MCD: Mesh Collision Detection with Parallel Acceleration

Carnegie Mellon University, 15-418/618 Final Project, Spring 2023
Yufei Shi (yshi2) and Bo Ying Su (boyings)

Introduction

• Collision detection and minimum distance query are fundamental problem in various fields, including computer graphics, robotics, and physical simulations.

• As the objects become more complex (more number of vertices) and the number of objects increases, the collision detection problem becomes more challenging as traditional sequential algorithms struggle to provide real-time performance.

• To address this problem, we implemented a parallel algorithm for computing the minimum distance and point pair between convex meshes using CUDA. We also parallized the lookup function for Axis-Aligned Bounding Box (AABB) Hierarchy using OpenMP to filter mesh pairs that are far away from each other before passing the meshes to our algorithm.

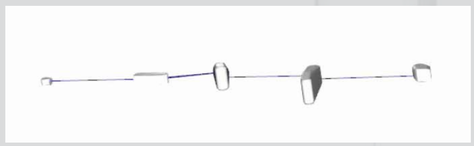

Figure 1. A simple scene can be more complex than it seems. Each mesh in this scene has more than 2000 faces.

References

[1] William Bittle. Gjk, distance, closest points, Apr 2010.
[2] Stephen Cameron. Computing the distance between objects.
[3] Christer Ericson. Real-Time Collision Detection. CRC Press, Inc., USA, 2004.
[4] E. Larsen. Pqp - a proximity query package, June 1999.
[5] Harold Serrano. Visualizing the gjk collision detection algorithm, Jul 2016.
[6] Wikipedia. Minimum bounding box, Jun 2022.
[7] Wikipedia. Bounding volume hierarchy, Apr 2023.
[8] Wikipedia. Gilbert-johnson-keerthi distance algorithm, Mar 2023.

Approach

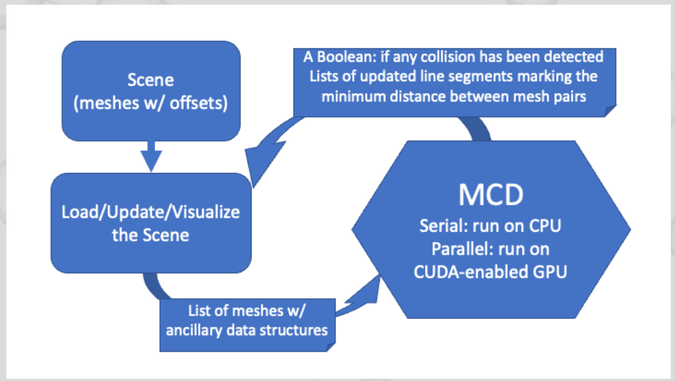

Figure 2. Mapping of the problem to a software implementation.

We created a framework that handles scene set-up, visualization, and invokes MCD for the scene and performs updates repeatedly.

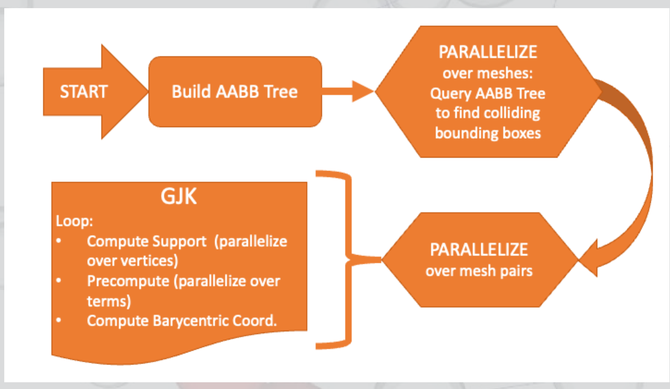

Figure 3. Flow chart of the parallelized MCD implementation.

At the core of MCD are an AABB tree (which provides accelerated filtering) and the GJK algorithm. They are parallelized using OpenMP and CUDA, respectively.

Results

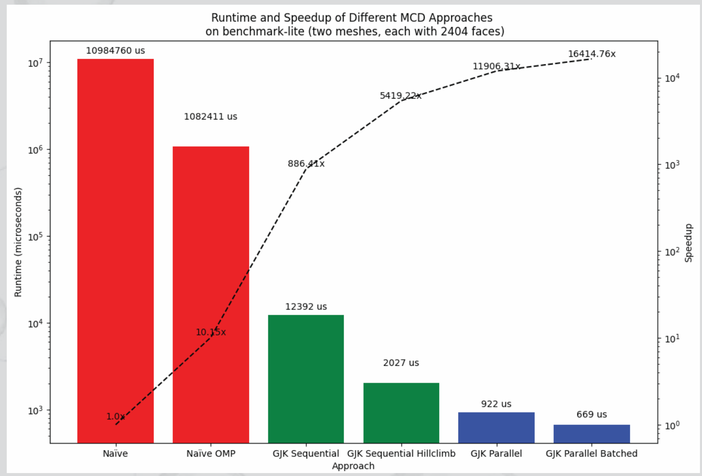

Figure 4. Runtime and speedup of different MCD implementations.

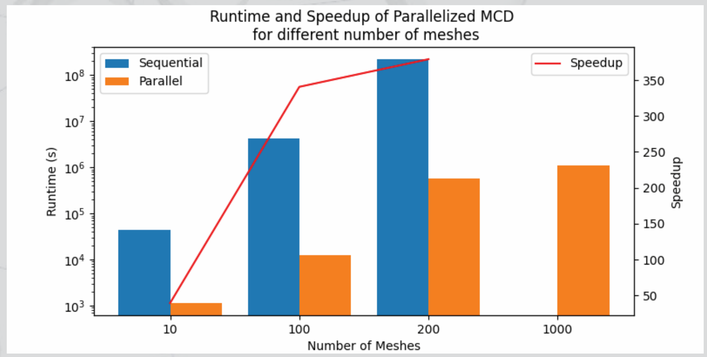

Figure 5. Runtime and speedup of parallel MCD on different number of meshes.

In minimum distance query, our parallelized GJK algorithm running on CUDA-enabled GPU significantly outperforms the state-of-the-art sequential baseline (Figure 4. GJK Sequential Hillclimb)
Figure 5. shows the speed up of our parallelized GJK algorithm over the sequential implementation when varying the number of meshes in the scene. The performance data of sequential implemenataion with 1000 objects is missing since it was taking too long.