LITERATURE REVIEW: Eikonal Solver for GPU

Olivier Hamel
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
olivierhamel@cmail.carleton.ca

October 24, 2016

1 Introduction

The Eikonal equation is used to model the propagation of wave-fronts through a (typically isotropic) field. It has numerous applications (e.g. graphics, computer vision, pathfinding, etc.) but the canonical algorithm for solving it, the Fast Marching Method, is difficult to parallelise. This has prompted a number of attempts to create methods more amendable to parallelisations. These methods can be classified into 3 main approaches: The Fast Marching Method, the Fast Sweeping Method[8], and the Fast Iterative Method[6].

Both FSM and FIM are much easier to parallelise than FMM, but both also perform poorly when confronted with non-straight characteristic lines. (FSM is particularly awful in this regard.) Fixing this problem requires that the causal wave-front be tracked in some fashion to minimise the re-computation of nodes in the grid.

We intend to implement and evaluate an Eikonal solver for GPUs which improves on FIM's performance by avoiding re-computations due to non-straight characteristic lines. The evaluation will consist of a performance comparison between the implemented algorithm and vanilla FIM on the GPU executing a number of benchmarks (synthetic and non-synthetic). The algorithm in question will be a GPU implementation of the Semi-Ordered Fast Iterative Method (SOFI)[3] in two dimensions.

2 Literature Review

There are three main approaches for solving Eikonal equations: FMM, FSM, FIM.

FMM is essentially a Dijkstra with a more complicated cost function. Since its node expansion follows the causal wave-front it minimises the number of cost function evaluations, but is difficult to parallelise due to the sequential ordering required by its open queue. It has a number of variants, most of which are concerned with reducing the cost of managing its priority queue. These include Simplified FMM (nodes are not deleted from the queue when reinserted), Untidy FMM[10] (quantisation of the queue to provide a precision/performance trade-off).

FSM operates by performing a number of Gauss-Seidel iterations until the solution converges. For straight characteristic curves was shown by Qian et al. 2007 to require only 2^d for d dimensions. Non-straight characteristic curves require repeated sweeps until the error becomes acceptable. It performs terribly whenever the characteristic curves' tangents

change quadrant direction frequently due to the number of iterations this entails. Some improvements include Locked Sweep Method (tracks which sweep planes can no longer yield an improvement, 'locking' them) and the Two Queue method (resolve near-future cells before far-future cells).[1] In general the FSM is inferior to FIM. (See the performance evaluations by Gomez et al. 2015.[4])

FIM is half-way between FSM and FMM. It follows a node expansion wave-front, but unlike FMM it does not use a sorted queue but an ordered list. This entails some iterations for the solution to converge as eagerly evaluated nodes are recalculated, but typically not as much as FSM would. There are a few variants such as Block FIM[6] which attempts to maintain memory access locality to improve performance, and the Semi-ordered Fast Iteration Method (SOFI)[**?**] which takes the Two-Queue technique and applies it FIM.

Aside from these there are a number of papers focused on domain decomposition of the grid-space for parallel execution of these subdomains,[2][5][9], and use of non-grid spaces[7]. Finally, while it does not bare directly on our project, it is interesting to note Zhou & Zeng 2015 where they parallelise A* (and therefore Dijkstra and other queue-based label correcting algorithms) for execution on the GPU.[11]

References

[1] Stanley Bak, Joyce McLaughlin, and Daniel Renzi. Some improvements for the fast sweeping method. *SIAM Journal on Scientific Computing*, 32(5):2853–2874, 2010.

[2] Miles Detrixhe and Frédéric Gibou. Hybrid massively parallel fast sweeping method for static hamilton–jacobi equations. *Journal of Computational Physics*, 322:199–223, 2016.

[3] Tor Gillberg. A semi-ordered fast iterative method (sofi) for monotone front propagation in simulations of geological folding. In *MODSIM2011, 19th International Congress on Modelling and Simulation*, pages 641–647, 2011.

[4] Javier V Gomez, David Alvarez, Santiago Garrido, and Luis Moreno. Fast methods for eikonal equations: an experimental survey. *arXiv preprint arXiv:1506.03771*, 2015.

[5] Sumin Hong and Won-Ki Jeong. A multi-gpu fast iterative method for eikonal equations using on-the-fly adaptive domain decomposition. *Procedia Computer Science*, 80:190–200, 2016.

[6] Won-Ki Jeong and Ross T Whitaker. A fast iterative method for eikonal equations. *SIAM Journal on Scientific Computing*, 30(5):2512–2534, 2008.

[7] Mohammad Mirzadeh, Arthur Guittet, Carsten Burstedde, and Frederic Gibou. Parallel level-set methods on adaptive tree-based grids. *Journal of Computational Physics*, 322:345–364, 2016.

[8] Jianliang Qian, Yong-Tao Zhang, and Hong-Kai Zhao. A fast sweeping method for static convex hamilton–jacobi equations. *Journal of Scientific Computing*, 31(1):237–271, 2007.

[9] Josef Weinbub and Andreas Hössinger. Shared-memory parallelization of the fast marching method using an overlapping domain-decomposition approach. In *Proceedings*

of the 24th High Performance Computing Symposium, page 18. Society for Computer Simulation International, 2016.

[10] Liron Yatziv, Alberto Bartesaghi, and Guillermo Sapiro. O (n) implementation of the fast marching algorithm. *Journal of computational physics*, 212(2):393–399, 2006.

[11] Yichao Zhou and Jianyang Zeng. Massively parallel a* search on a gpu. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1248–1254. AAAI Press, 2015.