On Hybrid Lagrangian-Eulerian Simulation Methods: Practical Notes and High-Performance Aspects

YUANMING HU, XINXIN ZHANG, MING GAO, CHENFANFU JIANG

Hybrid Lagrangian-Eulerian simulation schemes have attracted a tremendous amount of attention in both academic and industrial communities of physics-based animation. Exemplified by the Fluid-Implicit-Particle (FLIP) method for incompressible fluids and the Material Point Method (MPM) for elastoplastic solids, increasingly more schemes of this category assist in asserting that a well-designed combinational usage of connectivity-free particles and a regular Cartesian background grid allows benefits from both representations. Despite the increasing amount of implementations of hybrid methods in industrial software, academic codebases, and open-source repositories hosted online, it appears that a significant portion of such implementations fails to pay enough attention to the most crucial aspect, performance, in such solvers. To alleviate this situation, the proposed course serves to disclose practical and impactful notes and techniques for achieving high-performance implementations of hybrid methods on a consumer-level workstation. With a focus on MPM and vortex methods, we cover aspects ranging from the data structure, algorithmic design, and low-level optimization for multi-threaded CPUs and massively parallel GPUs. We expect this course to sharply improve the overall code quality for a large number of practitioners from both academic and industrial groups, as well as contribute to improving the course content of physics-based animation in university programs.

1 INTRODUCTION AND COMPUTER ARCHITECTURE BACKGROUND (13 MIN)

Nowadays, it is possible for a $5K engineering workstation to provide a total 19 TFlops of compute density and 700GB/s aggregate memory bandwidth, while a single rackmount server valued under $25K can combine 112TFlops of peak compute performance, 5TB/s aggregate memory bandwidth while offering 768GB (DDR4) and 120GB (GDDR5X) aggregate memory capacity. It is noteworthy that the latter system would have contested for a spot on the TOP500 list just few years ago.

Although these advances in computational platform hardware have revealed opportunities for transformative improvements in scale and performance, the emergence of hybrid Lagrangian/Eulerian methods has made it possible to capture physical behaviors with a significantly easier engineering effort and a better defined path to performance, those opportunities are by no means guarantees of either performance, or feature completeness of a straightforward implementation. In order to have this potential materialize into simulation systems that truly delivers the features and scalability required of their driving applications, we present this course to cover essential aspects we found crucial in incorporating high-performance design into graphics simulation systems.

This 1.5-hour course is designed for researchers, engineers and students with prior experience implementing a hybrid Lagrangian-Eulerian solver and who know basic CPU, GPU and multilevel memory architectures. Some background on modern computer architecture and compiler will be covered in the very beginning.

Sample source code of covered techniques will be available online for assisting the teaching and serving as permanent references.

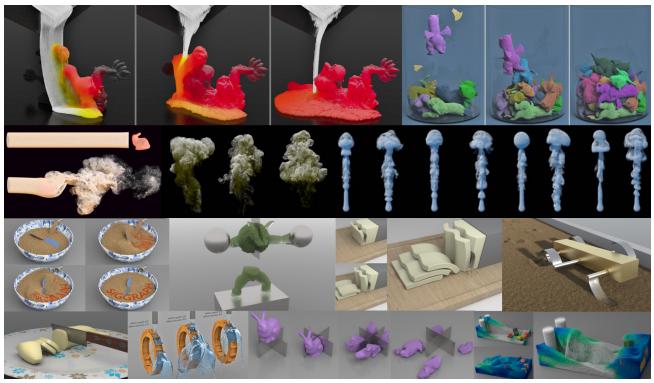

Fig. 1. Hybrid Lagrangian-Eulerian methods can simulate heat conduction, elastic dragon, bunny moving through smoke tunnel, raising ink drop, armadillo and bunny being cut, crawling robots, and so on.

2 HYBRID EULERIAN-LAGRANGIAN SIMULATIONS (20 MIN)

This section covers two types of Eulerian-Lagrangian simulations that recently became popular in computer graphics research, which are MPM methods and hybrid Eulerian-Lagrangian vortex methods.

While a majority of MPM implementations in computer graphics follow the algorithm by Stomakhin et al. [2013], two crucial algorithmic improvements are sometimes overlooked. The first technique, the Affine Particle-In-Cell (APIC) method [Jiang et al. 2015], replaces the FLIP-style particle-grid transfers with an affine PIC style. For MPM elastoplasticity simulations, APIC often results in an improved stability that permits a larger time step. We will also cover the more advanced Polynomial Particle-In-Cell method. The second improvement comes from the adoption of the Moving Least Squares kernel (MLS-MPM) [Hu et al. 2018]. MLS-MPM simplifies the implementation of force computation and halves the computational cost for explicit MPM schemes without affecting any accuracy or visual quality. This part of the course will focus on instructions of practical implementation of these two schemes.

3 DATA STRUCTURES (5 MIN)

In Lagrangian-Eulerian methods where both particles and grids play important roles, grid data storage needs special treatment. Clearly, only a narrow band around the particles needs grid nodes. Since particle distribution is irregular, sparse grids instead of dense ones are usually required to save memory. In this part, we will cover the following techniques:

Sparse Volumetric Data Structures. SPGrid [Setaluri et al. 2014], VDB [Museth 2013] and their variants on GPU [Gao et al. 2018; Wu et al. 2018] are typical choices. Sparsity comes at a cost of complexity,

so accessing a single node in SPGrid or VDB is computationally more expensive. To resolve this issue, the lowest level of the data structure hierarchy organization is usually blocks of nodes, because the expensive node access will only happen once for the "anchor" node in the block, and accessing other nodes in the same block is much cheaper (linear addressing). Typical values for block size are between $4 \times 4 \times 4$ and $8 \times 8 \times 8$.

Particle Sorting, Reordering, and Memory Layout. To make use of such locality, we need to process particles in a block-by-block manner. This means certain sorting is necessary to keep track of lists of particles within the range of each block.

To this end, each block should maintain a list of indices (i.e. pointers) of particles that reside in its range. A linear array is usually used to store the particle data. Sorting indices is much cheaper than sorting the particle data.

To maintain spatial locality it would be helpful to reorder the actual particle data in the linear array according to their belonging block. In practice, doing so every 50 time steps leads to a nice balance between acceleration and sorting overhead.

Local Scratchpad Grids. As mentioned before, brute-force operations on sparse data structures have high cost. Fortunately, after sorting, if we process particles in block-order, **local scratchpads grids** can be used. Assuming 3D MPM with quadratic kernel (i.e. each particle accesses a $3 \times 3 \times 3$ neighbourhood) and blocks are of size $4 \times 4 \times 4$, the scratchpad grid are dilated copies of local $6 \times 6 \times 6$ neighbouring nodes, i.e. dense arrays of $\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z$ (grid velocities) and m (mass).

Doing so can greatly improve performance, because (1) there are usually ~ 8 particles per cell, caching the nodes amortizes the global data structure access overhead, and (2) accessing nodes within the local scratchpad that fits CPU L1 data cache/GPU shared memory saves memory bandwidth and addressing instructions.

4 MULTITHREADING (5 MIN)

The grid-to-particle transfer is embarrassingly parallelizable since there is no data race. During particle-to-grid transfer, however, we need to accumulate mass and momentum in the local scratchpad grids to the global grid. Data races happen since local scratchpads of neighboring blocks overlap. Expensive per-cell locking can be used to ensure correct results when multiple threads are processing neighboring blocks.

We will cover techniques to make this process **lock-free** during particle-to-grid transfer. Specifically, we partition the blocks into 2^D sets where D is the simulation dimensionality, so that in each set any two blocks do not share overlapping global node.

5 VECTORIZATION AND OTHER LOW-LEVEL OPTIMIZATIONS ON CPU (5 MIN)

Modern CPUs are often equipped with Single Instruction, Multiple Data (SIMD) units, and GPUs provides the Single Instruction, Multiple Thread (SIMT) execution model . Such capability makes it possible to batch process vector arithmetics with 4 (SSE, released 1997), 8 (AVX2, released 2013), 16 (AVX-512) or 32 (GPUs) lanes. The fused-multiply-add units further enhance peak floating-point performance by executing $a \times b + c$ in a single instruction. With well-designed data structures, correctly using these efficient instructions is key to high performance. In this part, we will introduce vectorization options through SSE (4× float32), where each vector register holds a $(x, y, z, mass)$ tuple, coding tricks for the compiler to emit efficient instructions such as loop unrolling. Coding via x86_64 intrinsics (e.g. _mm_add_ps) or their wrappers ensures the compiler to emit vectorized instructions on CPUs.

6 GPU MPM ACCELERATION (22 MIN)

GPU, with its many-core SIMT architecture, has the potential to deliver better acceleration than relatively easier CPU parallelization in many other applications, such as cloth collision detection [Tang et al. 2018]. However, direct code implantation of a particle-grid hybrid pipeline from CPU to GPU results in severe performance deterioration in the the most critical particle-to-grid transfer (P2G) stage. In this course, we'd like to manifest how to design a simple yet efficient scheme to fully exploit the GPU intrinsic functions, e.g. **ballot**, to reach the GPU hardware saturation for the P2G kernel.

On the other hand, due to the large discrepancy of the parallelism potential and mechanism provided from the GPU and the CPU platforms, many other kernels need a rethinking and some of them require completely different redesigns. Therefore, we also cover some of the most important cases, such as fast GPU SVD implementation, particle sorting and particle reordering.

7 HIGH PERFORMANCE VORTEX METHODS (15 MIN)

For vortex modeling of turbulent flow, this course will illustrate a simpler-than-PPPM [Zhang and Bridson 2014] implementation and efficient approximated high-order temporal integration of large-scale vortex dynamics, with an emphasis on a weakly coupled Eulerian-Lagrangian formula of such turbulence model.

REFERENCES

Ming Gao, Wang, Kui Wu, Andre Pradhana-Tampubolon, Eftychios Sifakis, Yuksel Cem, and Chenfanfu Jiang. 2018. GPU Optimization of Material Point Methods. *ACM Transactions on Graphics (TOG)* 32, 4 (2018), 102.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 150.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 51.

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 27.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 205.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 102.

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (November 2018), 204:1–10.

Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. 2018. Fast Fluid Simulations with Sparse Volumes on the GPU. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 157–167.

Xinxin Zhang and Robert Bridson. 2014. A PPPM Fast Summation Method for Fluids and Beyond. *ACM Trans. Graph.* 33, 6, Article 206 (Nov. 2014), 11 pages. https://doi.org/10.1145/2661229.2661261