The challenge

- Roundoff errors: challenge in geometric computation.
- They can be avoided with exact rational numbers.

Big datasets:
- Greater chance of having errors.
- Computation with rationals: slower than native floats.
- People want exactness and performance.

Interval Arithmetic (IA)
- Interval arithmetic (IA) + arithmetic filtering can accelerate exact computation.
- Each coordinate/value: represented with exact part (rationals) and an interval approximation (floats).
- Computation is done with the approximation.
  - E.g.: [3.5] - [1.2] = [1.2-3.5] = [1.0,4.0]
  - The approximation [0.9,4.1] is not ok (contains [1,4])
  - The approximation [1.1,4.1] is not ok (does not contain [1,4])
- Interval arithmetic + IEEE-754 (rounding modes): computation can be done ensuring the interval will always CONTAIN be exact result (containment property).
- Containment property $\rightarrow$ sign of the exact result can be inferred from the intervals:
  - $a \in [0,1] \land b \in [0,1]$ positive $\iff a + b \in [0,2]$
  - $a \in [0,1] \land b \in [0,1]$ positive $\iff a + b \in [0,2]$
- Geometric predicates: typically computed with sign of a determinant (suitable for IA).

IA on GPUs
- IA: much faster than rationals, but slower than regular floating-point.
- GPUs: excellent for floating-point and intervals.
- Rounding mode can be switched on (on a GPU $\rightarrow$ would empty the pipeline).
- Example of the operator + using CUDA:
  $[a_2 + b_2] + [c_2 + d_2] = [a_2 + b_2 + c_2 + d_2]$

Steps of the algorithm

1. Uniform grid indexing
- 3D grid is created with a rogged array.
- Red and blue triangles inserted into the cells they intersect.
- For each cell c: bounding-box intersection tests are performed with the pairs of red-blue triangles in c.
- If there are pairs of potentially intersecting triangles, the GPU thread processes them two passes:
  - First: count the intersections.
  - Second: insert the intersecting pairs into an array.
- Each GPU thread processes some pairs.
- Challenge: determine the pair each GPU thread will process (irregular distribution of triangles among grid cells).
- Result: array of pairs of potentially intersecting triangles.

2. Triangle-triangle intersection
- For each pair of potentially intersecting triangles, intersection tests are performed.
- Uses orientation predicates implemented with IA.
- Orientation $\iff$ sign of determinant: IA returns positive, negative, 0 or unknown (failure).
- Each GPU thread processes a pair of potentially intersecting triangles.
- Result is two arrays:
  - Intersections: certainly intersecting pairs of triangles.
  - Failures: Interval failures (rarely happens) $\iff$ when orientation cannot be inferred using the intervals.

3. Post-processing
- The (typically few) failures (uncertainties) are re-evaluated on the GPU with GMRT rationals.
- Duplicated pairs of intersecting triangles are removed (using a GPU sort+unique implementation).

Results and conclusions

- Intel Xeon E5-2680 CPU at 2 GHz (3.2 GHz Turbo Boost), 256 GB of RAM, RTX 8000 GPU (48GB of RAM + 4688 CUDA cores).
- Datasets provided by IWR2024 and tetrahedralized with Gmsh:
  - Blue Crab: 25x10^{10} triangles $\rightarrow$ 45x10^{10} triangles in the ragged array.
  - Edgar Allan Poet: 13x10^{10} triangles $\rightarrow$ 64x10^{10} triangles in the ragged array.
  - Uniform grid: 100^3 cells, 87% are empty.
  - Baseline: sequential CPU implementation.
  - Steps:
    - Pre-processing: access index, perform bounding-box tests and distribute work among threads.
    - Post-processing: remove duplicates and re-evaluate interval failures with rationals.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>CPU</th>
<th>GPU Double</th>
<th>GPU Float</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-processing</td>
<td>Time (s)</td>
<td>64.86</td>
<td>1.00</td>
<td>1.09</td>
</tr>
<tr>
<td>Intersection</td>
<td>325.52</td>
<td>11.80</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>Post-processing</td>
<td>8.08</td>
<td>0.11</td>
<td>0.63</td>
<td></td>
</tr>
<tr>
<td>Data transfer</td>
<td>1.75</td>
<td>1.75</td>
<td>1.97</td>
<td></td>
</tr>
<tr>
<td>Total time</td>
<td>398.46</td>
<td>14.75</td>
<td>4.02</td>
<td></td>
</tr>
</tbody>
</table>

- Speedup: 993x on the intersection tests, 99x on the total time.
- Double precision: fewer (0) failures, but slower computation.
- Approximate floats on GPUs (where they shine) can accelerate exact geometric computation.

Future work:
- Employ this technique for other applications.
- Higher speedups could be achieved in applications where bigger bottlenecks could be moved into the GPU (performing more computation and fewer memory transfers).

Bibliography


Acknowledgement

W. Randolph Franklin: Rensselaer Polytechnic Institute, USA
Salles V. G. Magalhães: Universidade Federal de Viçosa, Brazil