A Synchronization Mechanism for Parallel Geometric Algorithms

Joel Fuentes[1], Fei Luo[2] and Isaac D. Scherson[3]

[1]Department of Computer Science and Information Technologies. Universidad del Bio-Bio, Chillán, Chile

[2]School of Information and Engineering. East China University of Science and Technology. Shanghai, China

[3]Donald Bren School of Information and Computer Sciences. University of California, Irvine, USA.

jfuentes@ubiobio.cl

IHPCSS 2017

Abstract

A new synchronization mechanism called Spatial Lock for parallel geometric algorithms is presented. We demonstrate that Spatial Locks can ensure thread synchronization in geometric applications that perform parallel operations over objects in 2D or 3D space. A parallel algorithm for mesh simplification was implemented using Spatial Locks to show its usefulness when parallelizing geometric application with ease. Experimental results illustrate the advantage of using this synchronization mechanism.

Introduction

Geometric algorithms are known for being highly compute-intensive. With the advent of computing systems that use silicon devices with many CPUs per chip, parallelizing these algorithms has become a desired objective in order to achieve significant computational improvement. For example, in triangulated surface meshes a typical technique for parallelizing geometric algorithms is decomposing the mesh into submeshes and run the algorithm sequentially on each part. However, there are several algorithms that require special order for processing objects, and decomposing the mesh into submeshes can lead to bad quality results.

We introduce a new synchronization mechanism called Spatial Locks that allows thread synchronization when updating objects concurrently in 2D or 3D space.

Spatial Lock

We use the concepts of Spatial Hashing and Axis-aligned Bounding Box (AABB) to build this synchronization mechanism. Spatial hashing is a data structure that subdivides the 2D or 3D space into uniformly-sized cells, and then stores data elements in these cells, as shown in Figure 1. It is called spatial hash because the cell index of a data element can be obtained in constant time by its coordinates (x, y and z) as a hash function. AABB is also a data structure that represent an axis-align building box for certain object.

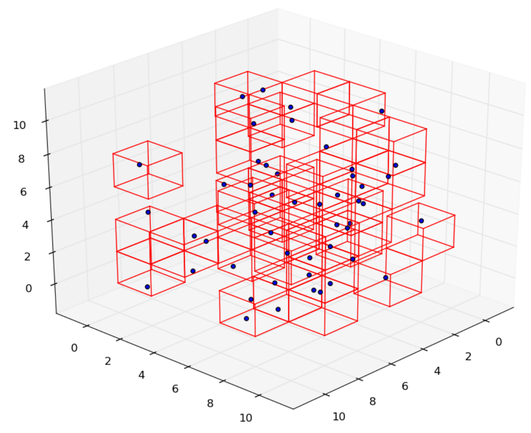

Figure 1: Spatial hashing where objects are mapped into uniformly-sized cells

The Spatial Locks mechanism works by protecting objects being updated by a thread from other thread's updates over the same object. For this purpose, it maintains an internal spatial hash table that reflects the status of concurrent updates by threads, it is in which cells are threads performing concurrent updates. The lock object operation is shown in Algorithm 1 where it can be seen that the spatial hash table is updated by using the atomic operation compare and swap (CAS) to guarantee thread safety.

Algorithm 1: Lock object (AABB) in Spatial Hash Table

```
Input : an axis-aligned bounding box (AABB)
minIndex = getCell(aabb.minPoint);
maxIndex = getCell(aabb.maxPoint);
if minIndex == maxIndex then
    while true do
        if table[minIndex] == -1 then
            if table[minIndex].CAS(-1, 1) then
                break;
            end
        end
    end
else
    //continue with other cases (minIndex != maxIndex)
end
```

Mesh Simplification

Surface mesh simplification is the process of reducing the number of faces used in a surface mesh while keeping the overall shape, volume and boundaries preserved as much as possible [1].

Figure 2: Surface mesh simplification algorithm applied to a mesh with different simplification levels.

The algorithm presented in [2] can simplify any oriented 2-manifold surface using a method known as edge collapse with quadratic errors. The method consists of iteratively replacing an edge with a single vertex, removing 2 triangles per edge collapse operation as shown in Figure 3.

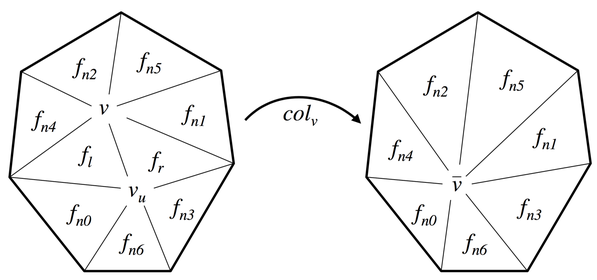

Figure 3: Contraction of the edge (v, v_u) into a single vertex. The shaded triangles become degenerate and are removed during the contraction.

Parallel Mesh Simplification using Spatial Locks

The introduced modifications to make the explained algorithm parallel are:

- A shared vector T is used to store the triangles. If priorities are considered, a concurrent priority queue can be used instead.
- Every thread takes a triangle t from T and starts analyzing the quadratic errors from its edges.
- If an edge is set to be collapsed, we lock the triangle by creating a corresponding AABB and inserting it into the spatial hash table.
- Once the edge is collapse and the shaded triangles removed, the corresponding AABB is remove from the spatial hash table.

The process of concurrent edge-collapse operation is shown in Figure 4 and Algorithm 2. If two threads are attempting to update adjacent triangles, only one will succeed locking the triangle and the other must wait.

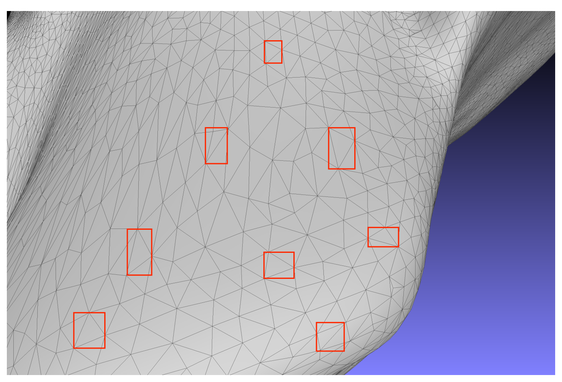

Figure 4: Edge-collapse operations being executed in parallel

Algorithm 2: Parallel Mesh Simplification using Spatial Lock

```
Input : Set of triangles
Output: Reduced set of triangles
parallel for every triangle t do
    for every edge e do
        err = calculateError(e);
        if err > threshold then
            continue;
        else
            aabb = createAABB();
            spatialHashTable.put(aabb);
            collapseEdge(e);
            spatialHashTable.remove(aabb);
        end
    end
end
```

Experimental Results

A set of experiments were carried out in order to evaluate the performance of the parallel mesh simplification algorithm and to compare it to its sequential version. Meshes with more than 1 million of triangles are particularly of our interest, since their simplifications require a big amount of edge-collapse operations.

Model	# triangles	# removed triangles	Original Alg. (sec.)	Parallel Alg. (sec.)
Bunny	69,664	34,832	0.3	0.2
Head	281,724	140,862	1.4	0.9
Wall	651,923	325,961	3.3	2.0
Einstein	674,038	337,018	4.1	2.9
Castle	2,436,234	121,8116	30.3	12.8

Table 1: Performance of mesh simplification algorithms with different models. Models obtained from https://ten-thousand-models.appspot.com

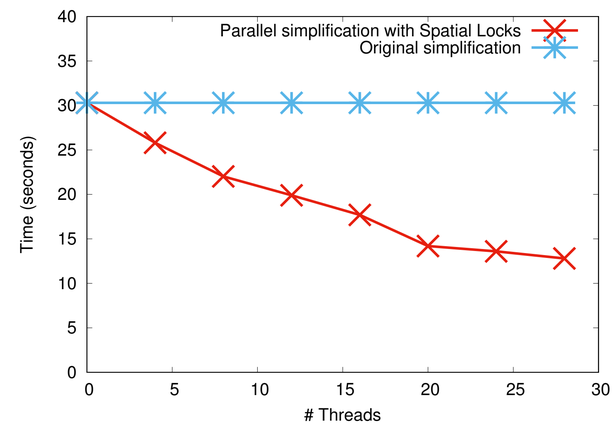

Figure 5: Total time spent by both algorithms in simplifying a surface mesh with 2,436,234 triangles.

Conclusions

Spatial Locks are a useful synchronization mechanism that allows to make parallel geometric algorithms thread-safe. Based on Spatial Hashing and AABB, they provide constant-time lock/unlock operations when updating an object. Experiments show that highly parallel executions can be obtained when using this mechanism for mesh simplification processes and big meshes.

Forthcoming Research

Compare our Parallel Mesh Simplification algorithm with other parallel proposals in terms of results quality and performance. Find more geometric algorithms that can benefit from using Spatial Locks.

References

[1] Fernando Cacciola. Triangulated surface mesh simplification. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.10 edition, 2017.

[2] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *Proceedings of the Conference on Visualization '98*, VIS '98, pages 279–286, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.