Accelerated 3D visualization of mock galaxy catalogues for the Dark Energy Survey

Tim Dykes,[1] Claudio Gheller,[2] Mel Krokos,[1] Marc Manera,[3] Marzia Rivi[3]

[1] *University of Portsmouth, Portsmouth, Hampshire, UK;*
tim.dykes@port.ac.uk,mel.krokos@port.ac.uk

[2] *CSCS-ETHZ, Lugano, Ticino, Switzerland;* cgheller@cscs.ch

[3] *University College London, London, UK;*
m.miret@ucl.ac.uk,m.rivi@ucl.ac.uk

Abstract.

We present a novel implementation of the visualization code Splotch[1], exploiting optimized atomic operations of modern GPU hardware found in todays heterogeneous supercomputers, and show its application to the visualization of mock galaxy catalogues for the Dark Energy Survey (DES)[2]. These mock catalogues are generated by numerical simulation codes producing results which can then be directly compared with observations or validated against other models. As use case we compare a catalogue based on the L-PICOLA[3] code with the MICE[4] catalogue for DES. We present and discuss the potentiality of our optimized visualization algorithm as a debugging tool for L-PICOLA, in particular focusing on gaining rapidly an insight into potential anomalies on large scales and aiding in planning of appropriate remediation measures.

1. Introduction

Nowadays numerical simulations (grid or particle based) constitute powerful instruments for describing, investigating and ultimately understanding, the dynamics and properties of a multitude of astrophysical objects. High Performance Computing (HPC) infrastructures are increasingly employed to execute such simulations resulting in progressively more accurate datasets thus contributing to dramatic growth in their sizes and complexity. Visualization can provide an effective way for data inspection, e.g. as an aid in rapidly focusing on relevant features of interest and detecting non-obvious correlations or even intrinsic data characteristics.

This paper focuses on the latest developments of an HPC-enabled volume rendering algorithm Splotch (Dolag et al. 2008). An optimized implementation of the GPU code is presented, with accompanying performance results, and applied to the

[1] https://github.com/splotchviz/splotch

[2] www.darkenergysurvey.org

[3] http://cullanhowlett.github.io/l-picola/

[4] http://maia.ice.cat/mice/

visualization of synthetic galaxy catalogues generated from MICE data (Carretero & et al. 2015). We visualize a catalogue built from data produced by the L-PICOLA code (Howlett et al. 2015), and compare with a catalogue built from MICE simulations. The exploitation of Splotch as a debugging tool for the underlying code pipeline is also discussed.

2. Visualization: Splotch

Splotch is a volumetric ray-casting algorithm for visualizing large-scale, particle-based numerical simulations. The code is optimized in terms of memory usage and exploitation of HPC architectures, e.g. multi-core, multi-node, multi-GPU heterogeneous supercomputing systems, through combination of the OpenMP, MPI (Jin & et al. 2010) and CUDA paradigms (Rivi & et al. 2014).

Firstly, Splotch *rasterizes* by applying geometric transformation and color assignment, and secondly it *renders* by solving the radiative transfer equation via ray-casting. Rendering particles in a highly parallel context introduces race conditions, where threads may try to simultaneously write to pixels affected by overlapping particles. To avoid this issue one must either arrange the source and destination data such that conflicting writes do not occur, for example with a tiling schema (Rivi & et al. 2014), or use atomic operations to ensure that each read-write combination occurs in uninterruptible isolation, and as such race conditions are not possible.

With the introduction of the NIVIDA Kepler and later architectures, the performance of atomic operations is greatly increased. The optimized implementation has been remodelled to take advantage of this development, replacing the tiling schema with a simpler strategy making use of the CUDA *atomic add* function to safely accumulate overlapping particles to a single pixel in a parallel context. The result is a reduction in both code complexity and computational overhead, increasing overall performance (see Sec. 4).

3. Scientific Case: Mock galaxy catalogues for DES

The optimized Splotch algorithm has been used for visualization of mock galaxy catalogues for DES, an on-going five year world effort to survey the southern sky. DES will provide a light-cone galaxy catalogue for a part of sky wide 5000 deg^2 and deep $z < 1.4$ for five flux bands.

The catalogues we are visualizing here are generated by populating dark matter halos produced by numerical simulations on volumes larger than or equal to that of DES. The galaxy catalogues are constructed to match observed luminosity functions, color distributions and clustering as a function of luminosity and color at low redshifts. Galaxy properties are then evolved into the past-lightcone using evolutionary models.

The full procedure is highly computationally demanding, and there are multiple codes that can be used to generate such simulations. Among them, L-PICOLA has the advantage of generating and evolving a set of initial conditions into a dark matter field much faster than a full non-linear N-Body simulation. Additionally, it has the ability to include primordial non-Gaussianity in the simulation and simulate the past light-cone at runtime, with optional replication of the simulation volume.

From the L-PICOLA dark matter fields one can generate galaxy catalogues which must be validated against other models and visualization provides a first prompt and effective way to compare different datasets addressing the solution of possible mismatches or even errors. To this end, we compared a catalogue produced from L-PICOLA data with a galaxy catalogue built from MICE simulations. MICE dark matter halos were generated running an N-body simulation with 70 billion particles in a $(3h^{-1}$ Gpc$)^3$ comoving volume equal to that of DES.

Through visualizing the bands difference G-R of a catalogue slice for MICE and L-PICOLA (Fig. 1), we can see a clear difference in the color of the plotted quantity, showing lower values for L-PICOLA. This is indicative of less massive halos generated by this code as galaxy colors are dependent on the mass of the halo hosting the galaxy.

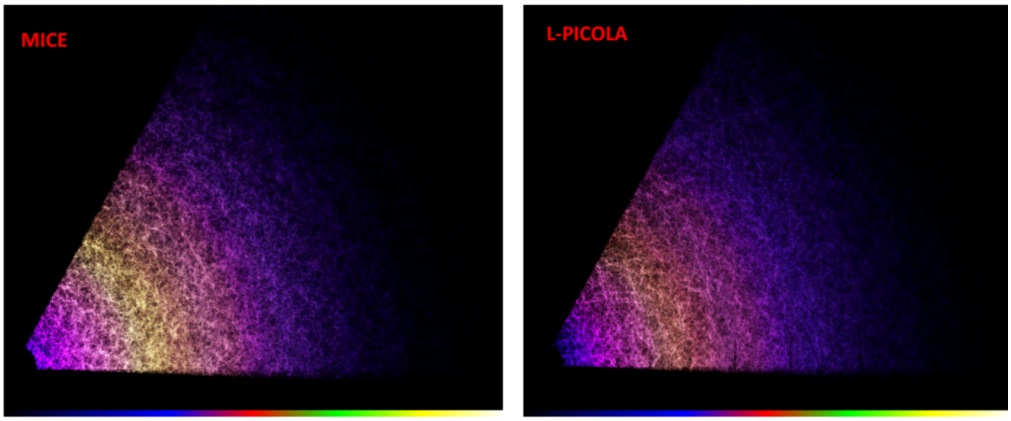

Figure 1. A 600 deg^2 slice of the MICE data catalogue (10.8M galaxies, 0.4GB) and the L-PICOLA data catalogue (11.8M galaxies, 0.44GB).

4. Performance Results

For testing we utilized an NVIDIA Tesla K20X, based upon the Kepler architecture, and an 8-core Intel Sandybridge Xeon E5-2670 CPU.

The atomic and tiled implementations are first compared by rendering a series of frames (Fig. 2) zooming in to the MICE catalogue data (~500M galaxies, ~18GB, 800^2 pixels). In this comparison most particles affect a small number of pixels (no more than 4), and the atomic update performs very well against the tiled schema, due to avoiding the overhead for sorting and tiling particles. Speed-up ranges from 1.3x where many particles are clipped, decreasing overhead in the tiled rendering phase, to 3.1x where all particles are visible (Fig 3).

A further comparison (Fig. 4) gauges speed-up in both total compute time and individual kernels against 8 cores of a Sandybridge CPU, parallelized with OpenMP, using a N-body simulation dataset (~220M particles, ~8GB, 1024^2 pixels) that, in close range, has many overlapping particles with large radii (large r_0); this can be particularly difficult for parallel rendering as described in Sect 2. The rasterization is highly parallel at between 5x and 6x speedup, this is unchanged from the previous implementation. For smaller radii (lower r_0), the atomic rendering code provides speed-up over the CPU in four out of seven cases, in comparison to a single case for the tiled code.

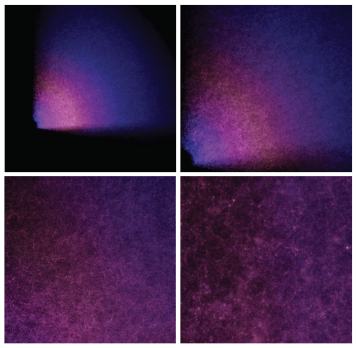

Figure 2. Zoom-in to galaxy catalogue, plotting G - R bands.

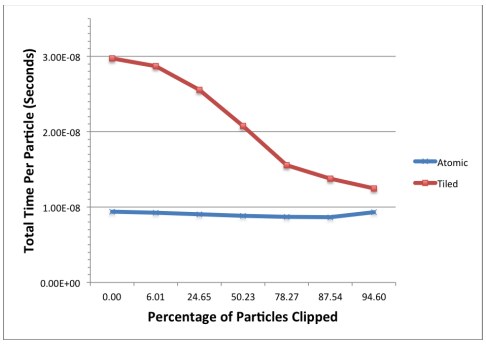

Figure 3. Total time / Total no. particles plotted against percentage of data clipped.

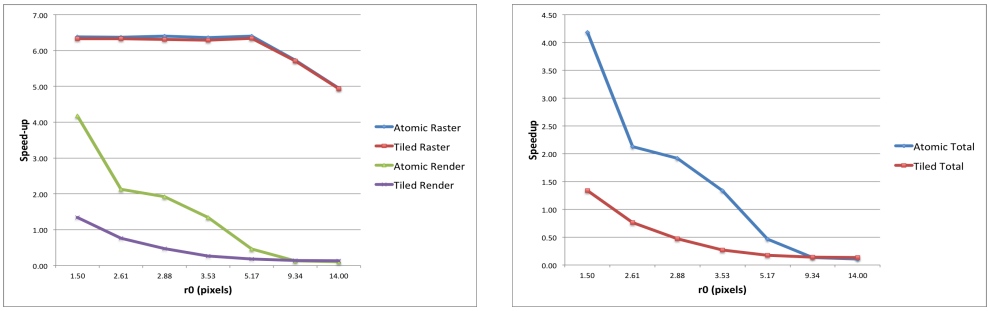

Figure 4. Speedup against 8-core CPU (*left*: single kernels, *right*: total computation).

5. Conclusions

We presented an optimised CUDA implementation (using atomic operations) of our 3D visualization code Splotch, comparing with a previous tiled rendering scheme. The rendering performance depends on the number of large particles for both CUDA versions due to balancing the work-load while rendering large, overlapping, particles in parallel. Use of atomic operations provides a significant performance improvement for modern GPU accelerators, but further optimization is still required for memory locality and load balancing. We applied Splotch as a visual exploration and debugging tool for simulation codes, comparing large mock galaxy catalogues produced from the L-PICOLA runs with MICE for DES and identifying an issue in the L-PICOLA pipeline.

Acknowledgments. Particular thanks to P. Fosalba, F. Castander, E. Gaztañaga and M. Crocce for providing the MICE catalogue used to test Splotch performance, and validate the L-PICOLA code.

References

Carretero, J., & et al. 2015, MNRAS, 447, 646
Dolag, K., Reinecke, M., Gheller, C., & Imboden, S. 2008, New Journal of Physics, 10
Howlett, C., Manera, M., & Percival, W. J. 2015, Astronomy and Computing, 12, 109
Jin, Z., & et al. 2010, Procedia Computer Science, 1, 1775
Rivi, M., & et al. 2014, Astronomy and Computing, 5, 9