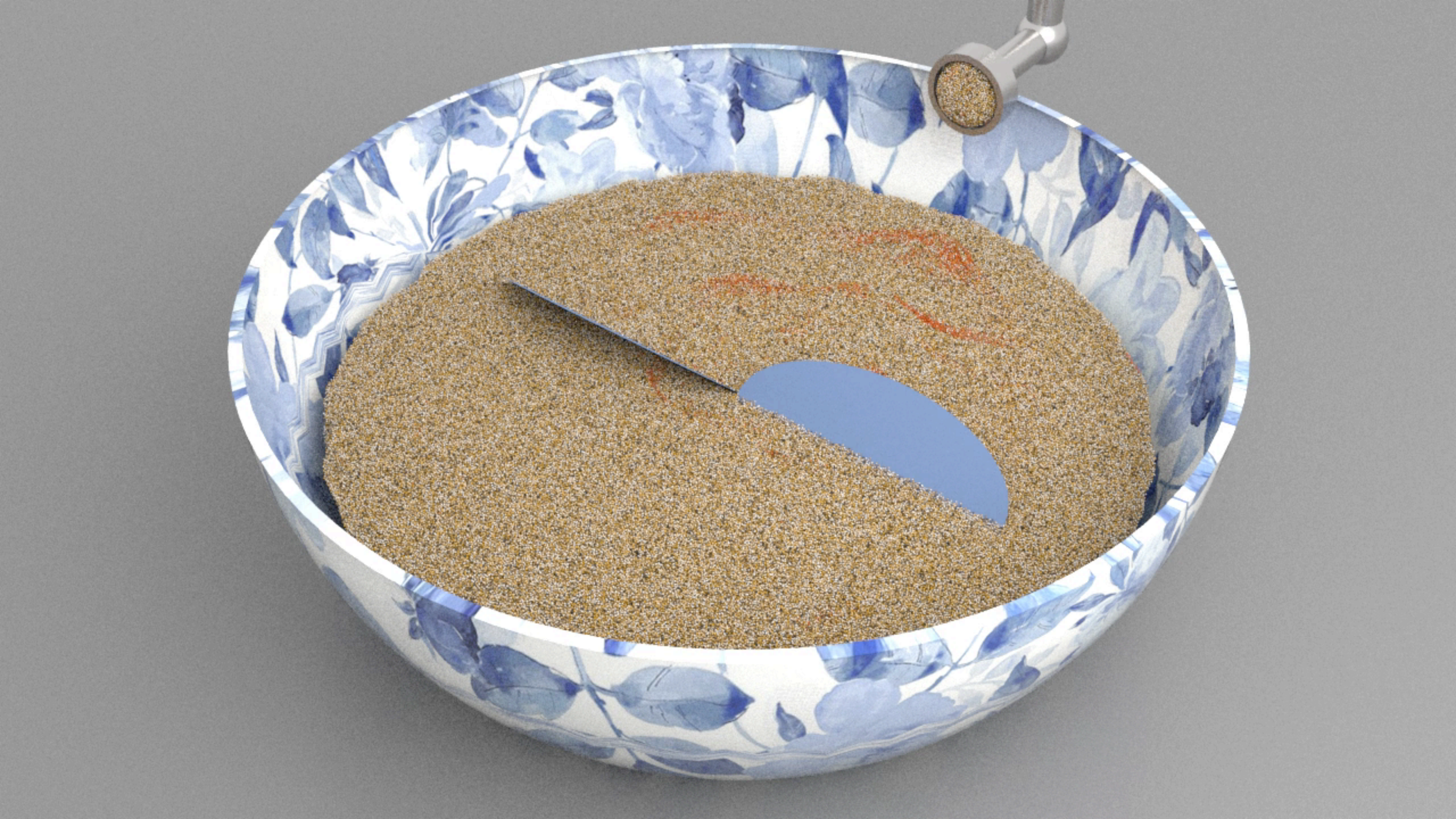


Incremental Cut


Boundary distance

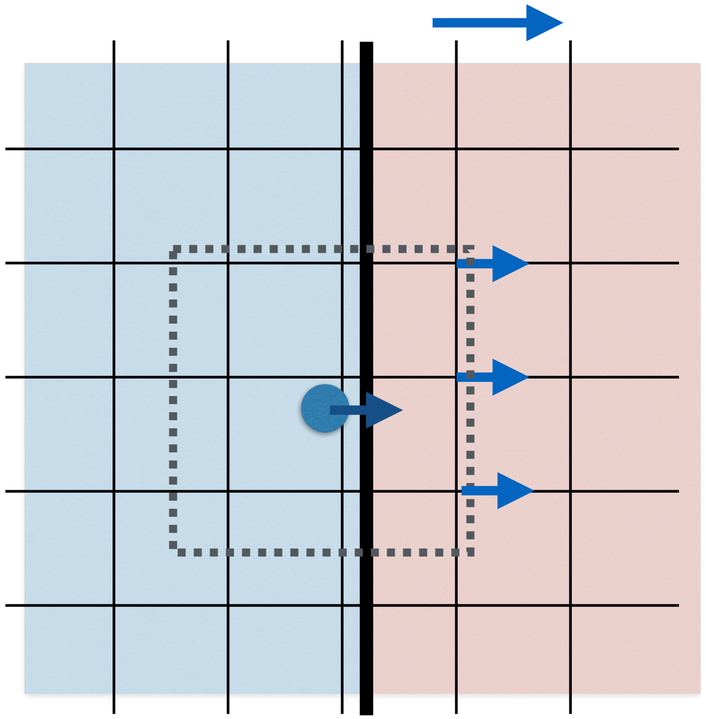




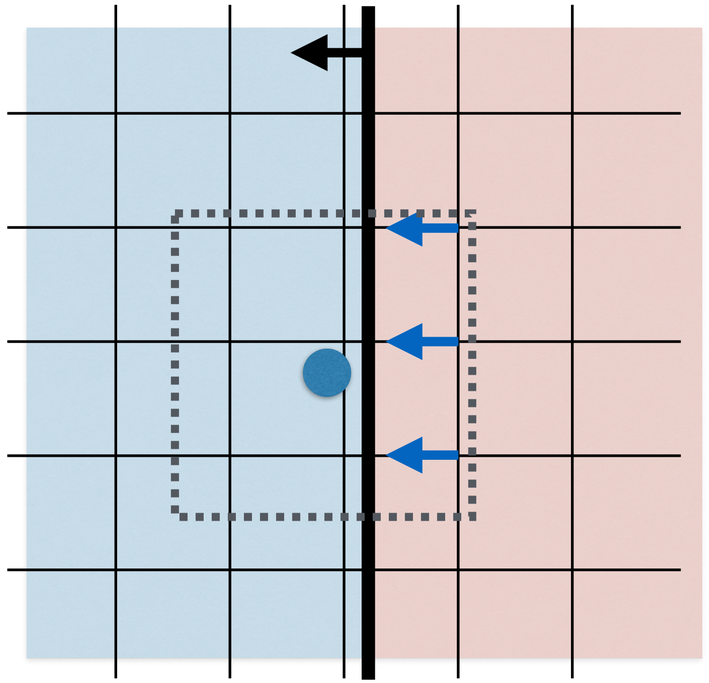


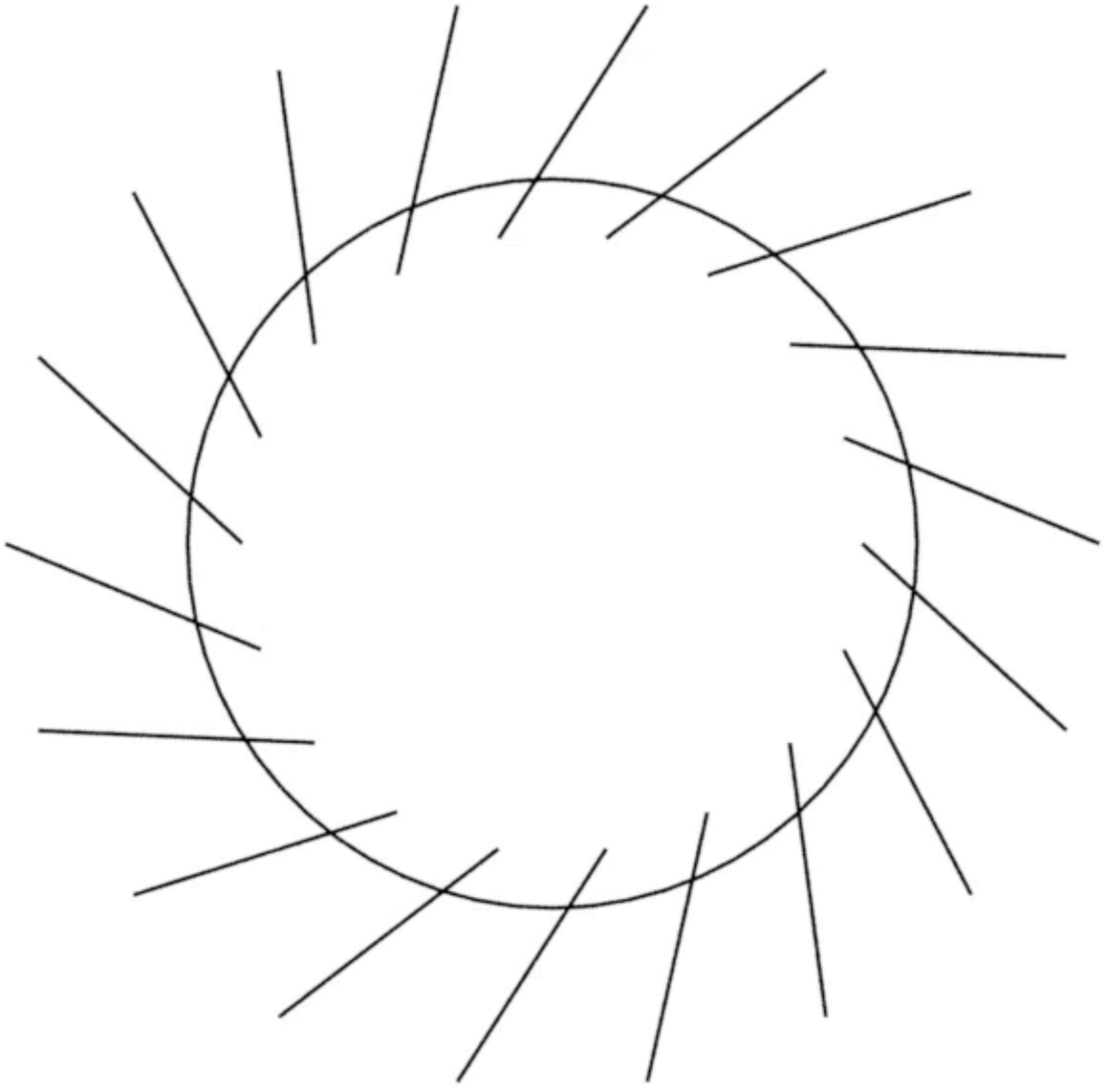
Two-way Rigid Body Coupling

Particle to rigid body
(rasterize)



Rigid body to particle
(resample)

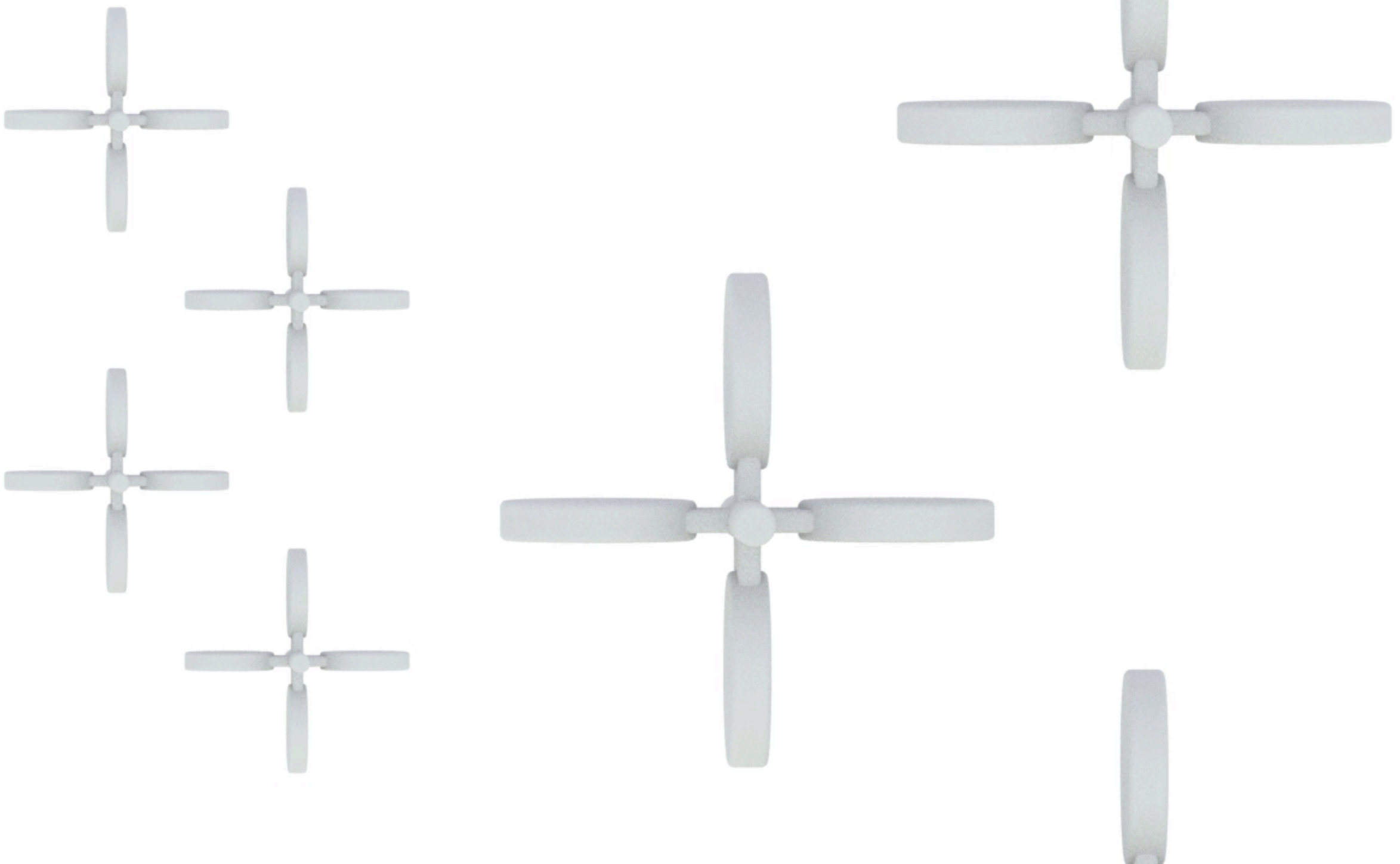




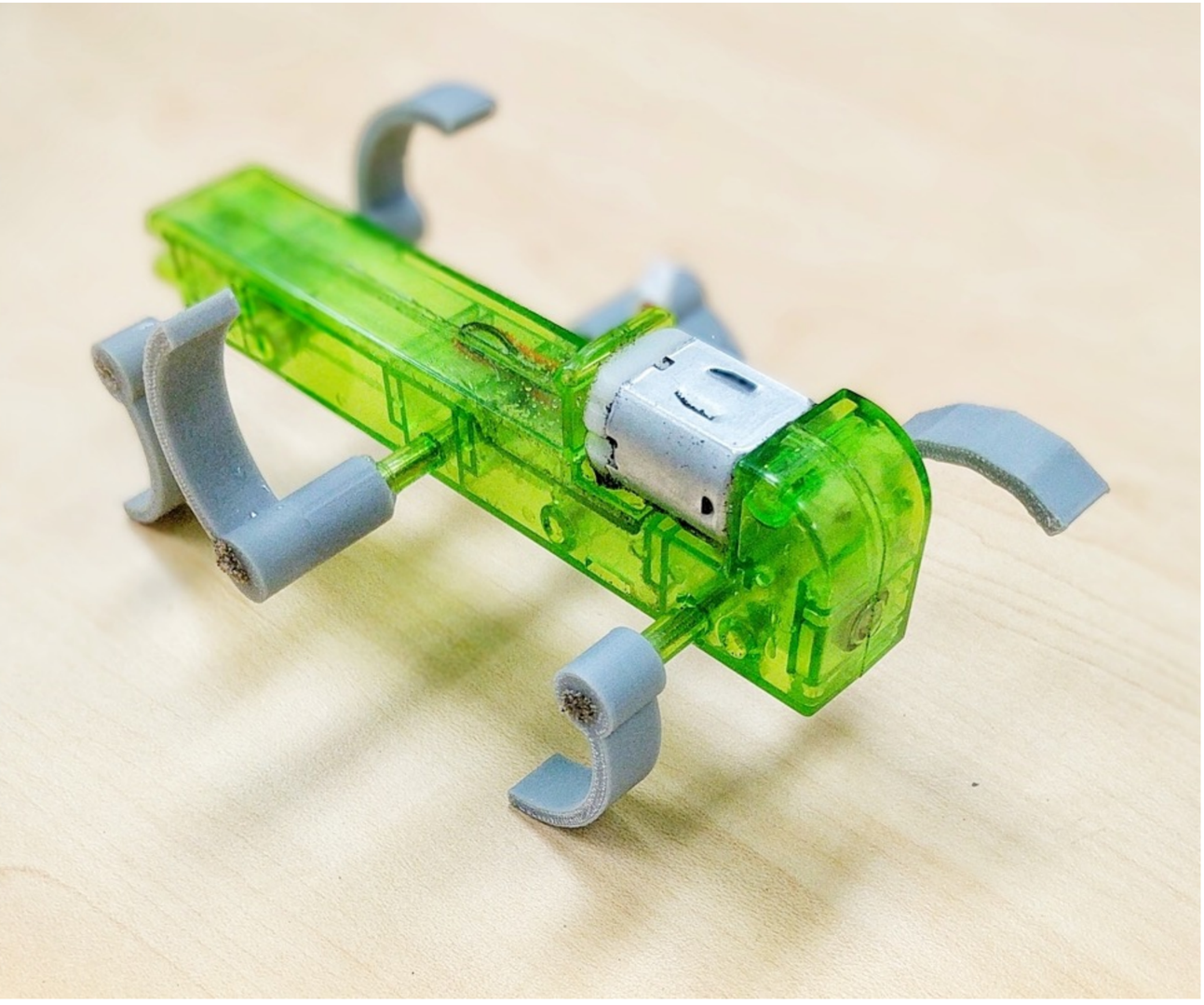
Inflow speed: 0.5
Wheel density: 1.0

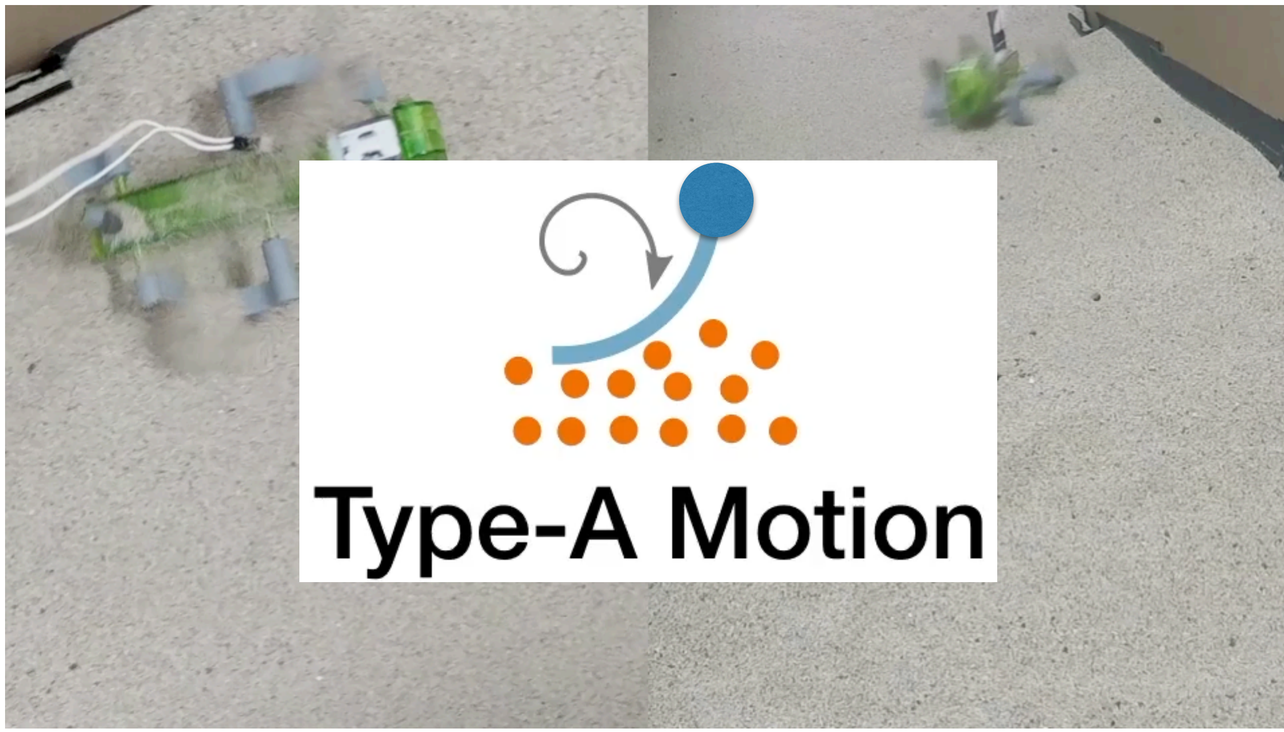
Inflow speed: 1.0
Wheel density: 4.0

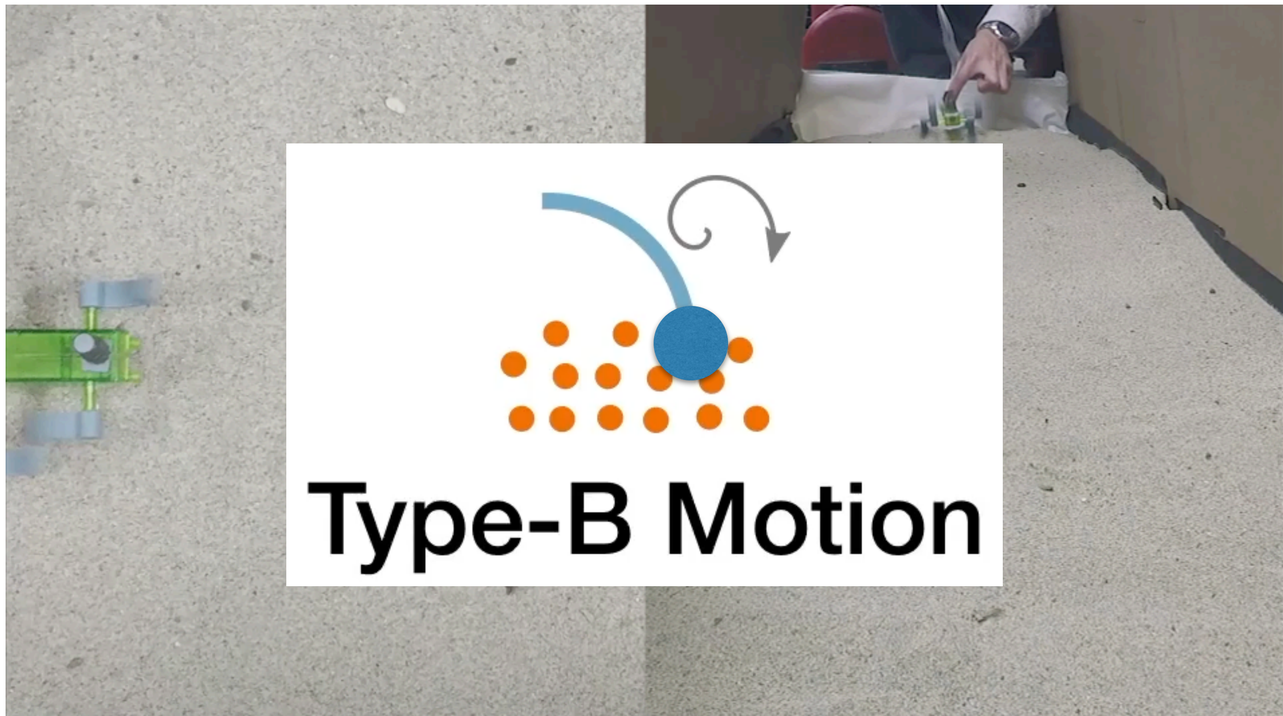
Inflow speed: 0.5
Wheel density: 4.0

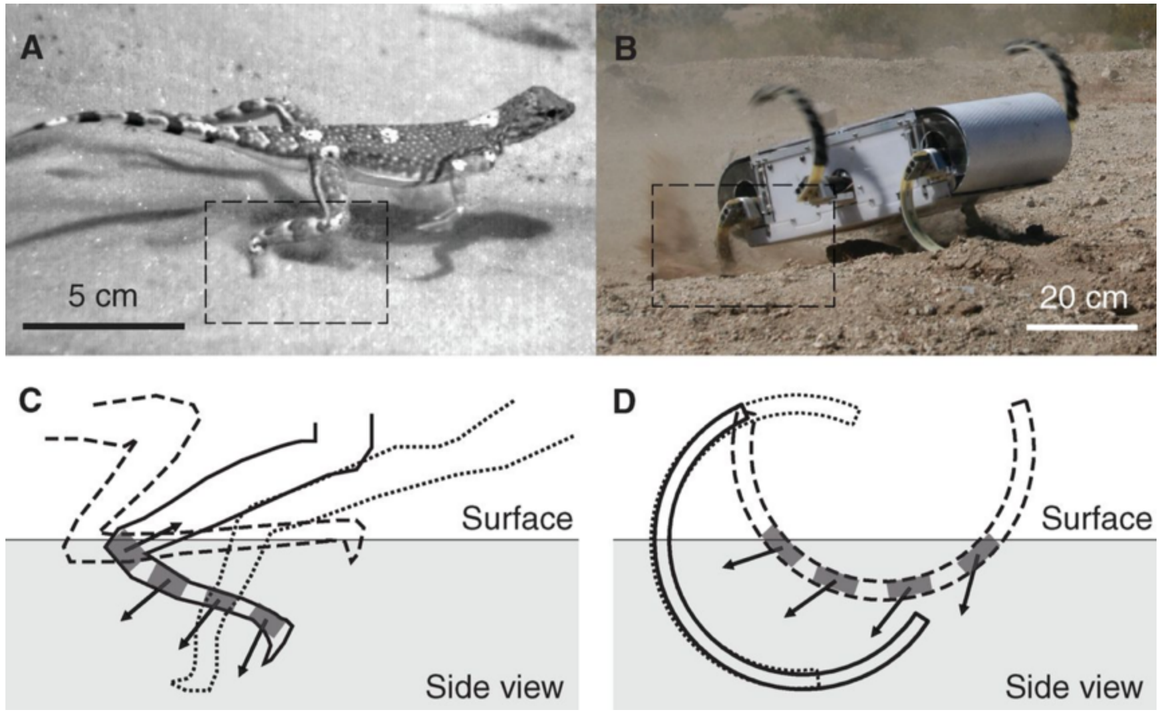


Terradynamics:
Robot and granular media



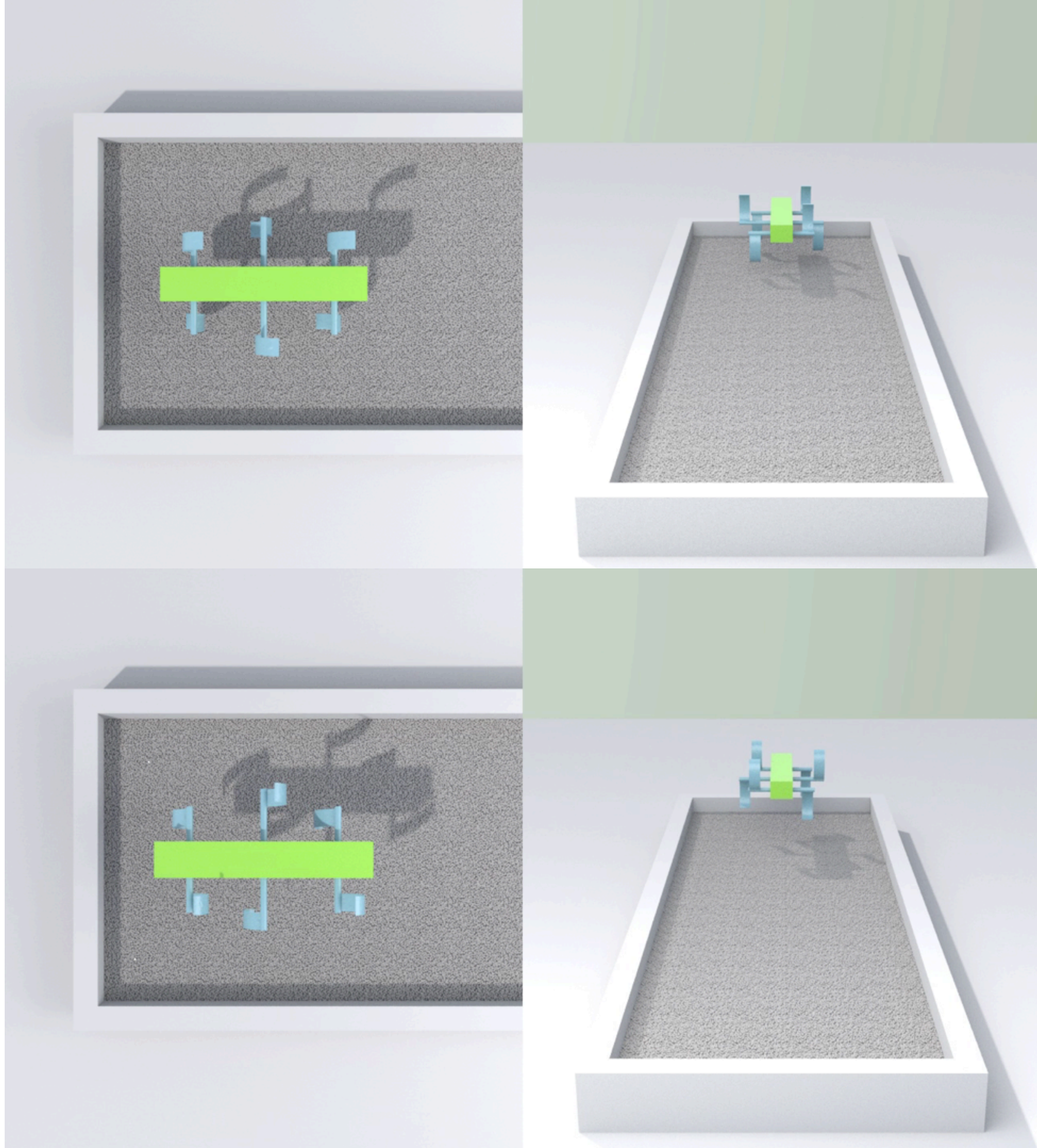


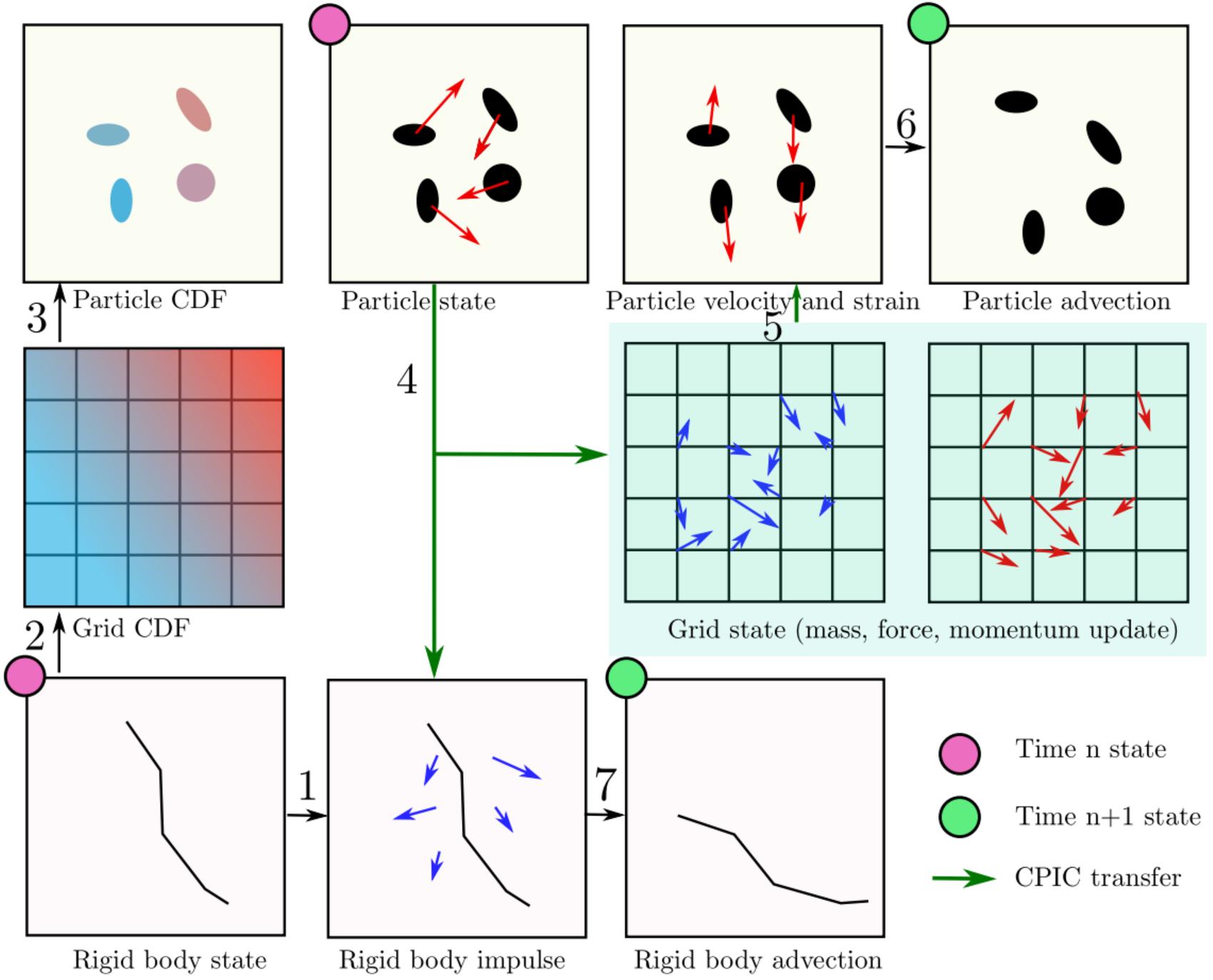




[Li et al., A terradynamics of legged locomotion on granular media. **Science 2013**]

This direction should be faster ⟶




Particle CDF
Particle state
Particle velocity and strain
Particle advection
3
4
5
6
Grid CDF
2
Grid state (mass, force, momentum update)
Rigid body state
Rigid body impulse
Rigid body advection
1
7
Time n state
Time n+1 state
CPIC transfer

Variable	Type	Meaning
u	any	any continuous function approximated with MLS
$\mathbf{x}_i$	vector	the location of sample/node i
$\mathbf{x}_p$	vector	the location of particle p
$\mathbf{z}$, $\mathbf{x}$	vector	an arbitrary continuous location
$\mathbf{P}(\mathbf{x})$	vector	the polynomial basis
$\mathbf{c}(\mathbf{x})$	vector	all basis coefficients
$\mathbf{M}(\mathbf{x})$	matrix	the moment matrix
$\mathbf{M}_p$	matrix	$\mathbf{M}(\mathbf{x}_p)$
$\xi_i(\mathbf{x})$	scalar	weighting function centered at $\mathbf{x}_i$
$\Phi_i(\mathbf{x})$	scalar	MLS shape function centered at $\mathbf{x}_i$
$N_i(\mathbf{x})$	scalar	B-spline basis function centered at $\mathbf{x}_i$
$\rho(\mathbf{x}, t)$	vector	the continuous density field
$\mathbf{v}(\mathbf{x})$	vector	the continuous velocity field
m_i	scalar	mass of node i
$\mathbf{v}_i$	vector	velocity of node i
$\mathbf{v}_i^n$	vector	velocity of node i at time n over domain Ω^{t^n}
$\hat{\mathbf{v}}_i^n$	vector	velocity of node i at time $n+1$ over domain Ω^{t^n}
m_p	scalar	mass of particle p
$\mathbf{v}_p$	vector	velocity of particle p
$\mathbf{C}_p$	matrix	affine matrix of particle p
$\mathbf{q}$	vector	test function in the weak form
$q_{\alpha,\beta}$	scalar	derivative of q_α wrt. x_β
$\boldsymbol{\sigma}$	matrix	Cauchy stress
$\mathbf{F}_p$	matrix	the deformation gradient on particle p
$\mathbf{f}_i$	vector	the force on grid node i

$$\frac{1}{\Delta t}\int_{\Omega^{t^n}} \rho(\mathbf{x},t^n)\left(\hat{v}_\alpha^{n+1}(\mathbf{x}) - v_\alpha^n(\mathbf{x})\right) q_\alpha(\mathbf{x},t^n)d\mathbf{x}$$
$$= \int_{\partial\Omega^{t^n}} q_\alpha(\mathbf{x},t^n)\mathcal{T}_\alpha(\mathbf{x},t^n)ds - \int_{\Omega^{t^n}} q_{\alpha,\beta}(\mathbf{x},t^n)\sigma_{\alpha\beta}(\mathbf{x},t^n)d\mathbf{x}, \tag{7}$$

$$\mathbf{f}_i = -\frac{\partial E}{\partial \mathbf{x}_i} = -\sum_p N_i(\mathbf{x}_p^n)V_p^0 M_p^{-1}\frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}_p^n)\mathbf{F}_p^{nT}(\mathbf{x}_i^n - \mathbf{x}_p^n), \tag{18}$$

$$m_i^n = \sum_{q\in\{p^{i+}\}} N_i(\mathbf{x}_q^n)m_q, \tag{23}$$

$$(m\mathbf{v})_i^n = \sum_{q\in\{p^{i+}\}} N_i(\mathbf{x}_q^n)m_q\left(\mathbf{v}_q^n + \mathbf{C}_q^n(\mathbf{x}_i - \mathbf{x}_q^n)\right). \tag{24}$$

$$\mathbf{Proj}(\mathbf{v},\mathbf{n},B,\mu) = \begin{cases} \vec{0}, & B \text{ is sticky,} \\ \mathbf{v}_t, & B \text{ is slip,} \\ \zeta\hat{\mathbf{v}}_t, & B \text{ is separate and } \mathbf{v}\cdot\mathbf{n} \le 0, \\ \mathbf{v}, & B \text{ is separate and } \mathbf{v}\cdot\mathbf{n} > 0, \end{cases} \tag{25}$$

$$\mathbf{v}_p^{n+1} = \sum_{j\in i^{p-}} N_j(\mathbf{x}_p^n)\tilde{\mathbf{v}}_p + \sum_{j\in i^{p+}} N_j(\mathbf{x}_p^n)\hat{\mathbf{v}}_j^{n+1}, \tag{27}$$

$$\mathbf{C}_p^{n+1} = D_p^{-1}\left(\sum_{j\in i^{p-}} N_j(\mathbf{x}_p^n)\tilde{\mathbf{v}}_p\mathbf{z}_{jp}^{n\,T} + \sum_{j\in i^{p+}} N_j(\mathbf{x}_p^n)\hat{\mathbf{v}}_j^{n+1}\mathbf{z}_{jp}^{n\,T}\right), \tag{28}$$

Summary (Part II, III)

- New discretization scheme that is faster and easier to implement and optimize [14x faster!]
- Displacement discontinuity + An easy way to achieve two-way coupling using CDF and CPIC.
- Developed based on Taichi. We will open-source the code.

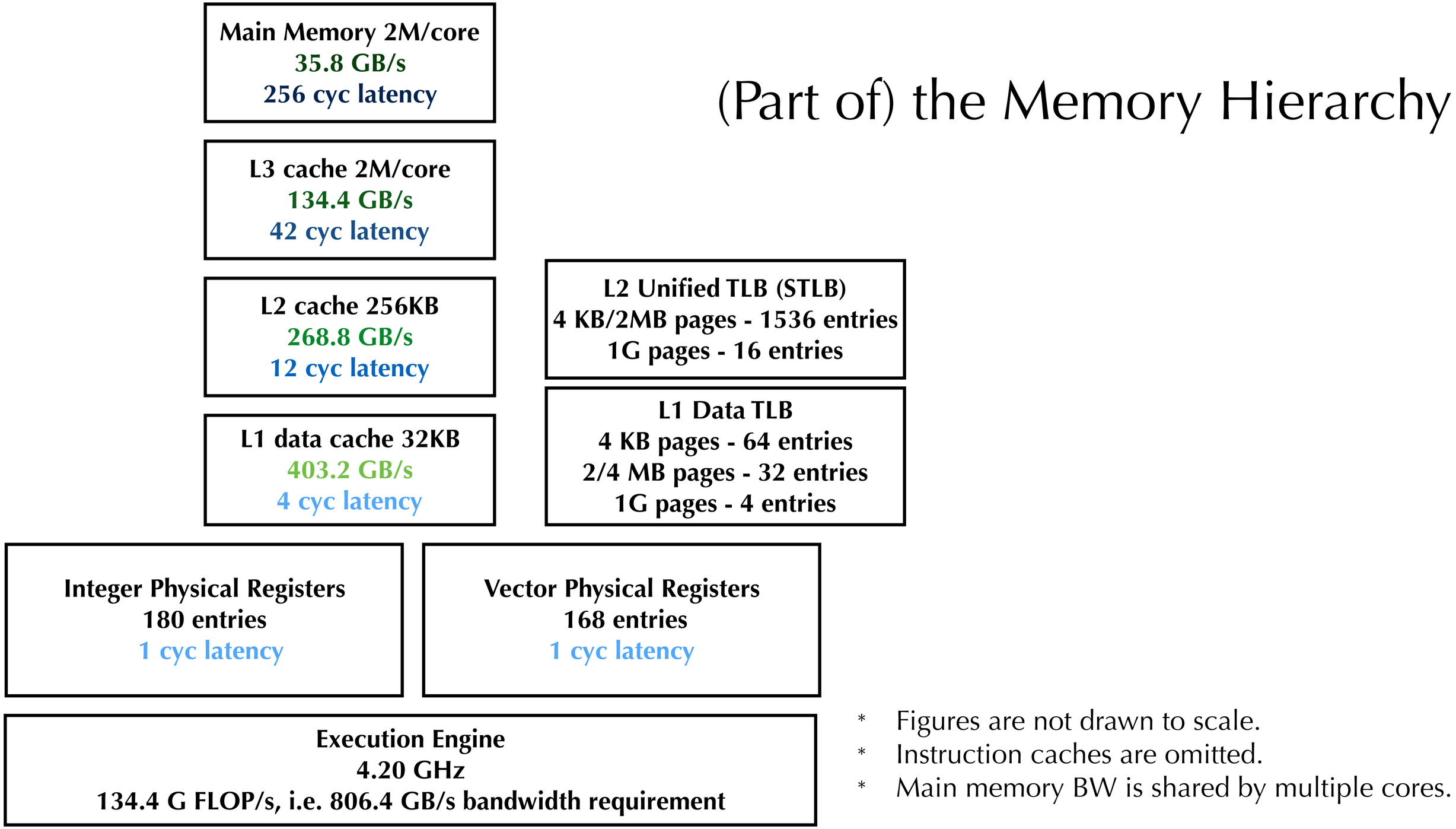
Part IV

High-performance Implementation of MPM

Table 2. Benchmarks of MPM transfer operations. Reliable reference implementation is from [Tampubolon et al. 2017]. Superscript * is with our performance optimization. All performance data are collected on an PC with an Intel Core i7-7700K CPU with four cores at 4.2GHz, and 4 × 8 GB DDR4 memory at 2400 MHz. Intel Turbo Boost is disabled for stable CPU frequency.

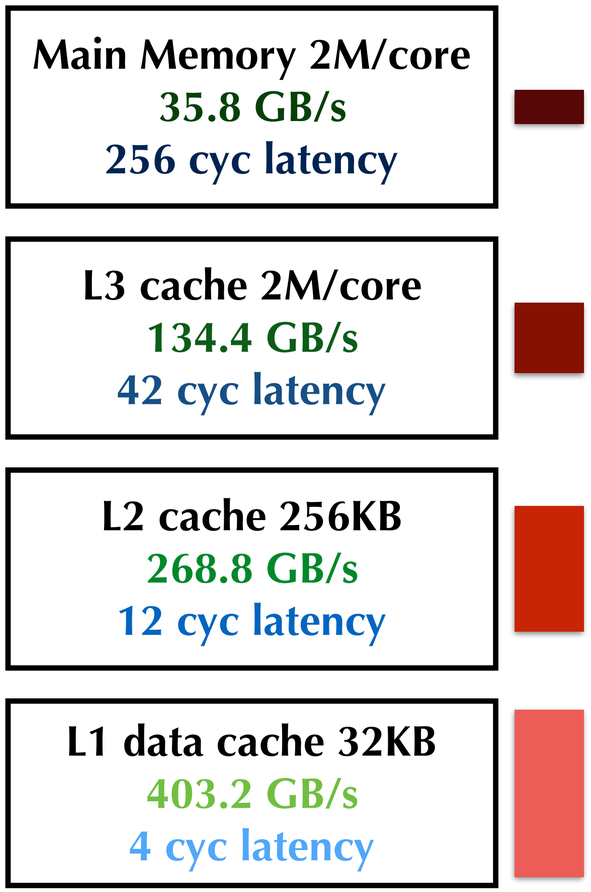
Timing (ms)	Reference	Ours (MPM)	Ours* (MPM)	Ours* (MLS-MPM)
P2G (1 thread)	4760 (1×)	5744 (0.83×)	2685 (1.77×)	1283 (3.71×)
P2G (4 threads)	1220 (1×)	1525 (0.80×)	688 (1.77×)	328 (3.72×)
G2P (1 thread)	8255 (1×)	7476 (1.10×)	1144 (7.21×)	589 (14.01×)
G2P (4 threads)	2070 (1×)	2011 (1.03×)	313 (6.61×)	163 (12.70×)

(Part of) the Memory Hierarchy

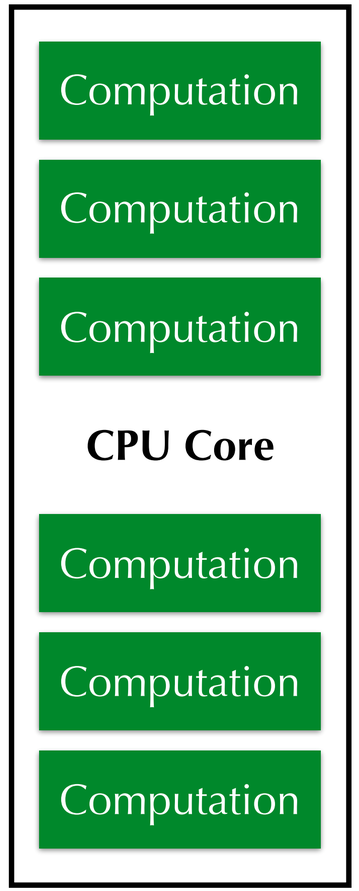


* Figures are not drawn to scale.
* Instruction caches are omitted.
* Main memory BW is shared by multiple cores.

Memory

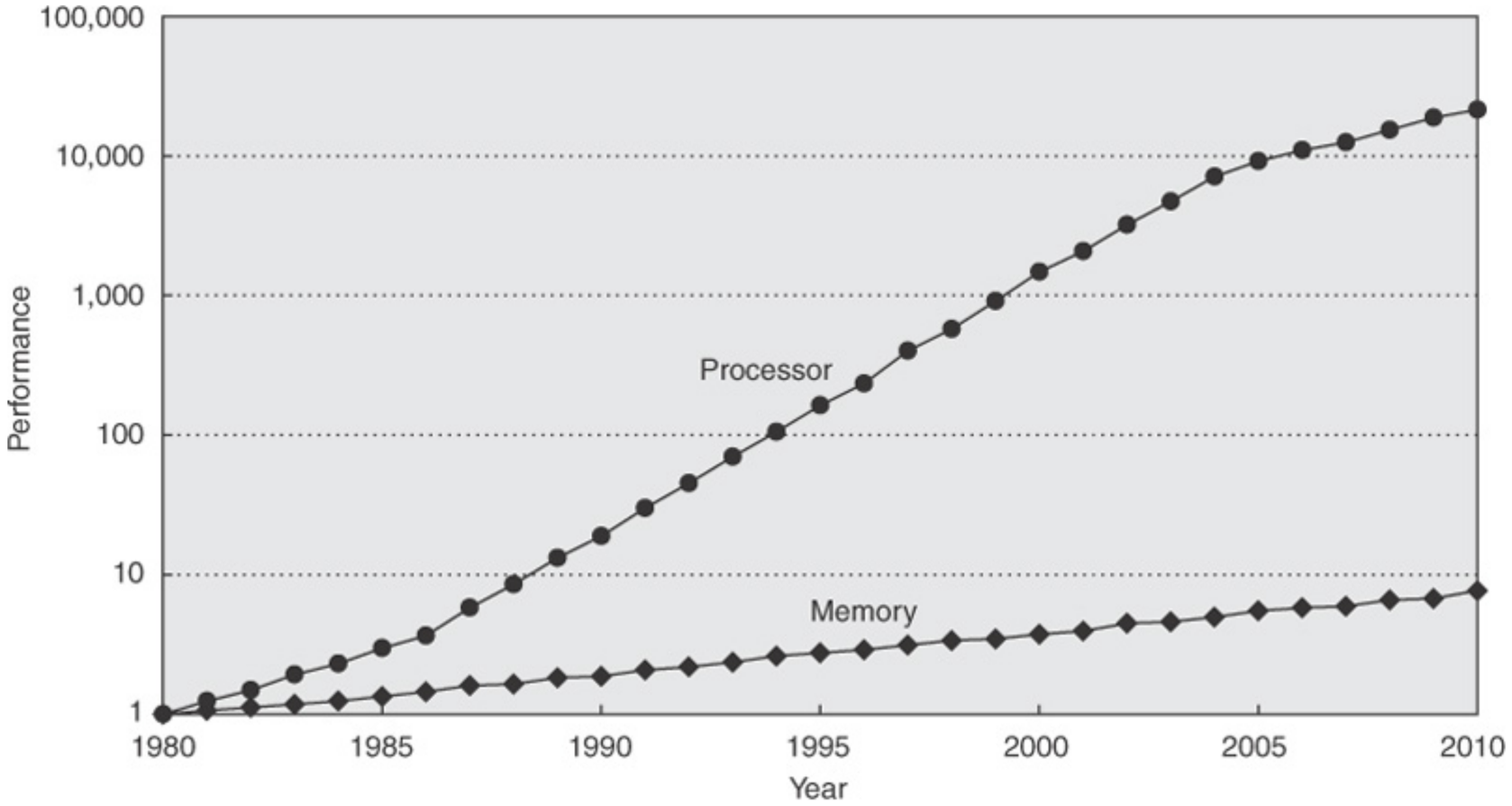


Computation

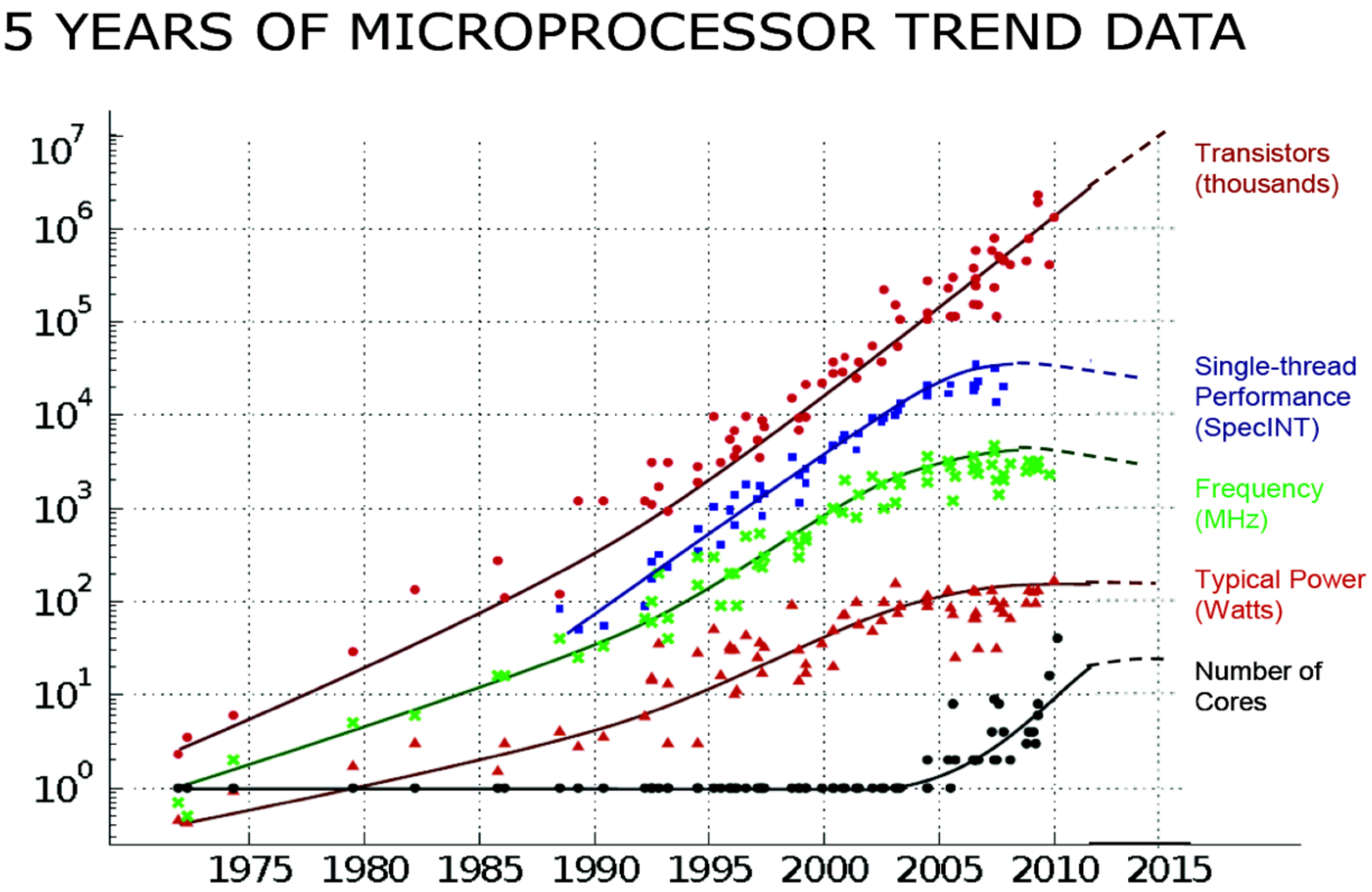


* Caches are not drawn to scale.
* Data collected from the Intel Skylake architecture.
* There can be multiple data transfers happening simultaneously.
* Access to slower memory is indirectly invoked by faster memory cache miss.

The era of slow memory…



…and parallelism.



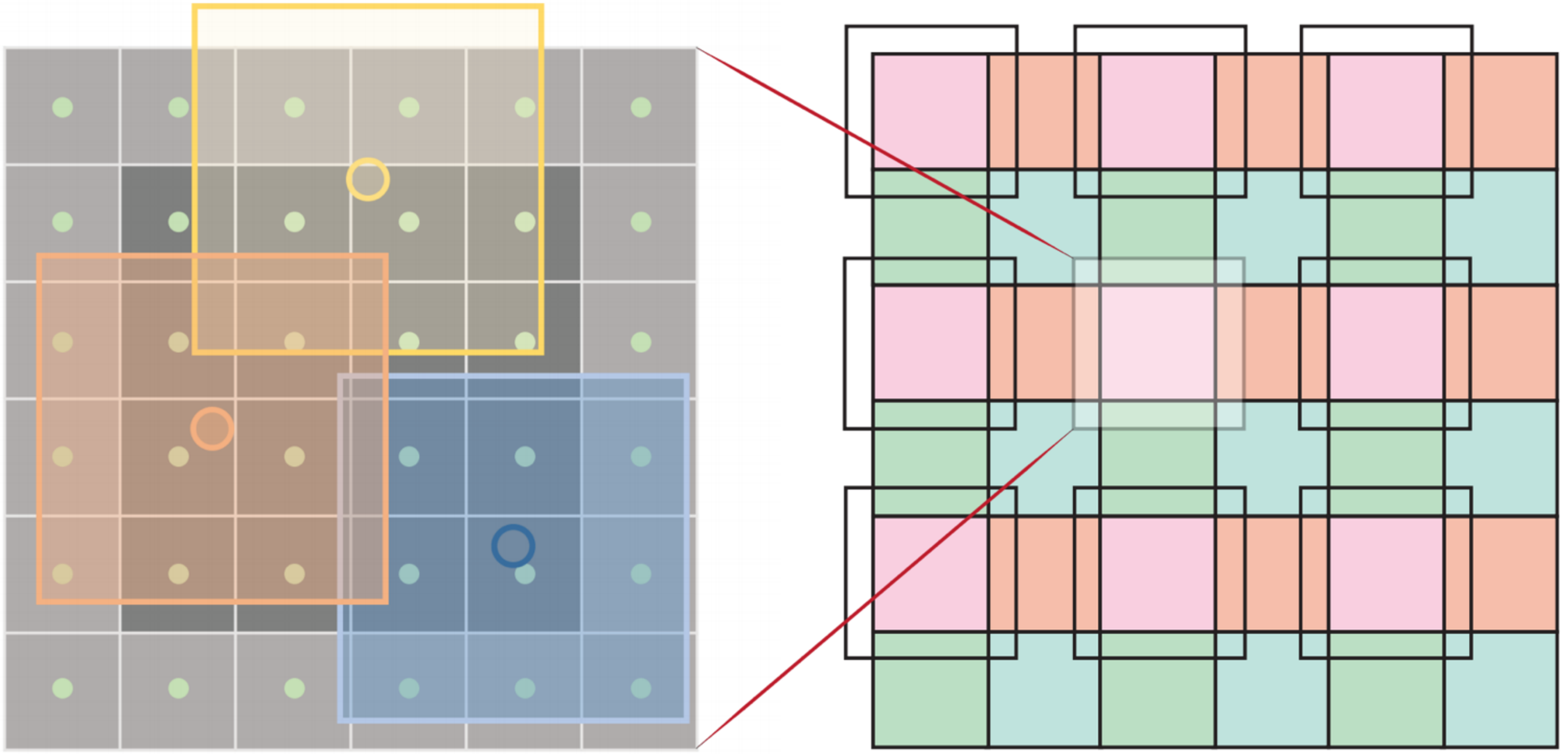
https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/

Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

Locality

- Spatial locality: access nearby data is cheaper than accessing far away data
  - Higher utility of cache lines
  - Fewer TLB misses
  - Better hardware prefetching
- Temporal locality: reuse the data as much as you can
  - Higher cache-hit rates
- Shrink the working set
  - so that data resides in lower-level (higher throughput, lower latency) memory
  - **> Blocked MPM transfer**

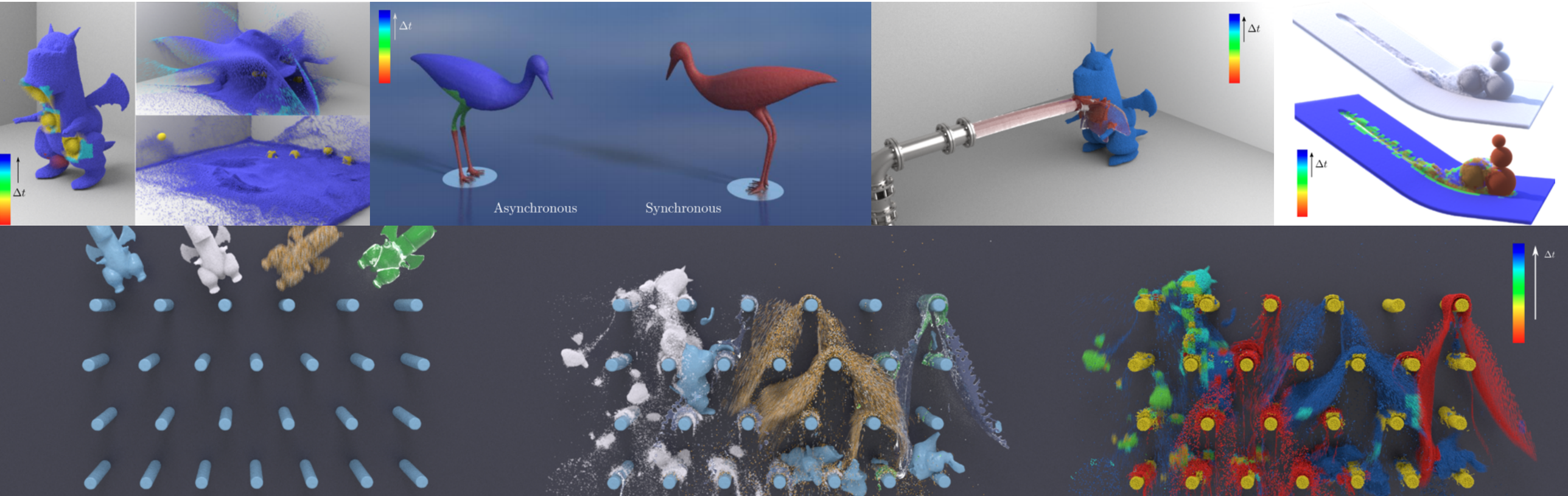
Blocked MPM Transfer & Colored Parallelisation



Part V
A Temporally Adaptive Material Point Method with Regional Time Stepping SCA 2018

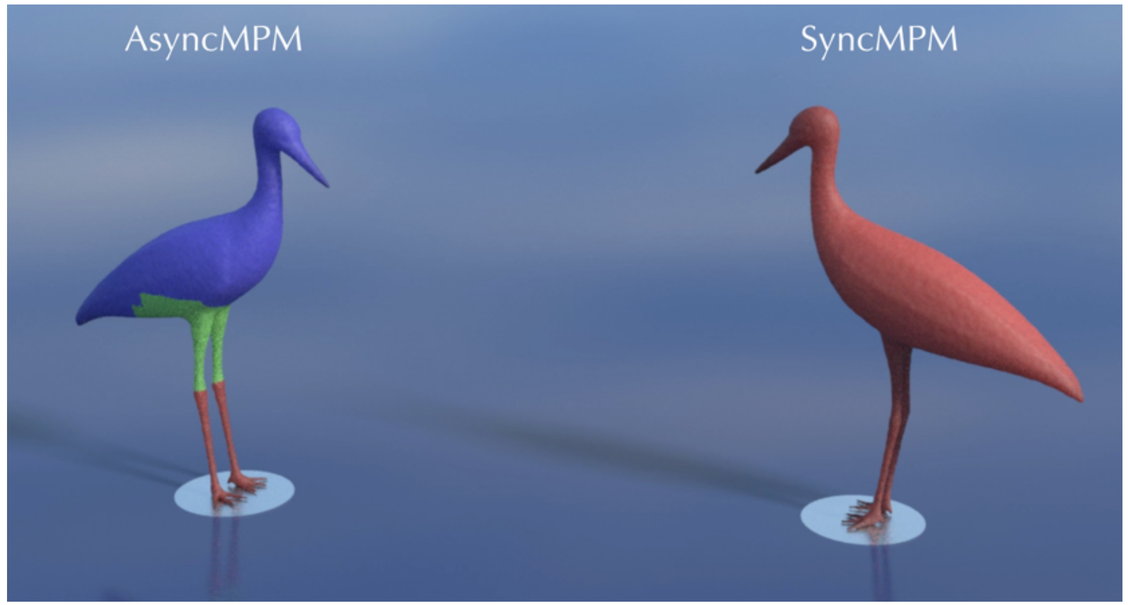
Yu Fang[1] Yuanming Hu[2] Shi-Min Hu[1] Chenfanfu Jiang[3]

[1]Tsinghua University [2]MIT CSAIL [3]University of Pennsylvania

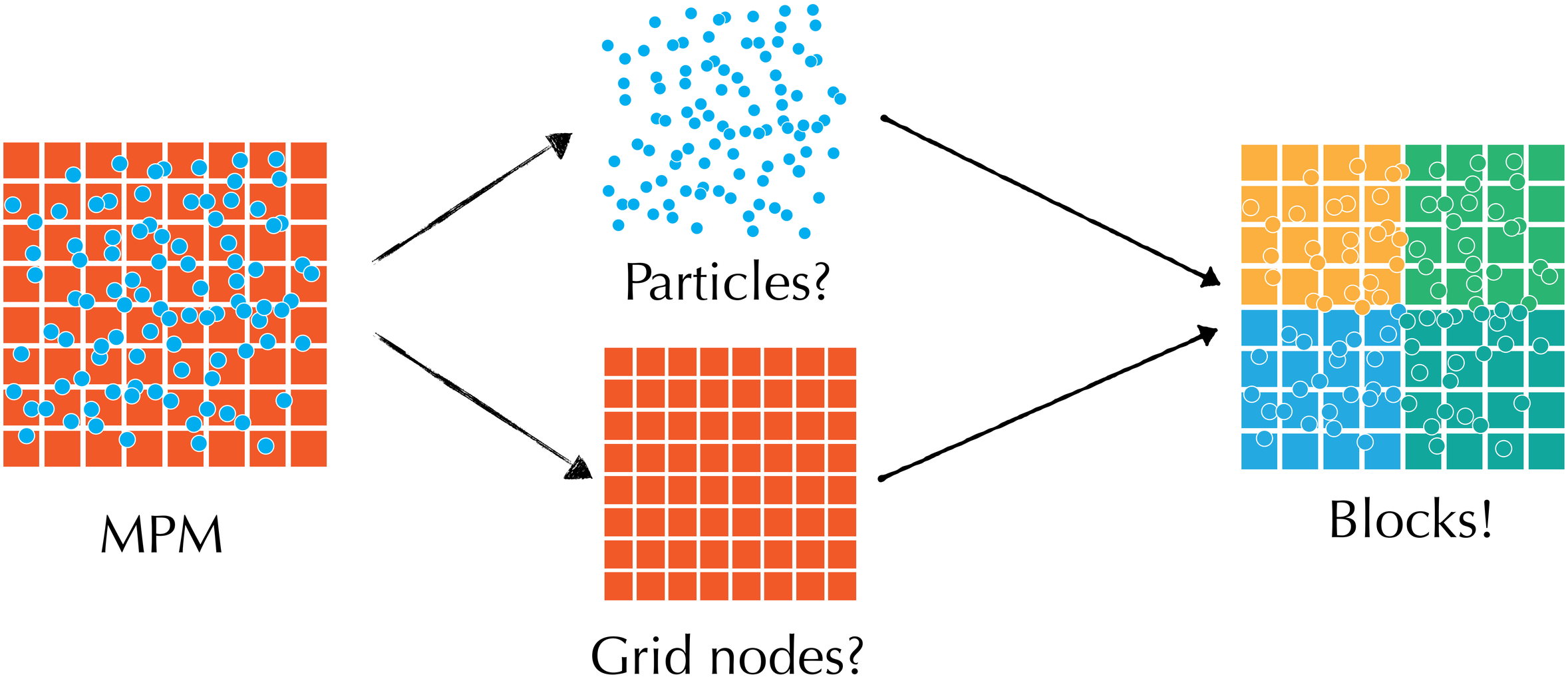


Motivation

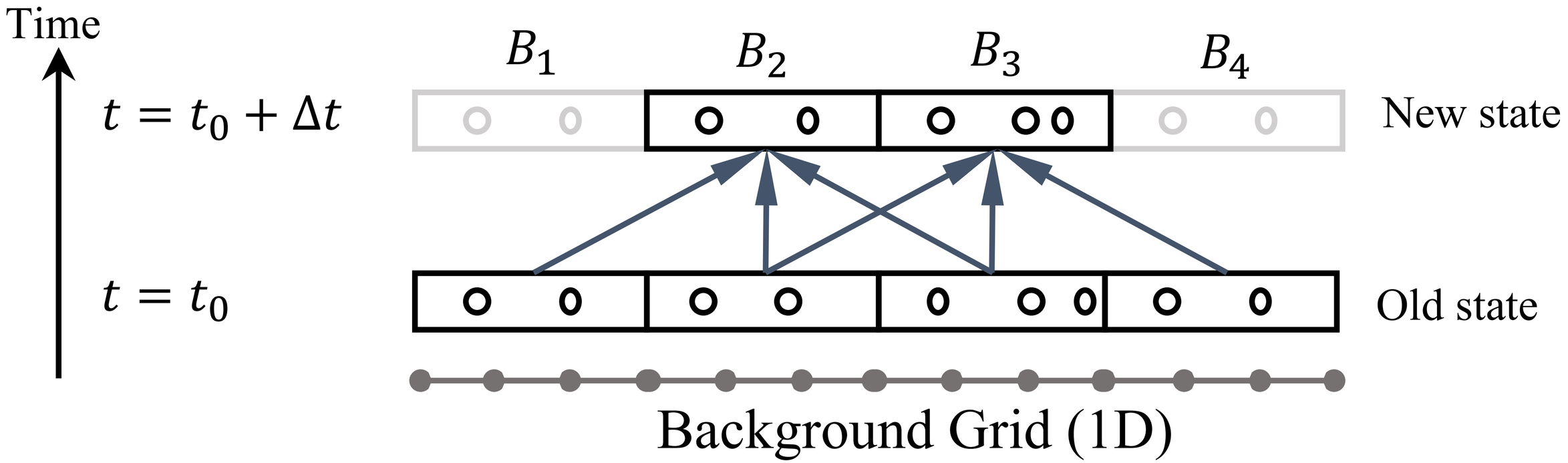
- **Global Time step is limited by region with most restrictive conditions**
  - E.g. stiffest or fastest moving regions (CFL conditions)
- **Can we use different time steps in different regions?**
  - If so, when most region needs only a large time step, we get higher efficiency



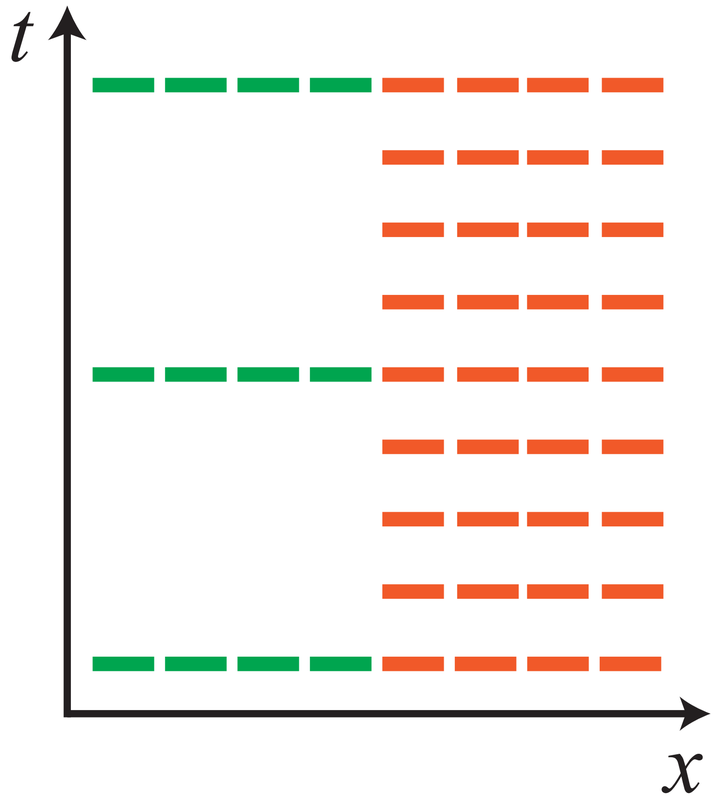
Challenge: Scheduling Uint

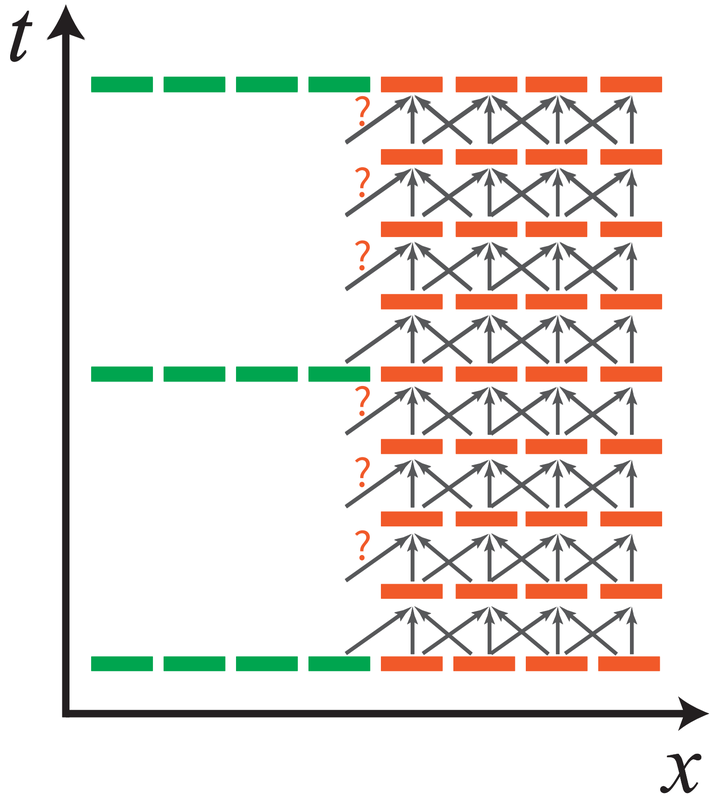


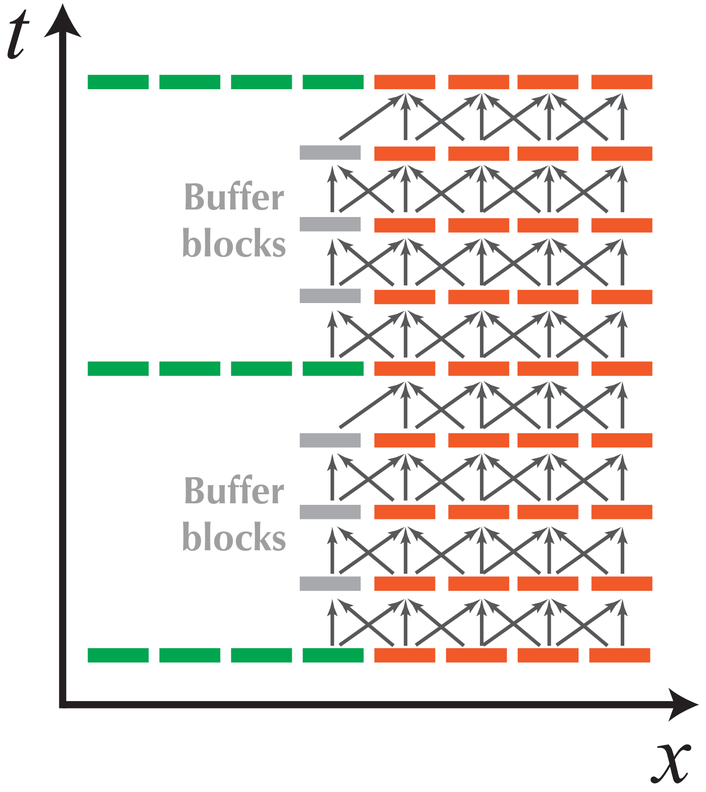
Challenge: Block Decency



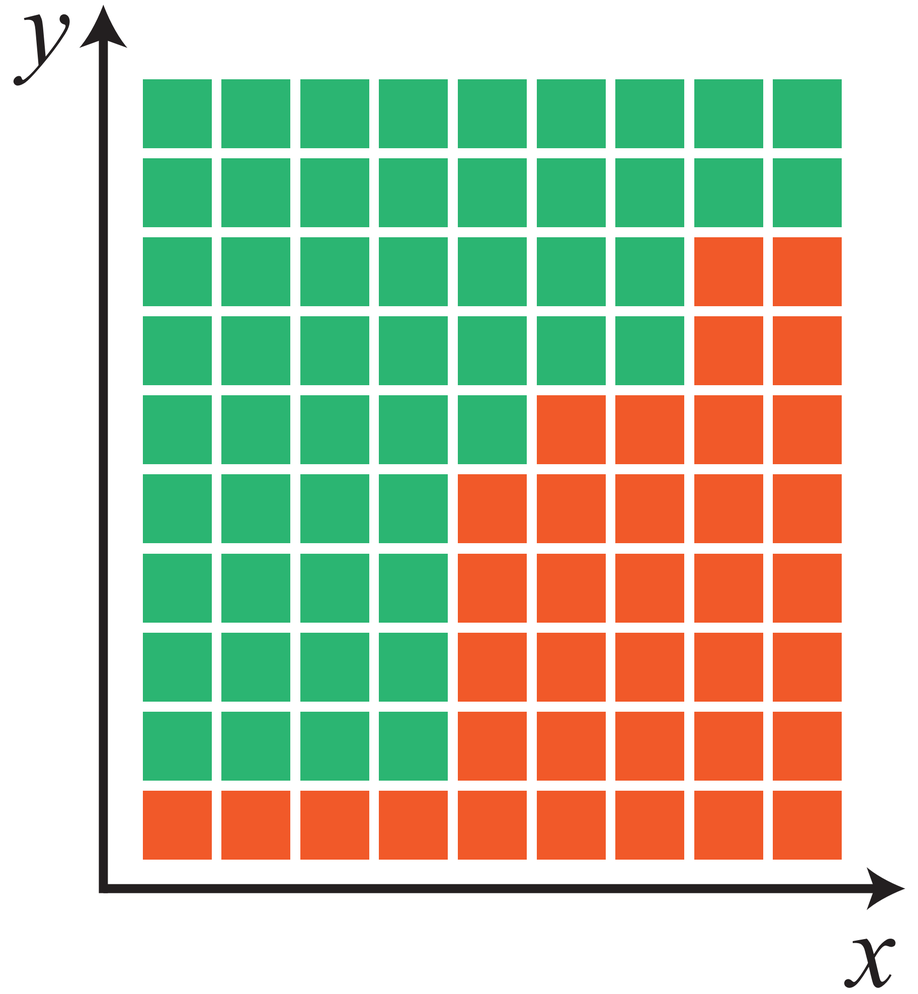
Challenge: Temporal Adaptivity



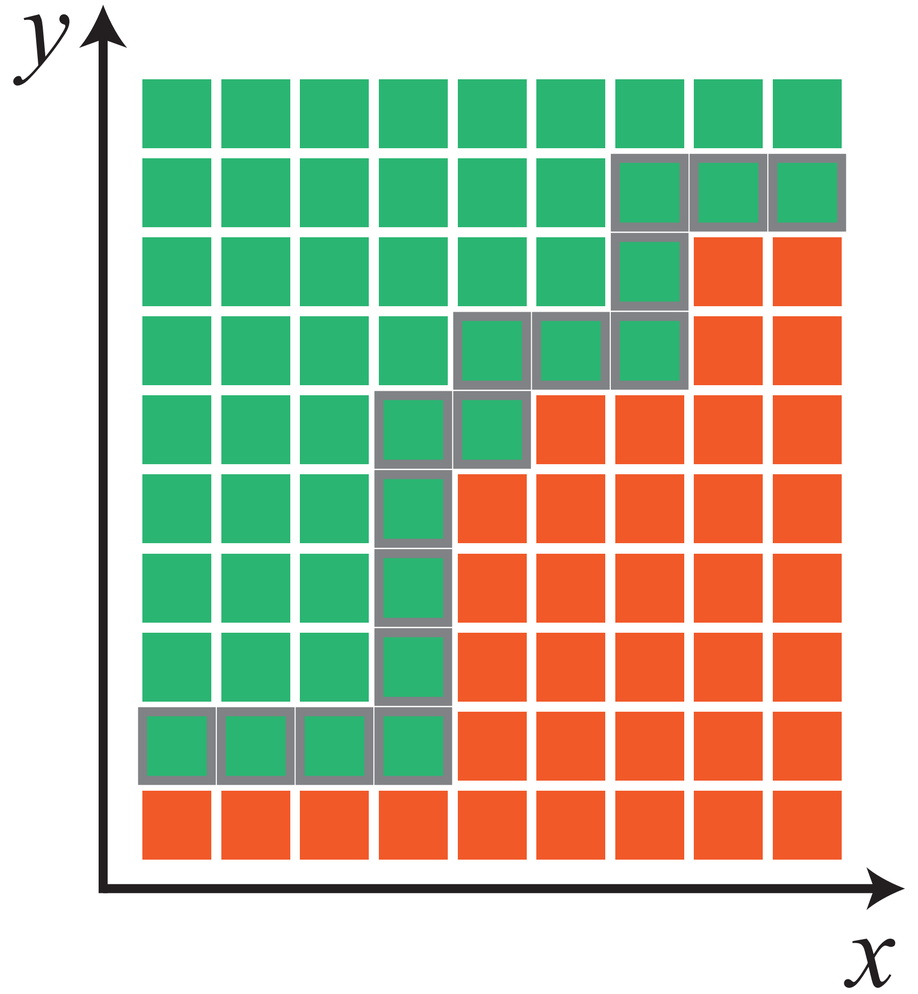




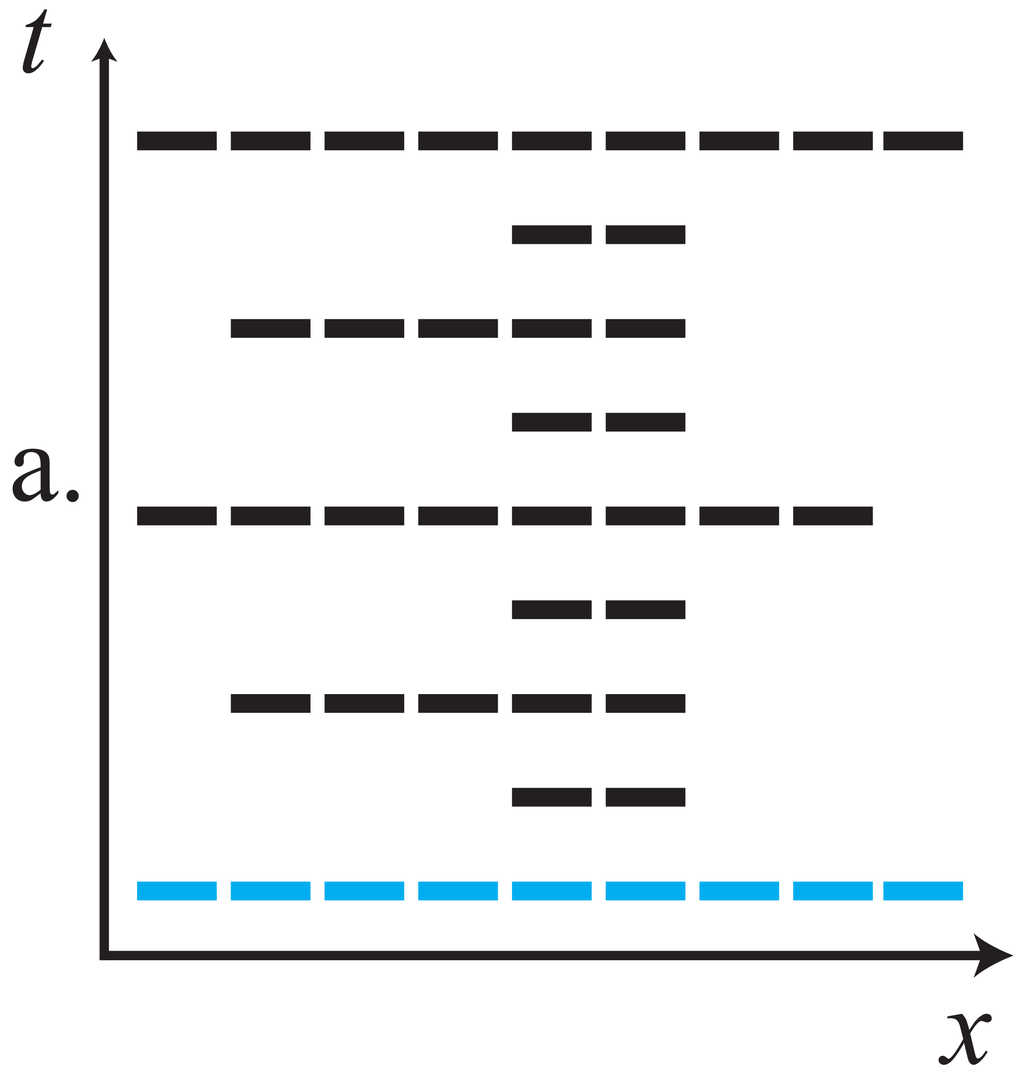
Challenge: Temporal Adaptivity



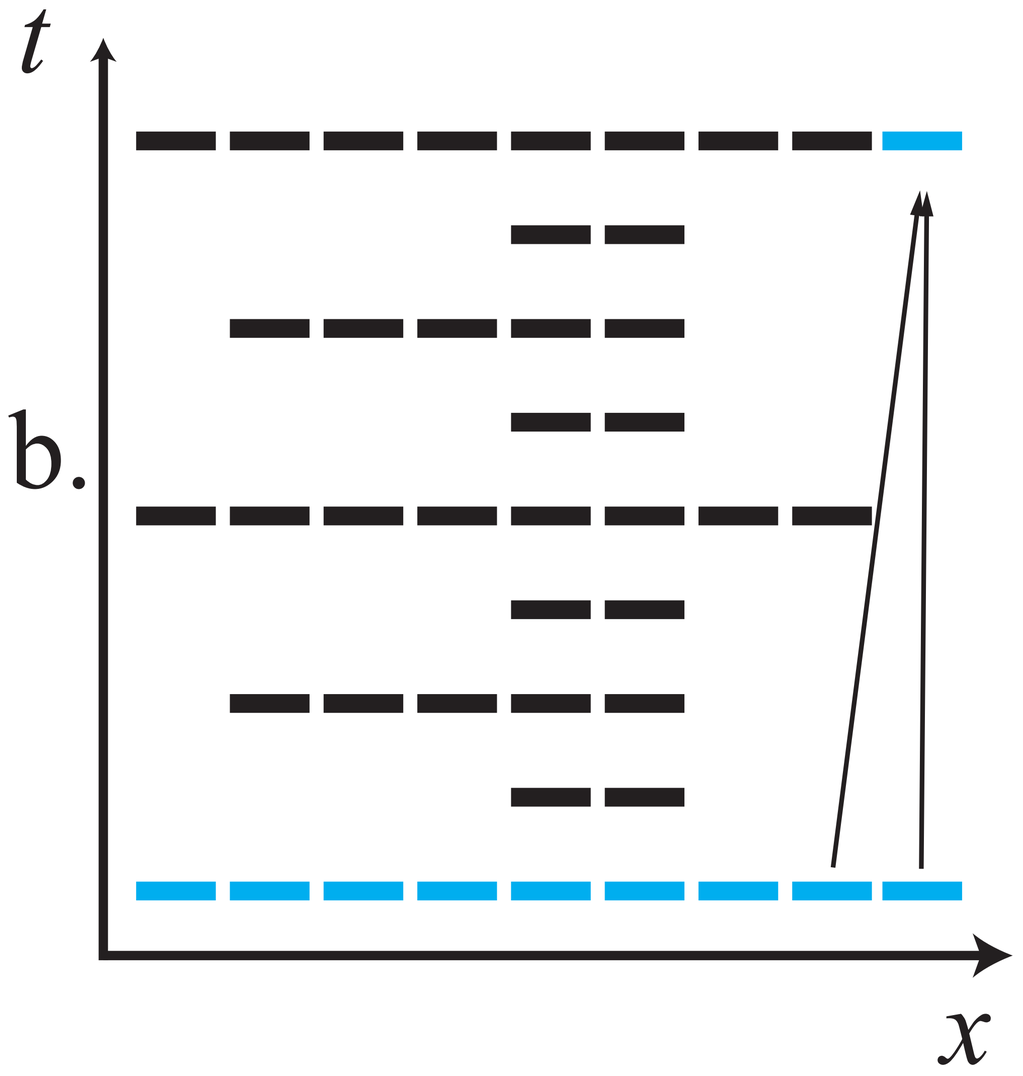
Challenge: Temporal Adaptivity



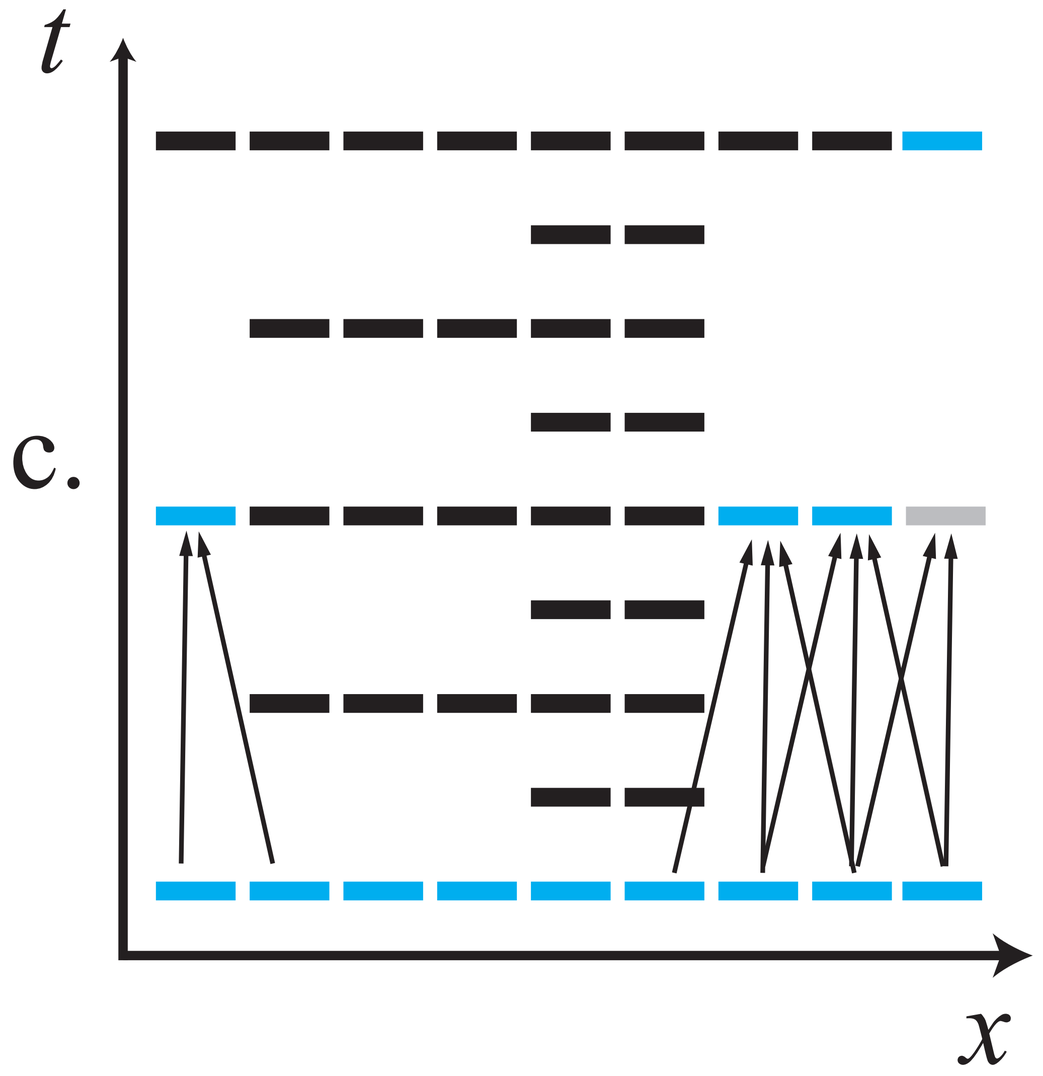
Solution: Block-Based Scheduling



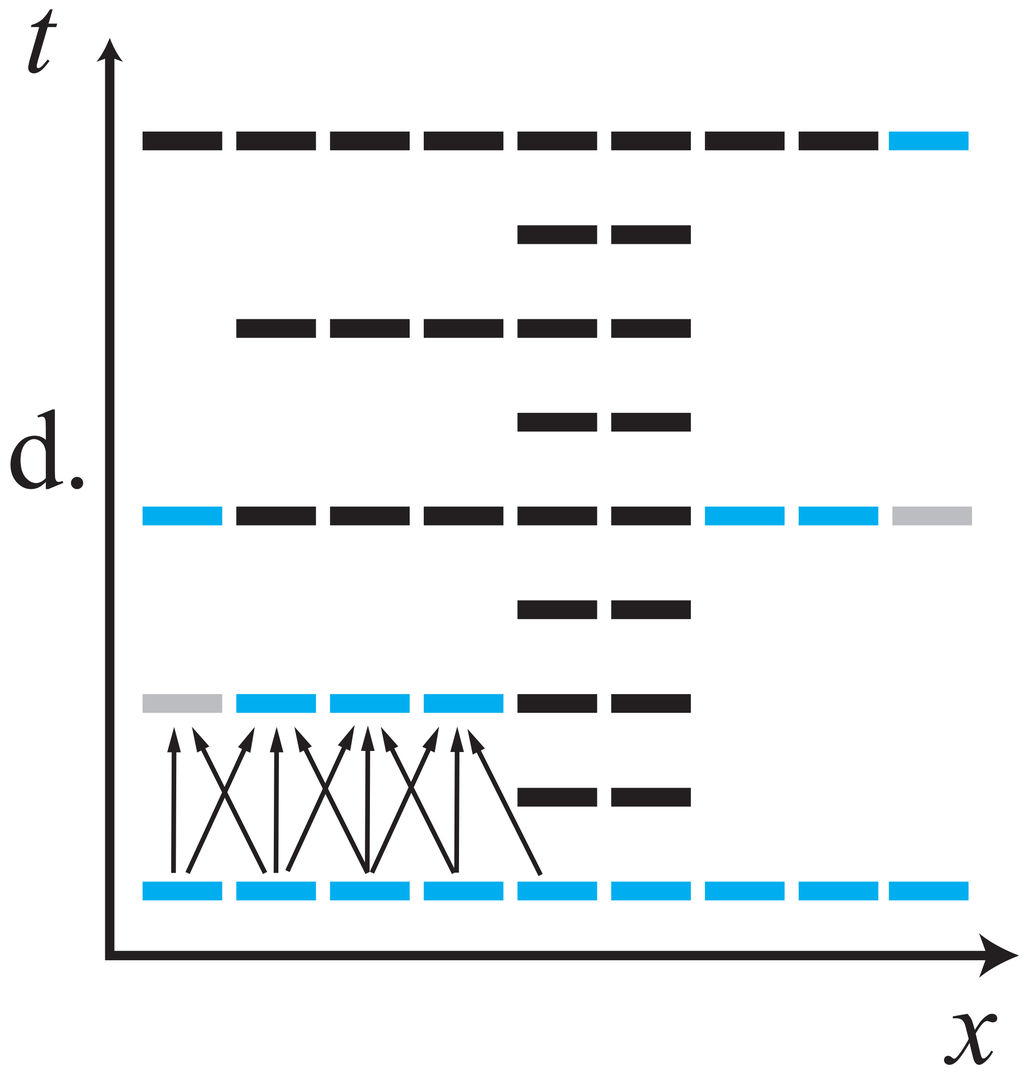
Solution: Block-Based Scheduling



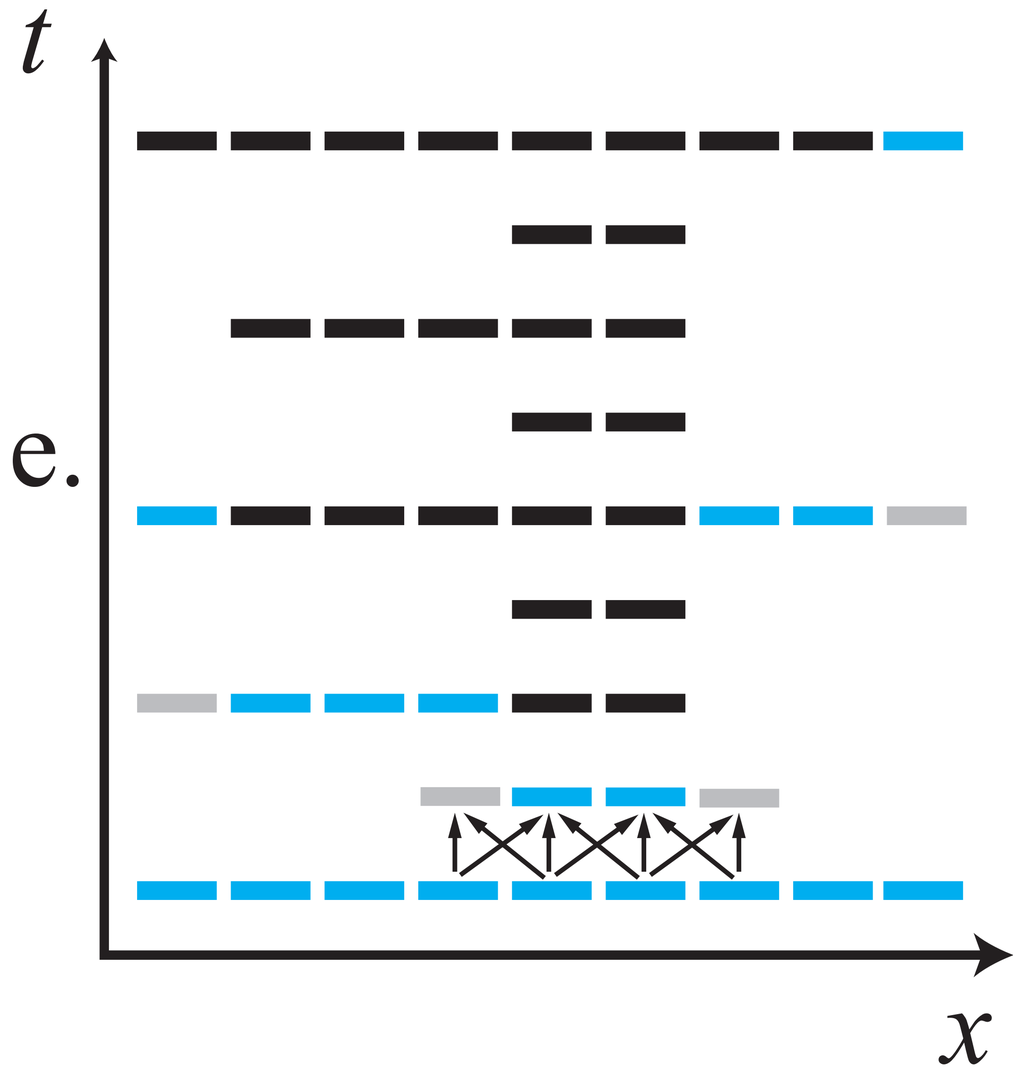
Solution: Block-Based Scheduling



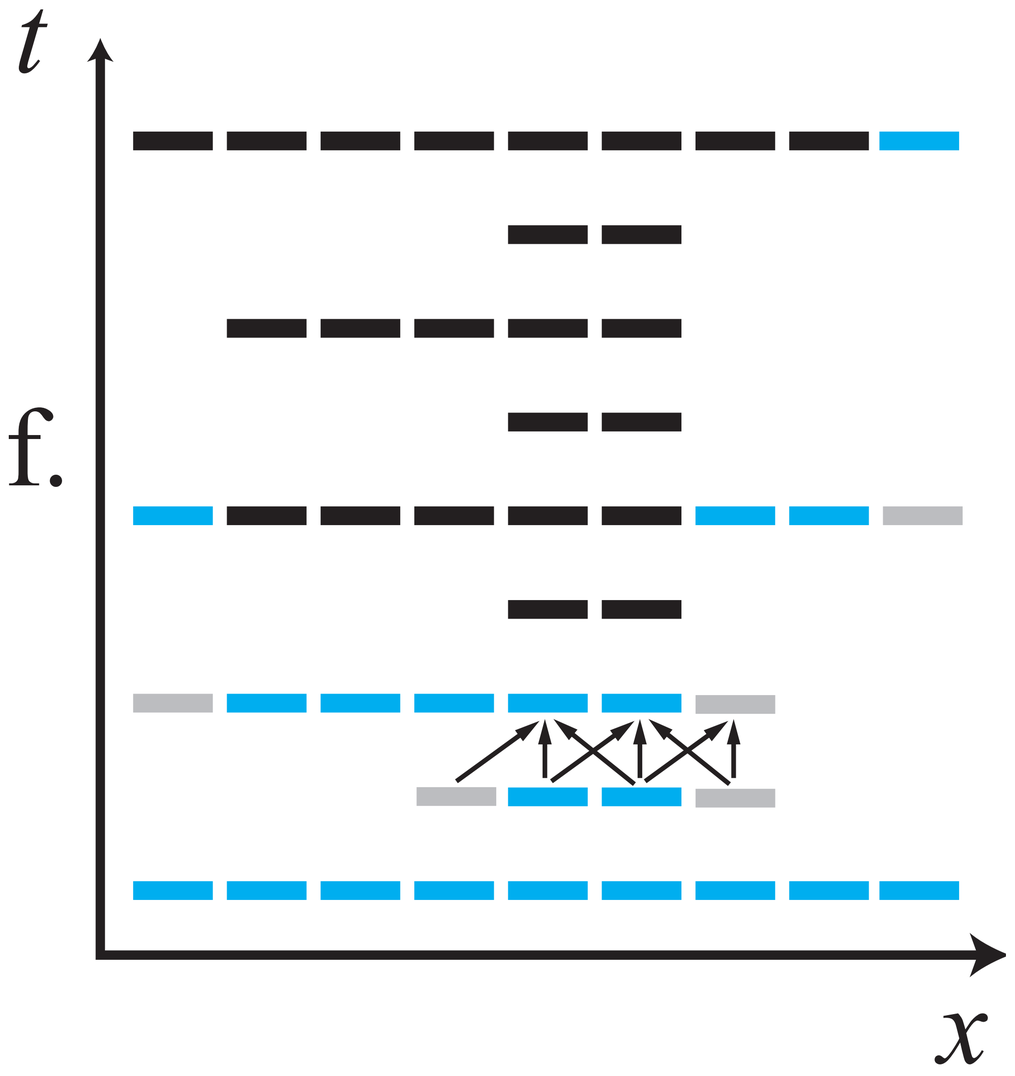
Solution: Block-Based Scheduling



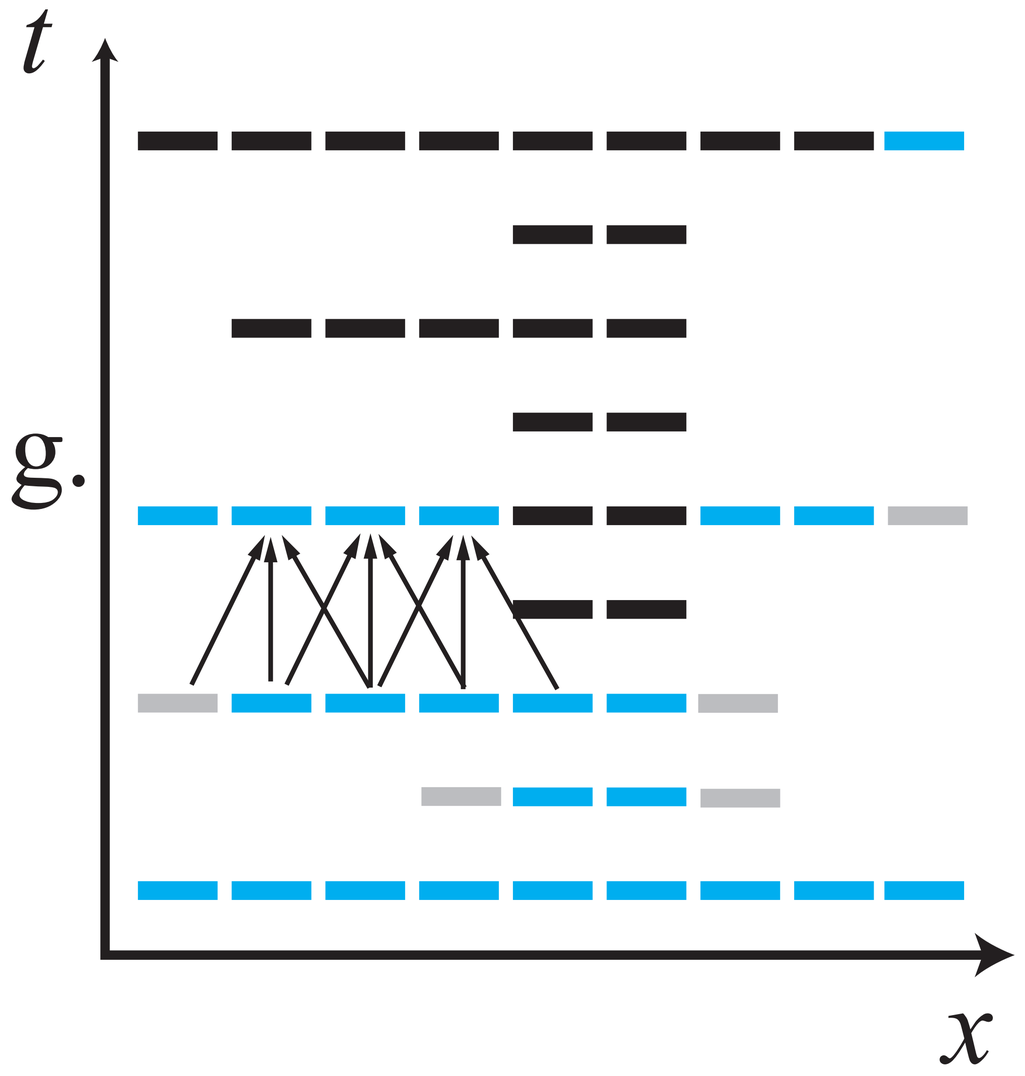
Solution: Block-Based Scheduling



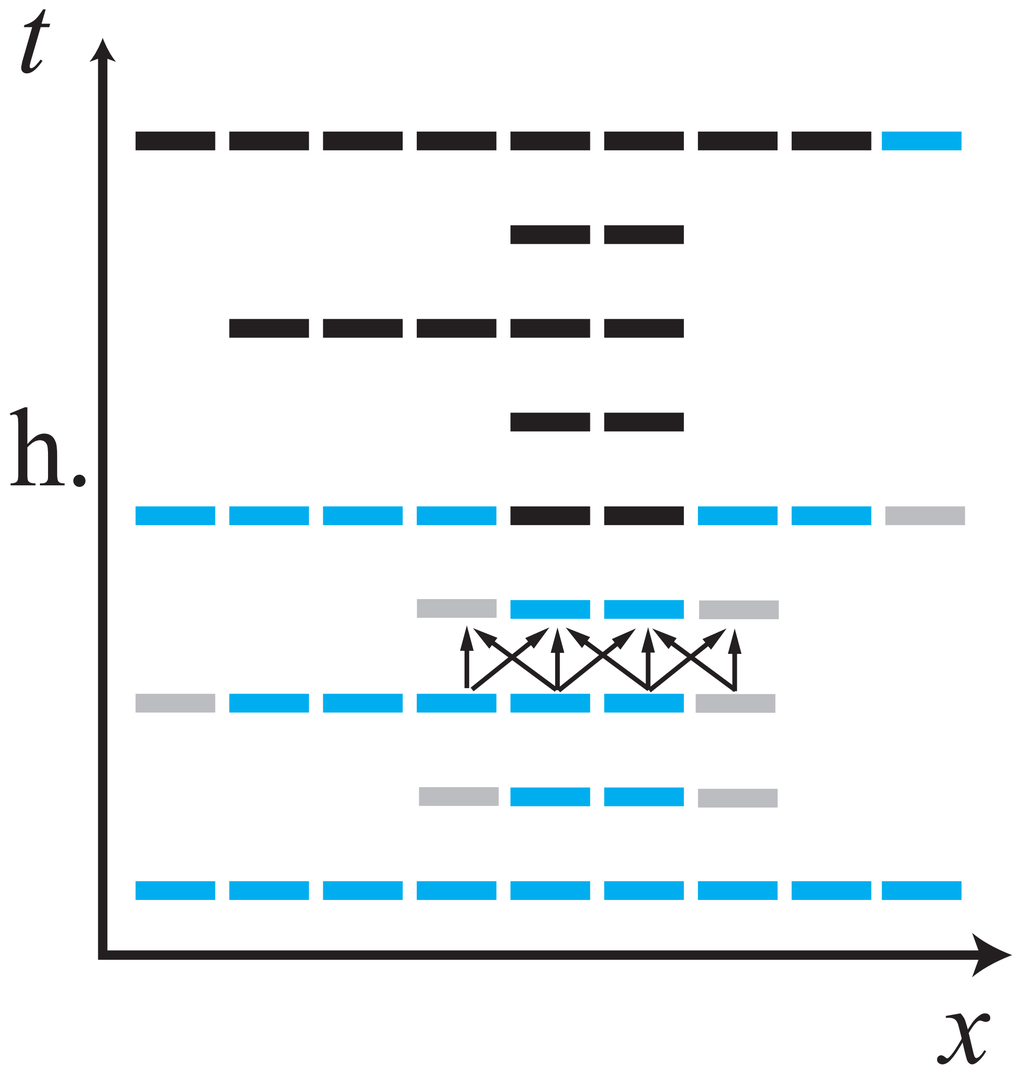
Solution: Block-Based Scheduling



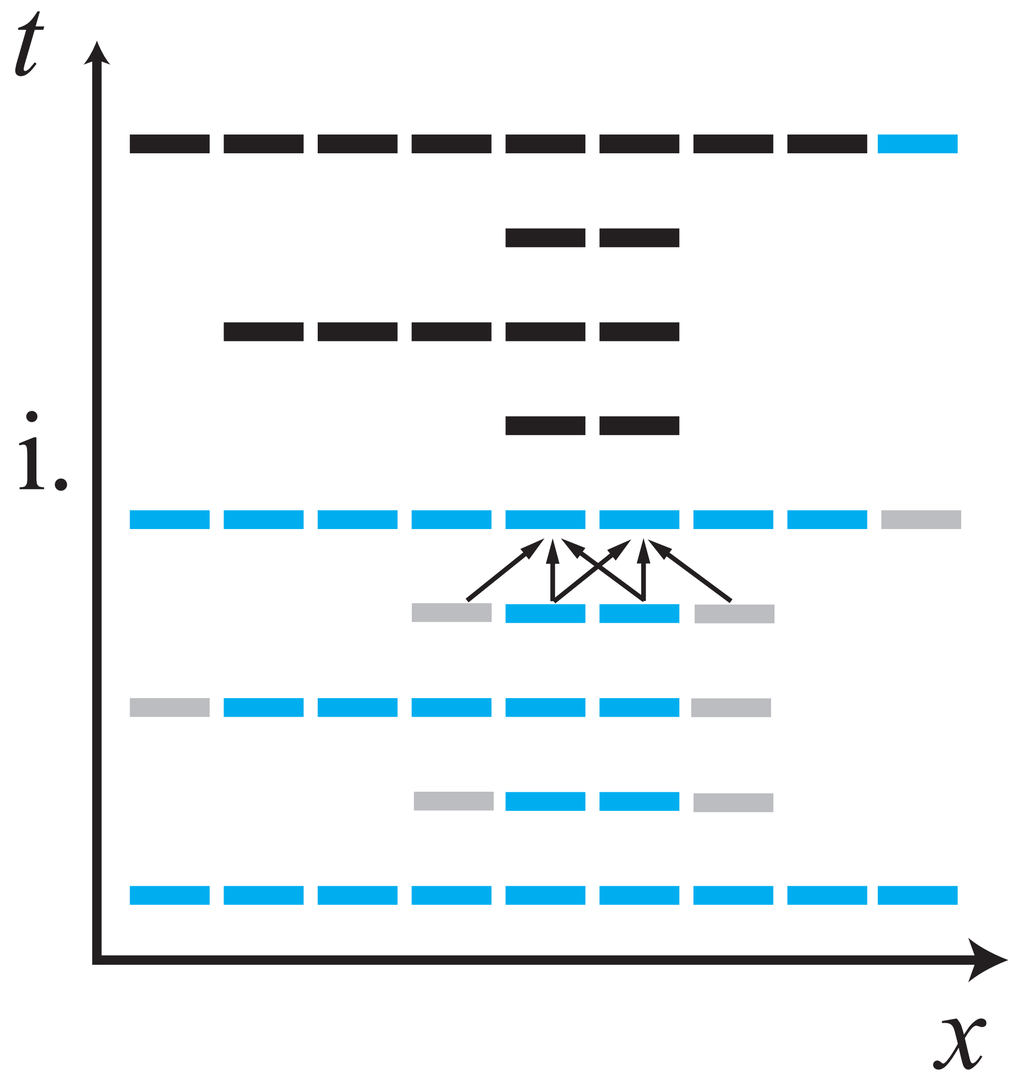
Solution: Block-Based Scheduling



Solution: Block-Based Scheduling



Solution: Block-Based Scheduling



Stork Couple

Particles: 1M

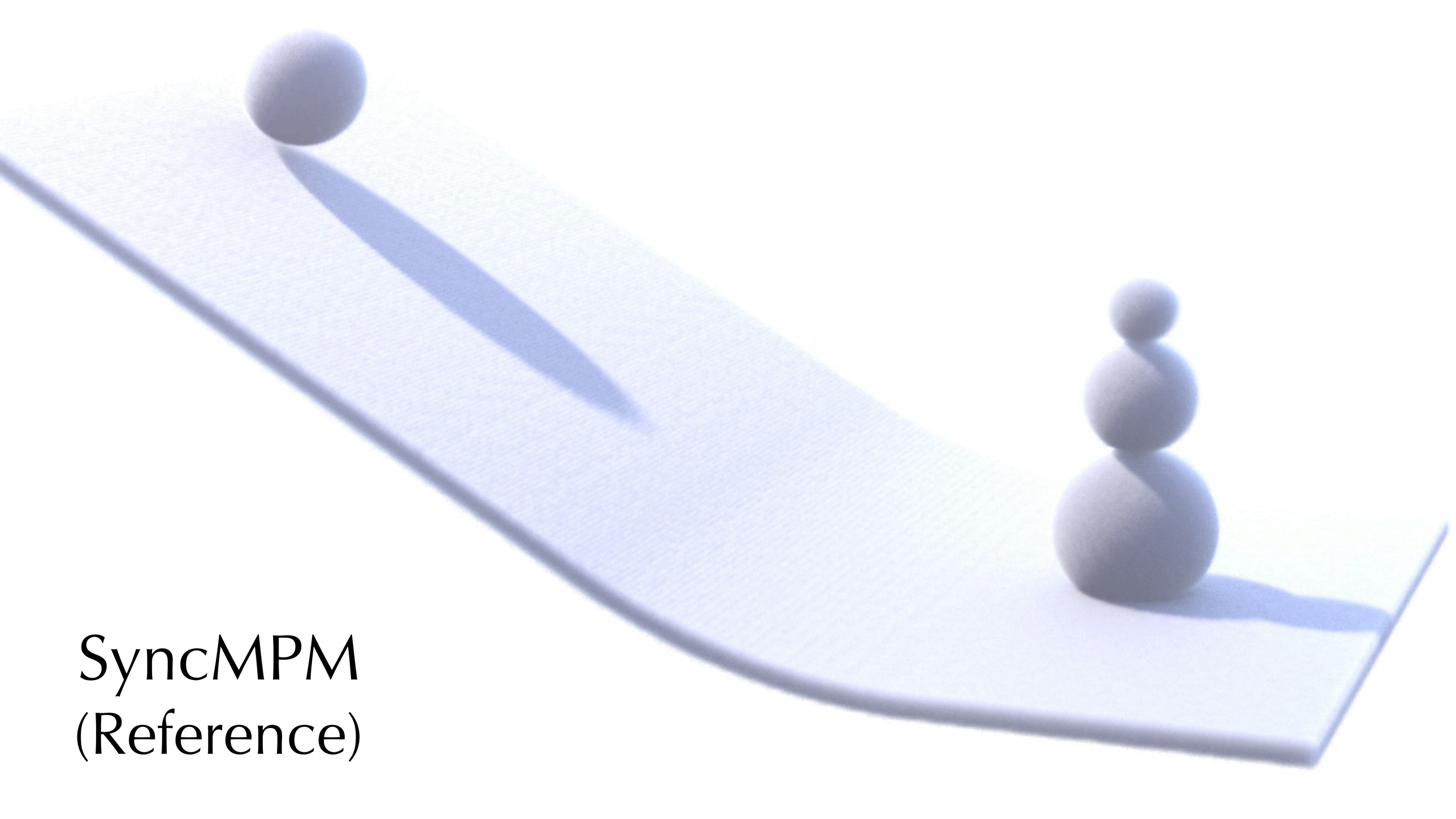
Async: 120 s/frame

Sync: 400 s/frame

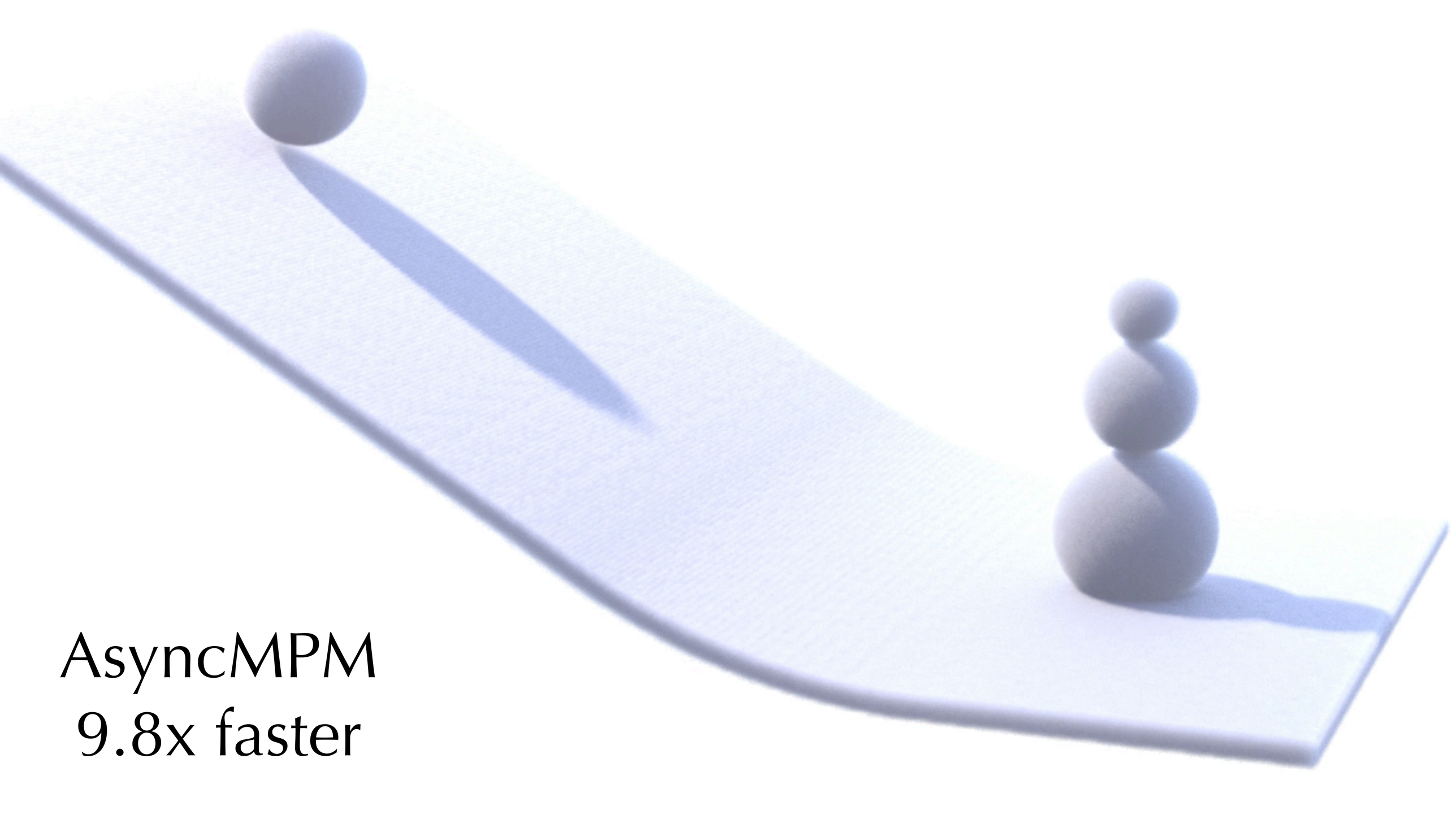
AsyncMPM

SyncMPM

Δt

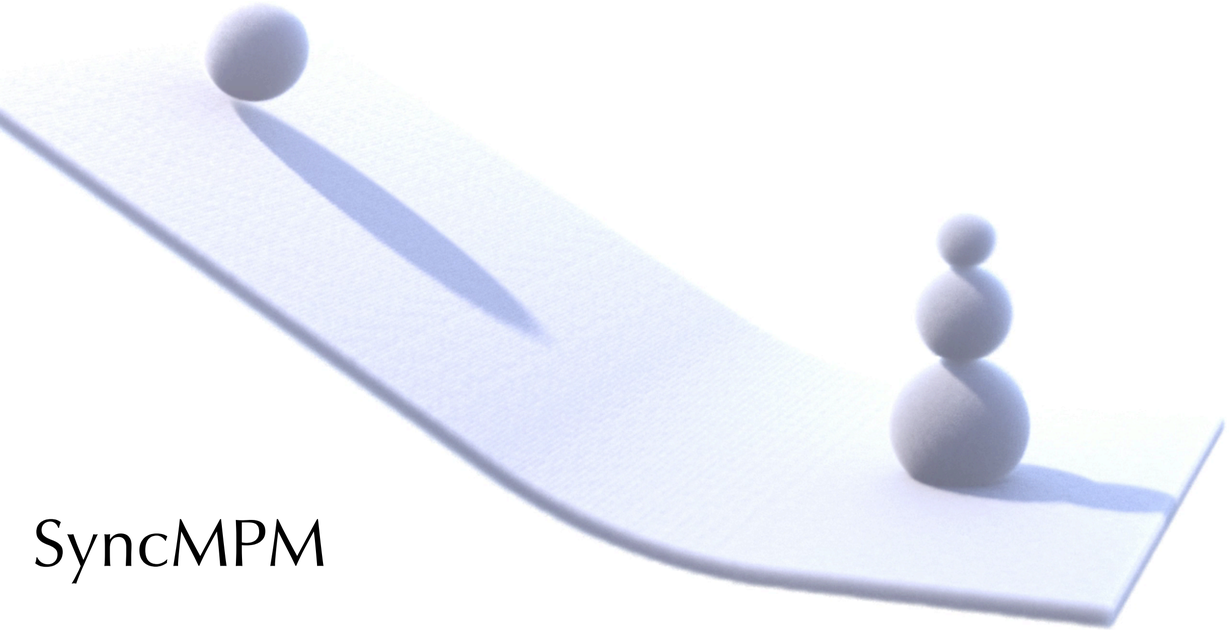


SyncMPM
(Reference)

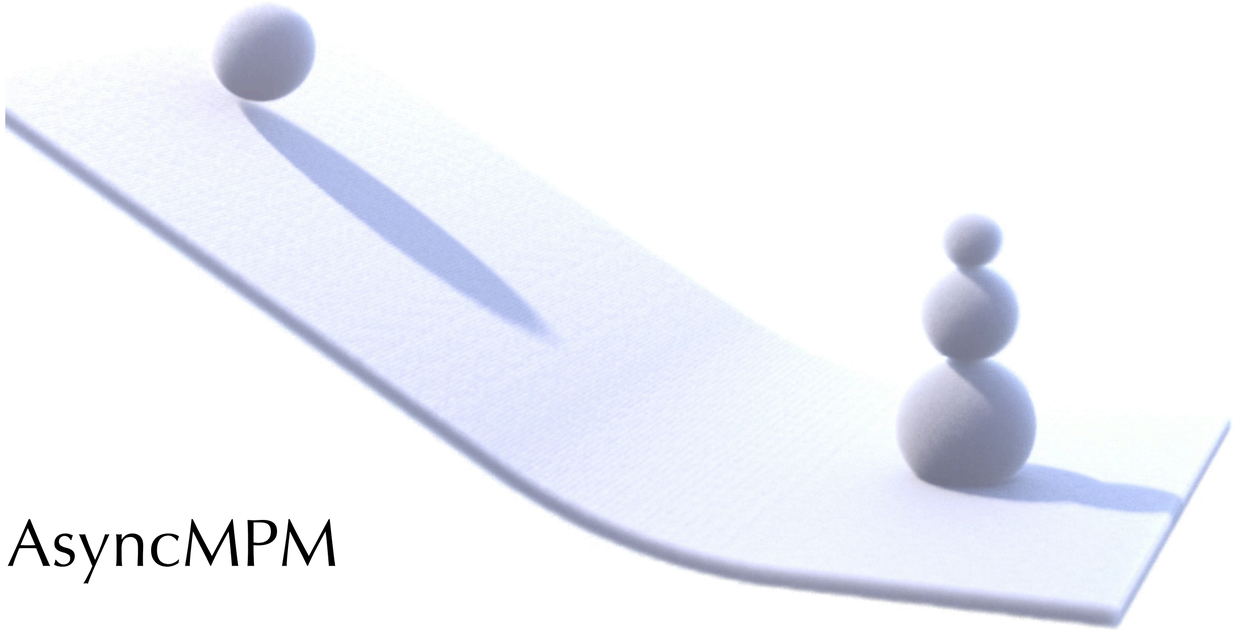

AsyncMPM
9.8x faster

Snow Slope

Side-by-side Comparison



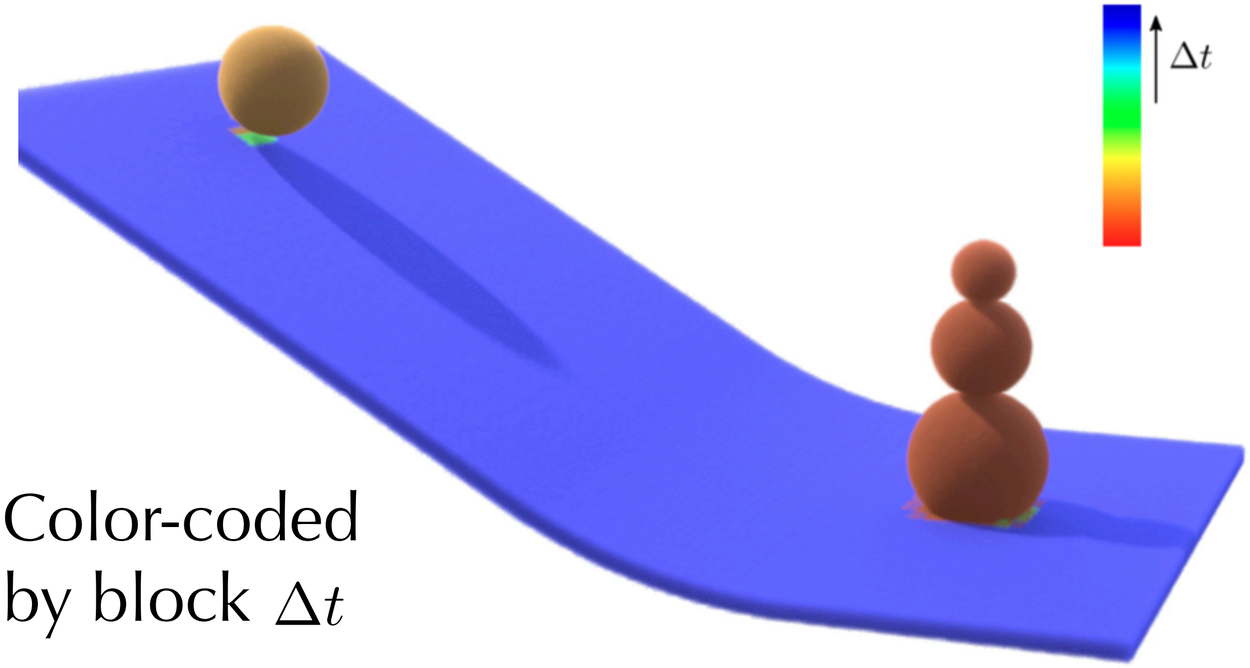
SyncMPM



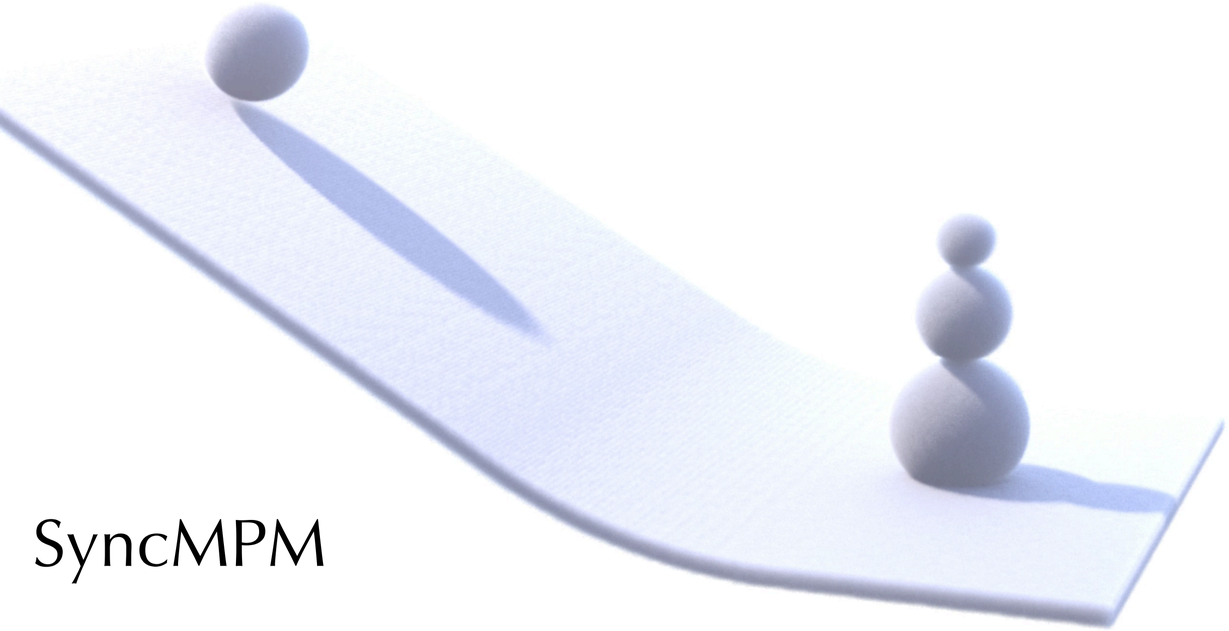
AsyncMPM

Snow Slope

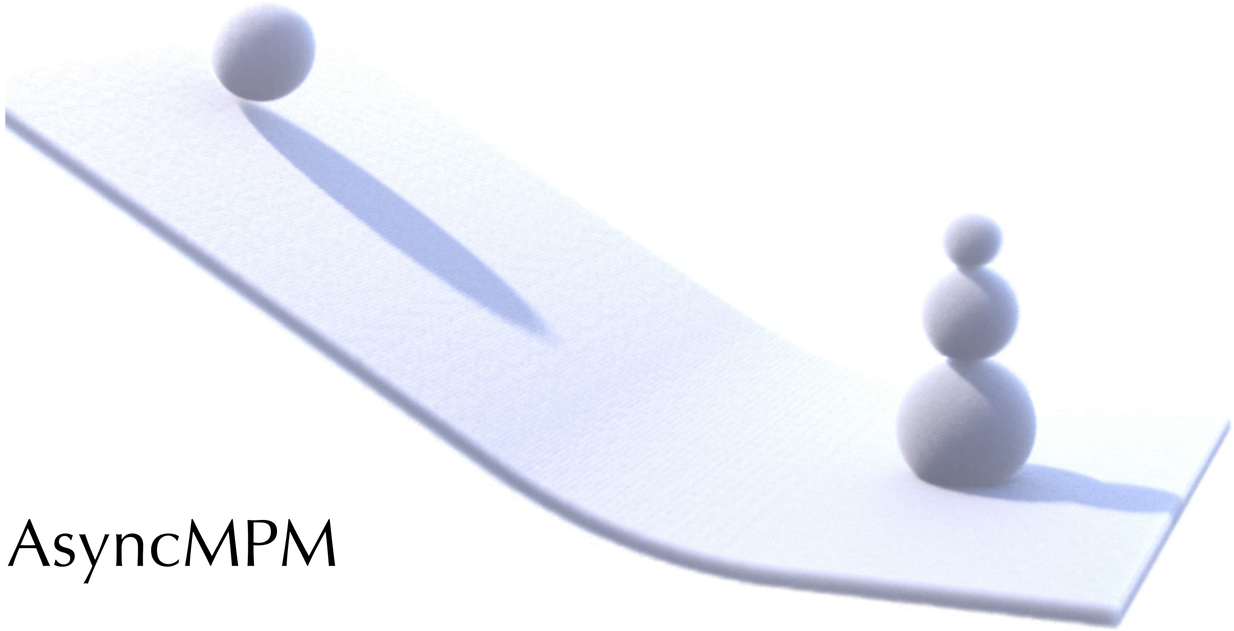
Side-by-side Comparison



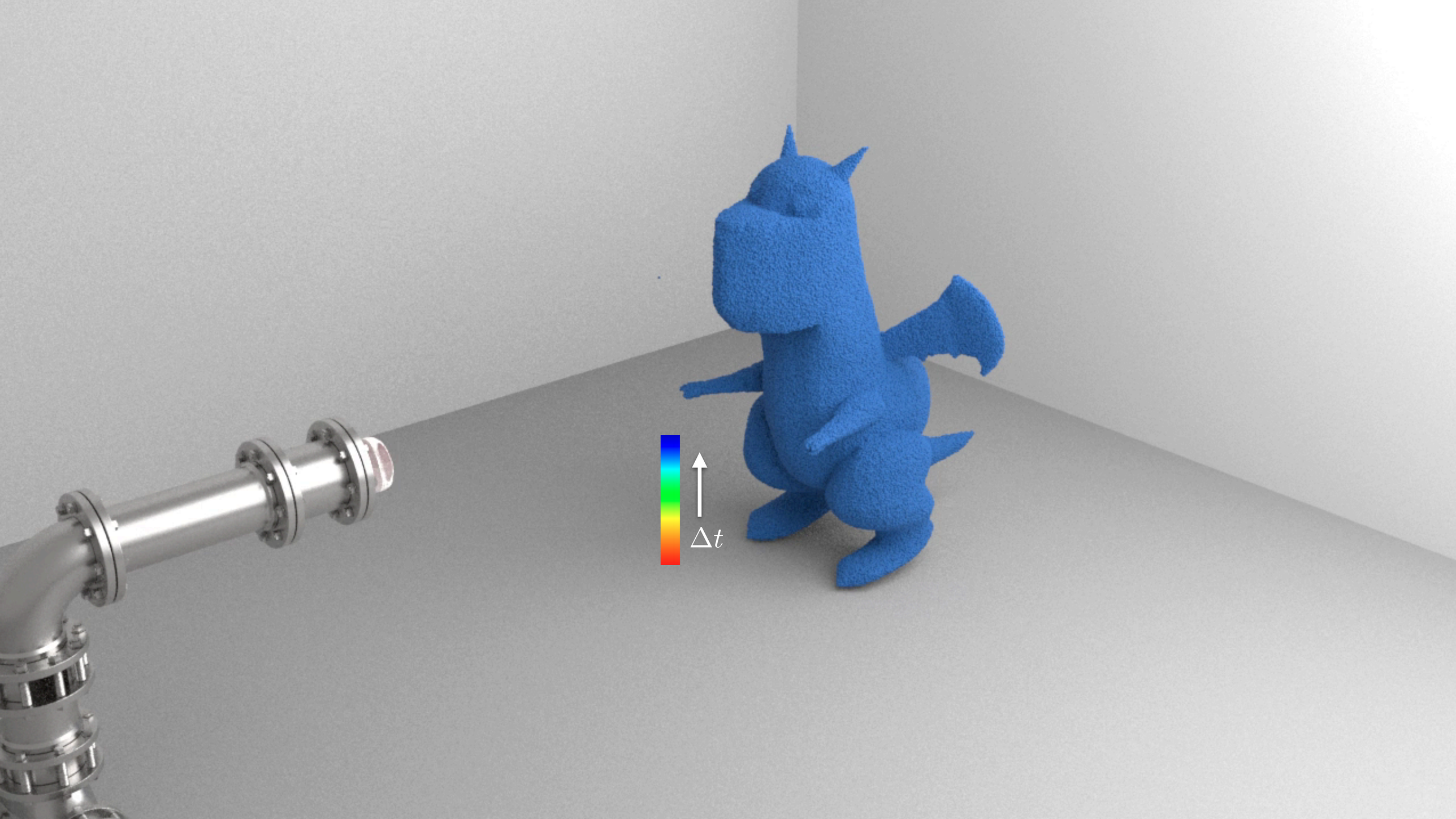
Color-coded
by block Δt

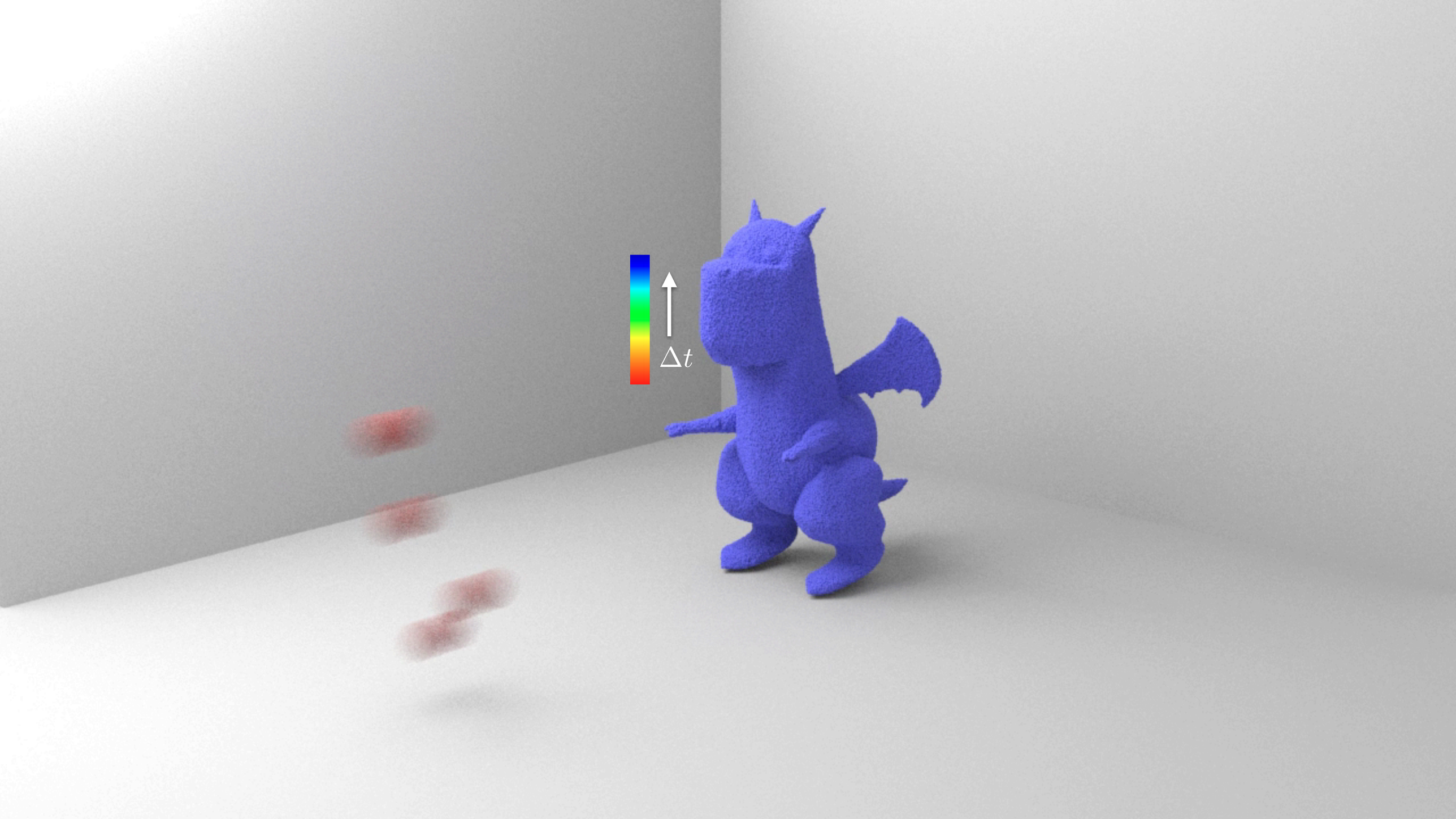


SyncMPM

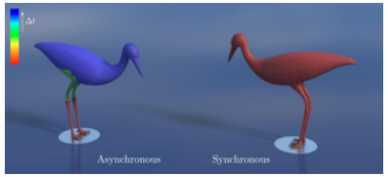


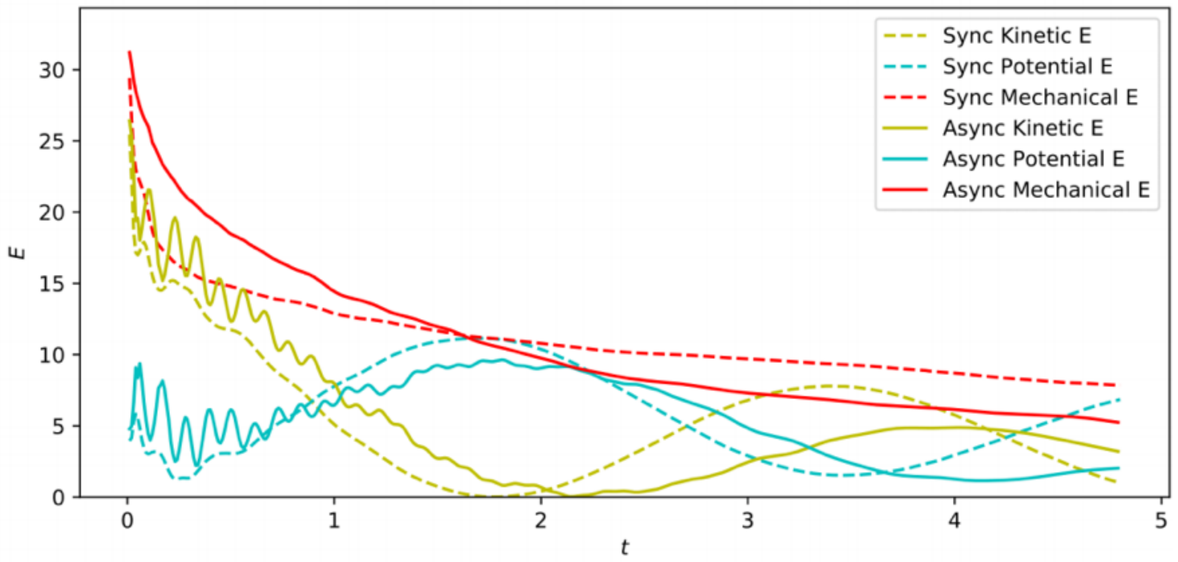
AsyncMPM

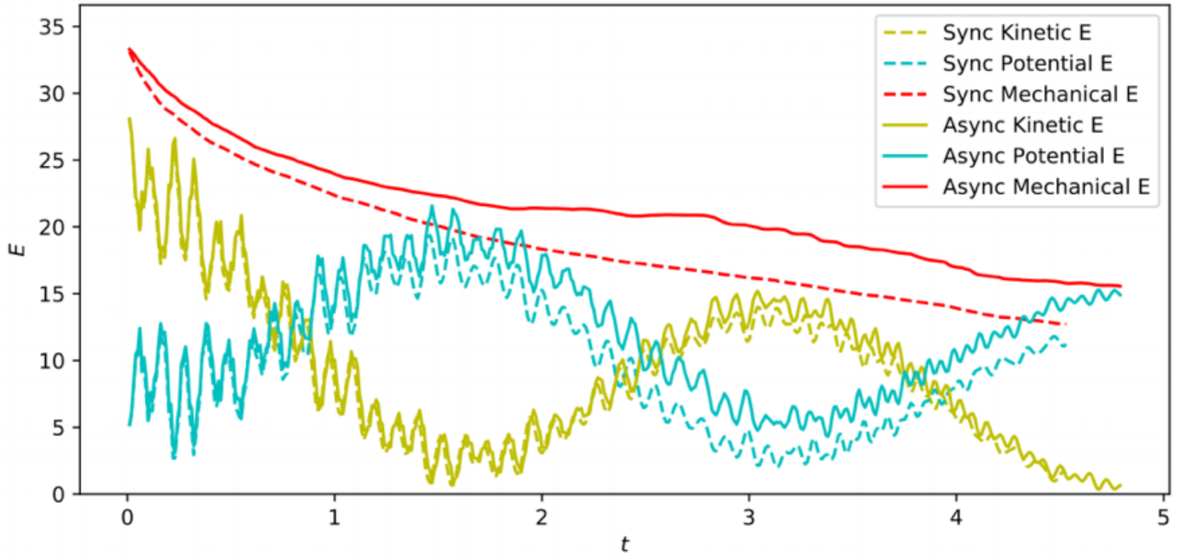

Δt


Δt

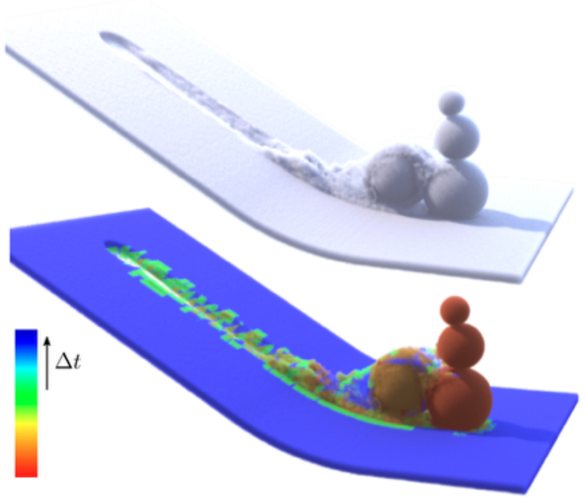
Energy Evolution

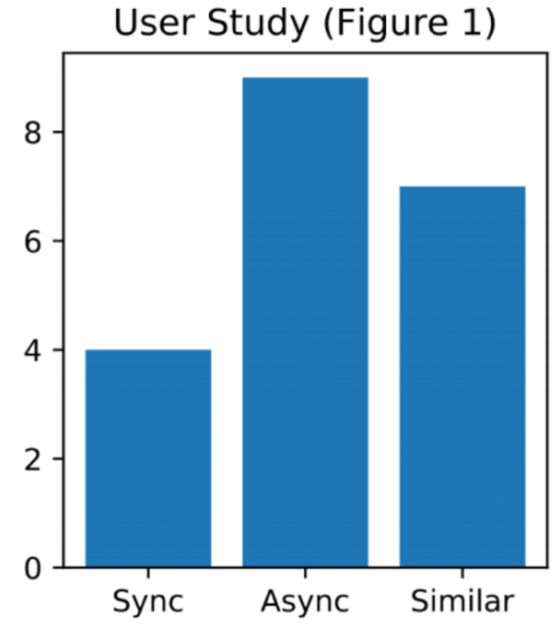






People like it!





Next Thursday:

Part I: Exposure: A White-Box Photo Post-Processing Framework
(ToG, SIGGRAPH 2018)

Part II: From TensorFlow to Taichi: Building the Computer Graphics Infrastructure

Contact: http://taichi.graphics/me

MPM Course: http://mpm.graphics

The End

Thank you!