Accurate Boundary Condition for Moving Least Squares Material Point Method using Augmented Grid Points

Riku Toyota and Nobuyuki Umetani

The University of Tokyo

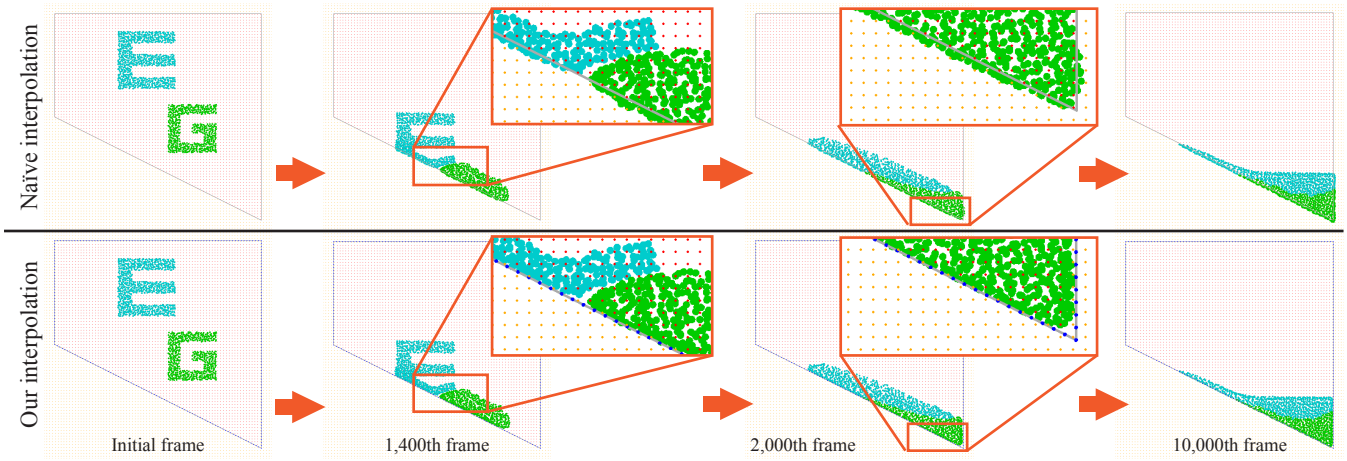

Figure 1: *Simulation of viscous fluid on an inclined plane with non-slip boundaries. (Top) The original MLS-MPM simulation had inaccurate boundary conditions causing particle sinking. (bottom), we augmented the background grid by adding points on the boundary (blue) to achieve more accurate boundary behavior without particle penetration.*

Abstract

This paper introduces an accurate boundary-handling method for the moving least squares (MLS) material point method (MPM), which is a popular scheme for robustly simulating deformable objects and fluids using a hybrid of particle and grid representations coupled via MLS interpolation. Despite its versatility with different materials, traditional MPM suffers from undesirable artifacts around wall boundaries, for example, particles pass through the walls and accumulate. To address these issues, we present a technique inspired by a line handler for MLS-based image manipulation. Specifically, we augment the grid by adding points along the wall boundary to numerically compute the integration of the MLS weight. These additional points act as background grid points, improving the accuracy of the MLS interpolation around the boundary, albeit with a marginal increase in computational cost. In particular, our technique makes the velocity perpendicular to the wall nearly zero, preventing particles from passing through the wall. We compare the boundary behavior of 2D simulation against that of naïve approach.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

The material point method (MPM) [SZS95, SSC*13] is widely used for simulating various deformable objects such as water, snow, and hyper-elastics. It robustly handles topological changes by discretizing deformable body using both particles and a regular grid. The particles and regular grids were coupled using alternating particle-to-grid and grid-to-particle interpolations. The moving least-squares MPM (MLS-MPM) [HFG*18] enhances traditional MPM simulations by interpolating grid point velocities for each particle using the least-squares method weighted by B-splines.

DOI: 10.2312/egs.20241022

However, typical MPM methods struggle with accurately defining boundary conditions related to rigid walls (e.g., full-slip or non-slip) because these boundary conditions are usually set on the grid points on the corner points in a regular grid. It is difficult to specify boundary conditions if the wall is misaligned with a regular grid. Additionally, in contrast to the particle-in-cell (PIC) method, MPM's interpolation schemes have a broader range of support (e.g., a 3×3 window for the quadratic weight) that exceeds the size of the grid cell. This leads to boundary influence being spread out, resulting in artifacts in which particles are trapped inside the wall(see Figure 1) or slide on the nonslip wall.

This study introduces a novel method for accurately handling boundary conditions. Our method was inspired by an MLS-based image-deformation technique using line handlers [SMW06]. While this technique initially computes the line integral for MLS interpolation weight determination, we replace the line integral with the sum of the discrete points on the boundary (i.e., the boundary grid points). In essence, we sample points along the boundaries at intervals matching the grid size and incorporate them as additional grid points(i.e., boundary grid points). Owing to the nature of the MLS-MPM, we can flexibly formulate the interpolation from both points in the regular grid (i.e., regular grid points) and boundary grid points. Our method simulates various materials' deformation while avoiding the particles penetrating into walls.

2. Related Work

Particle-in-cell Similar to the MPM, the particle-in-cell (PIC) method [ZB05] connects the background grid and particles to simulate the fluid accurately. In PIC methods, accurate boundary representation is possible using a cut-cell mesh [TBBC*22]. However, applying a cut-cell approach to MPM is challenging due to MPM's interpolation extending beyond a single cell; that is, the influence of interpolation extends to several adjacent cells.

Material point method The conceptual foundation of MPM is commonly attributed to Sulsky et al. [SZS95] who extend the PIC method. Stomakhin et al.'s groundbreaking work on snow simulation [SSC*13] introduced MPM to computer graphics. Since then, various improvements have emerged, such as a temporally adaptive scheme for multi-material scenes [FHHJ18] and the integration of GPU acceleration [GWW*18].

An innovative contribution from Hu et al. [HFG*18] introduced moving least-squares interpolation [SMW06] for MPM. This least-squares interpolation is more flexible than the interpolation on a regular grid, allowing us to augment the regular grid to increase the resolution of the velocity field around the boundary.

Boundary handling for MPM Hu et al. [HFG*18] extend MLS-MPM by introducing compatible particle-in-cell (CPIC), designed for material cutting and coupling with rigid bodies. This approach incorporates a colored distance field (CDF) that encodes the minimum distance from a grid point to the boundaries, the set of nearby surfaces, and the side on which the grid point is located. Similar to our method, the CPIC distributes points on the boundaries. However, CPIC treats these boundary points as a representation of rigid body surfaces, while our method regards them as augmented grid points. Additionally, CPIC applies penalty forces to prevent particle penetration of boundaries.

In pursuit of enhanced simulation accuracy, Gao et al. [GTJS17] proposed a spatially adaptive-resolution MPM. This method strategically refines the particles and background grid in the proximity of the boundaries. While effective in achieving accurate boundary handling, this approach incurs a significant computational cost.

3. Method

Traditional MPM methods utilize a regular background grid to depict a continuous velocity field, from which velocity gradients are derived. However, the interpolation of the MPM has wide support (e.g., $2h \times 2h$ for a quadratic weight function, where h is the grid interval), and the influence of the grid points may extend inside the wall (see Figure 2-right), leading to particle penetration.

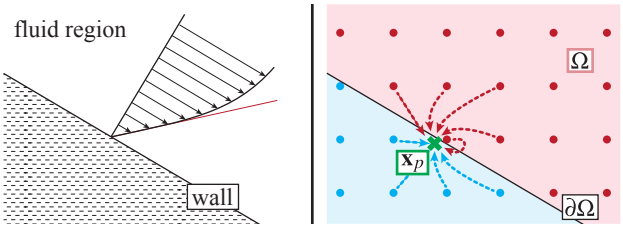

Figure 2: *Left: the linear velocity changes of laminar flow around the non-slip boundary. Right: the typical interpolation of MPM fails to achieve boundary condition as the value at $\mathbf{x}_p$ interpolates from values of the fluid region Ω around the boundary $\partial\Omega$.*

Desirable interpolation property One of our target materials is a viscous fluid, which develops a laminar boundary layer on non-slip walls. It is well known that the velocity *linearly* changes according to the distance to the non-slip wall near the boundary (see Figure 2-left). Prominent flow scenarios like Couette flows and laminar flows down inclined planes display such linear behavior. Our objective is to ensure that the interpolation satisfies the linearly changing velocity around the wall.

MLS interpolation We first briefly explain the existing MLS interpolation roughly following the notations used in [HFG*18]. For simplicity, we explain the method in the 2D case; however, the notation can easily be generalized to 3D cases. The (Linear) MLS interpolation approximates the scalar value u around location $\mathbf{x}_p$ as a linear field:

$$u_p(\mathbf{x}) = \mathbf{c}_p \mathbf{P}(\mathbf{x} - \mathbf{x}_p), \tag{1}$$

where $\mathbf{P}(\mathbf{x}) = [1, x_1, x_2]^T$ is the bases of the polynomial and $\mathbf{c}_p = [u_p, \partial u_p/\partial x_1, \partial u_p/\partial x_2]$ is their coefficients corresponding to the estimation of value and its gradients at $\mathbf{x}_p$. The coefficients $\mathbf{c}_p$ can be obtained by minimizing the weighted squared errors between the data points $\mathbf{x}_i$ and the linear field $u_p(\mathbf{x})$.

MLS interpolation from line Our goal here is to make $u_p(\mathbf{x})$ (i.e., the linear approximation of u around the $\mathbf{x}_p$) take the given value $\bar{u}$ at the boundary $\partial\Omega$. To this end, we penalized the errors on the

boundary by introducing a penalty term in the MLS interpolation. Inspired by image deformation using the line handler in [SMW06], we *integrate* the weighted squared errors on the boundary assuming that the data points $\mathbf{x}_i$ are also distributed on the boundary

$$J_{BC}(\mathbf{c}_p) = \int_{\mathbf{x}_i \in \partial\Omega} w(\mathbf{x}_i - \mathbf{x}_p) \left\| u_p(\mathbf{x}) - \bar{u}(\mathbf{x}_i) \right\|^2 d\mathbf{x}_i, \qquad (2)$$

where $w(\mathbf{x})$ is a weight function. This error is added to the traditional interpolation errors of MLS and is jointly minimized by optimizing the coefficient $\mathbf{c}_p$.

Weight function We adopt the same weight function for the boundary integral (2) as in MPM. Whereas [SMW06] used a weight function with infinitely wide support, the support of the MPM weight function covers the range of several grid points. Since the direct influence of boundary conditions on kinematic simulation is confined to immediate neighbors, a smaller support is more suitable. In contrast to the *bounded* weight function of the MPM, the weight function in [SMW06] is infinitely large at the point of interpolation $\mathbf{x}_p$ to *exactly* achieve the boundary conditions. However, this sharp weight function profile leads to abrupt changes in interpolated values around the boundary, making it unsuitable for achieving linear velocity changes.

We exclude grid points within the region near the boundary where the distance to the wall is below a certain threshold δ, excluding those outside the fluid region. By ignoring the grid points located inside and near the wall, the boundary integral in (2) acts as a "*hinge*" allowing the interpolated field to change linearly while satisfying the fixed velocity at the boundary. This also aids in reducing the CFL condition by keeping the distance between grid points above the threshold δ.

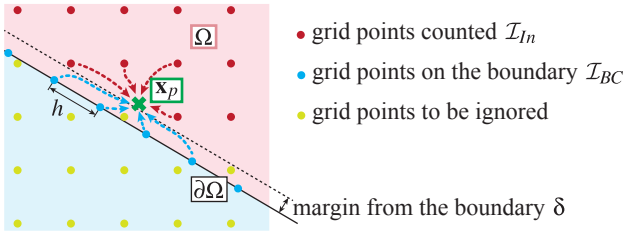

Figure 3: *We interpolate value at the location $\mathbf{x}_p$ from points inside the region Ω with the distance from the wall greater than δ and the points on the boundary.*

Boundary discretization Theoretically, the boundary integral in (2) could be computed analytically due to the simple polynomial nature of the MPM weight function. However, the integration is very complicated because the integration is on the line while the integrand is defined on the rectangular domain. Instead, we opt to integrate it *numerically* by replacing line integration with the summation of uniformly sampled points on the boundary $\mathcal{I}_{BC}$. We used the grid interval h of the regular grid for the internal points on the boundary. The resulting minimization target for computing coefficient $\mathbf{c}_p$ becomes

$$J(\mathbf{c}_p) = \sum_{i \in \mathcal{I}_{In} \cup \mathcal{I}_{BC}} w(\mathbf{x}_i - \mathbf{x}_p) \left\| u_p(\mathbf{x}) - \bar{u}(\mathbf{x}_i) \right\|^2, \qquad (3)$$

where $\mathcal{I}_{In}$ is the set of grid points located inside region Ω where the distance to the wall is longer than the threshold δ. In this study, we set the threshold $\delta = 1/4h$. This threshold is selected such that the weight matrix is not singular in Ω.

Interpolation for MPM At particle point $\mathbf{x}_p$, MPM interpolates the physical quantity u and estimates its gradient from grid point $\mathbf{x}_i$. As discussed in [HFG*18], MLS-based interpolation can be expressed as

$$\mathbf{c}_p^T = \begin{bmatrix} u_p \\ \nabla u_p \end{bmatrix} = \mathbf{M}_p^{-1} \sum_i \mathbf{P}(\mathbf{x}_i - \mathbf{x}_p) w(\mathbf{x}_i - \mathbf{x}_p) u_i, \qquad (4)$$

where $\mathbf{M}_p$ is a 3×3 coefficient matrix, written as

$$\mathbf{M}_p = \sum_i w(\mathbf{x}_i - \mathbf{x}_p) \mathbf{P}(\mathbf{x}_i - \mathbf{x}_p) \mathbf{P}^T(\mathbf{x}_i - \mathbf{x}_p). \qquad (5)$$

The interpolated value u can be the density (for fluid simulation in [MPM]) or each component of velocity and momentum.

The original MLS-MPM study [HFG*18] formulated forces on the vertices of a *regular grid*. We derive the force on the vertex of the augmented grid, where the vertices can adopt arbitrary configurations

$$\mathbf{f}_i = -\sum_p V_p \sigma_p \begin{bmatrix} \bar{m}_{21} & \bar{m}_{22} & \bar{m}_{23} \\ \bar{m}_{31} & \bar{m}_{32} & \bar{m}_{33} \end{bmatrix} \mathbf{P}(\mathbf{x}_i - \mathbf{x}_p) w(\mathbf{x}_i - \mathbf{x}_p), \qquad (6)$$

where $\mathbf{M}_p^{-1} = (\bar{m}_{ij})_{1 \le i \le 3, j \le j \le 3}$, V_p is the particle volume and σ is the 2×2 Cauchy stress tensor at $\mathbf{x}_p$. Note that in the case of a regular grid, $\bar{m}_{21}$ and $\bar{m}_{31}$ become zero, and (6) is the same as that in [HFG*18].

4. Results

Implementation detail We implemented demonstration codes in Rust language based on Taichi [HLA*19]'s 88-line MLS-MPM code [Tai]. Following a web article [MPM], we developed an MPM fluid solver by estimating particle density by scattering particle masses onto grid points using MPM interpolation weights. Note that this MPM interpolation weight must be computed on the original regular grid, not on our augmented grid, as there is no boundary condition for the density. Our code is publicly available †.

Interpolation test As detailed in Section 3, the accuracy of the non-slip boundary condition depends on the representation capability of the linear field around the boundary. Figure 4 compares the interpolated distance field from the distance at the grid points. Utilizing the augmented grid points, our interpolation better represents the distance field compared to the naïve MPM interpolation.

2D Simulation comparison Figure 1 compares the naïve regular grid interpolation against our augmented grid interpolation for fluid simulation with non-slip boundary conditions. Figure 5 showcases simulations with full-slip boundary conditions for various material types, including viscous fluids (the Navier-Stokes model based on [MPM]), hyperelastic material (the St. Venant-Kirchhoff

† https://github.com/nobuyuki83/accurate_bc_for_mls_mpm

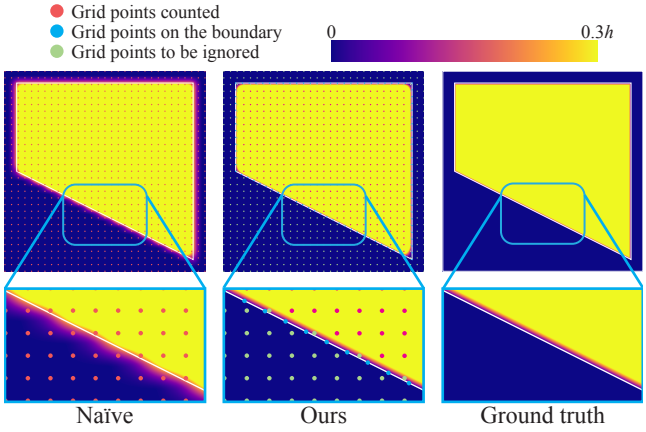

Figure 4: *Reconstructions of distance fields from the distances computed at the grid points.*

model), and plastic materials (the snow model in [SSC*13]), where the simulation settings (time steps, boundary shapes) are the same as those in Figure 1. Note that all the simulations adopt explicit time integration. The full-slip boundary condition was implemented by zeroing the normal component of boundary grid point velocities. In all cases, our method accurately handled the boundary conditions to prevent undesirable particle penetration. For further insight, please refer to the supplementary video of the animation and the comparison with circular boundary geometry.

Performance The simulation in Figure 1 takes 23 seconds for naïve interpolation and 38 seconds for our interpolation. There are approximately $1.5k$ particles on an 80×80 regular background grid. This timing was measured using a MacBook Pro (16-inch, late 2019 model). Both programs ran single-threaded on a CPU. The longer computation time originates from neighboring grid point search and the matrix inverse in (6). Currently, we are searching for neighboring boundary grid points using a brute-force approach. We anticipate potential acceleration for our interpolation by incorporating spatial hashing techniques (e.g., kd trees). We can also accelerate the computation by switching to the analytic matrix inverse when there are no boundary grid points around a particle.

5. Conclusion

We presented the accurate boundary handling technique for MLS-MPM to prevent the particles from sinking into the solid wall. We interpolate the velocity field on an augmented grid, where the additional grid points are added to the boundary.

Limitation & future work Our method requires additional computational overhead for searching boundary grid points around each particle. This may cause difficulty in efficient GPU parallelization. 3D simulation might be possible by uniformly sampling the grid points on the boundary but is left as a future work. We consider coupled simulation with moving rigid bodies as one of the interesting future directions.

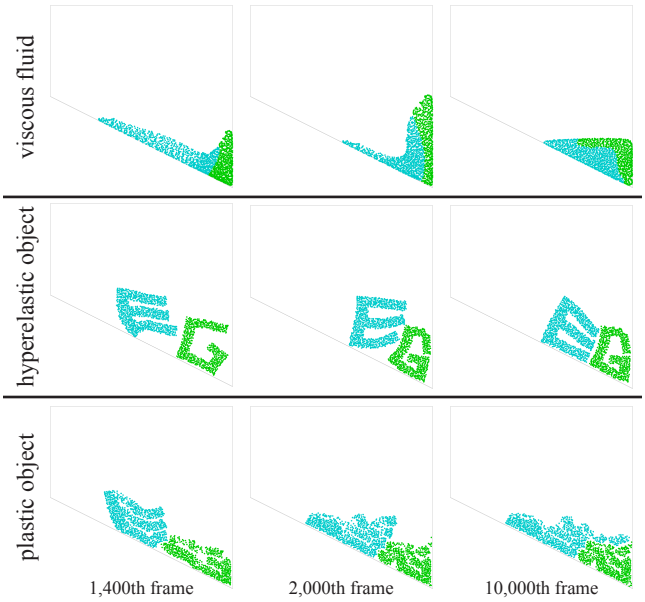

Figure 5: *Simulation of various materials using our augmented grid points with full slip boundary condition.*

References

[FHHJ18] FANG Y., HU Y., HU S.-M., JIANG C.: A temporally adaptive material point method with regional time stepping. *Computer Graphics Forum 37*, 8 (2018). 2

[GTJS17] GAO M., TAMPUBOLON A. P., JIANG C., SIFAKIS E.: An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Trans. Graph. 36*, 6 (2017). 2

[GWW*18] GAO M., WANG X., WU K., PRADHANA A., SIFAKIS E., YUKSEL C., JIANG C.: GPU optimization of material point methods. *ACM Trans. Graph. 37*, 6 (2018). 2

[HFG*18] HU Y., FANG Y., GE Z., QU Z., ZHU Y., PRADHANA A., JIANG C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics 37*, 4 (2018). 1, 2, 3

[HLA*19] HU Y., LI T.-M., ANDERSON L., RAGAN-KELLEY J., DURAND F.: Taichi: A language for high-performance computation on spatially sparse data structures. *ACM Trans. Graph. 38*, 6 (nov 2019). 3

[MPM] Notes and examples for the material point method. https://nialltl.neocities.org/articles/mpm_guide. 3

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3 (2006). 2, 3

[SSC*13] STOMAKHIN A., SCHROEDER C., CHAI L., TERAN J., SELLE A.: A material point method for snow simulation. *ACM Trans. Graph. 32*, 4 (2013). 1, 2, 4

[SZS95] SULSKY D., ZHOU S.-J., SCHREYER H. L.: Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications 87*, 1 (1995). Particle Simulation Methods. 1, 2

[Tai] Taichi MPM. https://github.com/yuanming-hu/taichi_mpm.git. 3

[TBBC*22] TAO M., BATTY C., BEN-CHEN M., FIUME E., LEVIN D. I. W.: VEMPIC: Particle-in-polyhedron fluid simulation for intricate solid boundaries. *ACM Trans. Graph. 41*, 4 (jul 2022). 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, Association for Computing Machinery, p. 965–972. 2