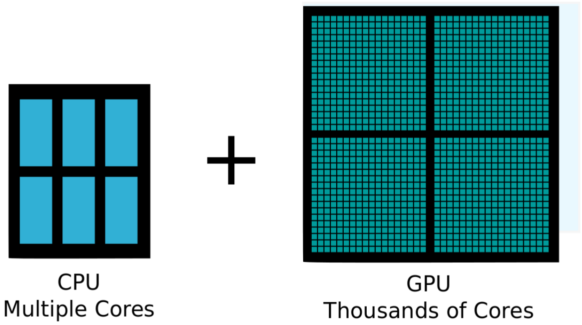
Computational geometry
Introduction to GPU programming:
GPU accelerated Convex Hull computation

FAGUREL Petro

December 2021

Why



[1]

- Laverage the parallel power

How

OpenCL MergeHull:

- https://github.com/pfagurel/OpenCL-ConvexHull/blob/main/GPUHull.pdf

CUDA QuickHull:

- https://timiskhakov.github.io/posts/computing-the-convex-hull-on-gpu

OpenCL QuickHull code:

- https://github.com/pfagurel/OpenCL-ConvexHull/tree/VsProject

Interface code:

- https://github.com/pfagurel/OpenCL-ConvexHull/tree/Interface

What

- MergeHull
- BottomUp MergeHull
- Parallel MergeHull
- Results

MergeHull

Algorithm

Algorithm CH

Input. A set $S = \{a_1, \ldots, a_n\}$, where $a_j \in E^d$ and $x_1(a_i) < x_1(a_j) \Leftrightarrow i < j$ for $i, j = 1, \ldots, n$.
Output. The convex hull $CH(S)$ of S.

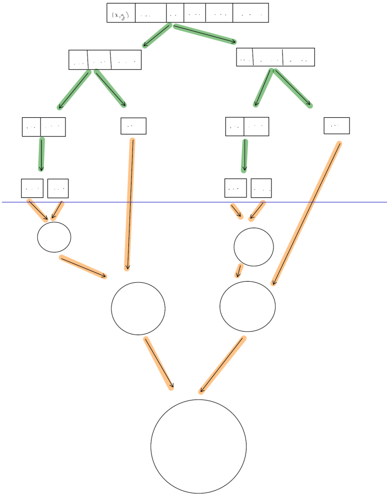
Step 1. Subdivide S into $S_1 = \{a_1, \ldots, a_{\lfloor \frac{1}{2}n \rfloor}\}$ and $S_2 = \{a_{\lfloor \frac{1}{2}n \rfloor + 1}, \ldots, a_n\}$.
Step 2. Apply recursively Algorithm CH to S_1 and S_2 to obtain $CH(S_1)$ and $CH(S_2)$.
Step 3. Apply a *merge* algorithm to $CH(S_1)$ and $CH(S_2)$ to obtain $CH(S)$ and halt.

[2]

- First presented in Preparata's and Hong's paper *Convex Hull of a Finite Set of Points in Two and Three Dimensions*
- Divide and Conquer technique

MergeHull

General View



MergeHull

Merge

```
bool done = 0;
while (!done)
{
    done = 1;
    if(m>1)
    while (side(W[ix2], V[ix1], V[(ix1 + 1) % m])>= 0)
        ix1 = (ix1 + 1) % m;
    if(n>1)
    while (side(V[ix1], W[ix2], W[(n + ix2 - 1) % n])<= 0)
    {
        ix2 = (n + ix2 - 1) % n;
        done = 0;
    }
}
```

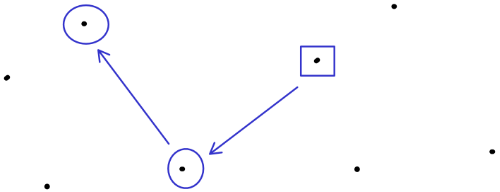
- Finding of upper tangent points
- Similar for lower

MergeHull

Merge

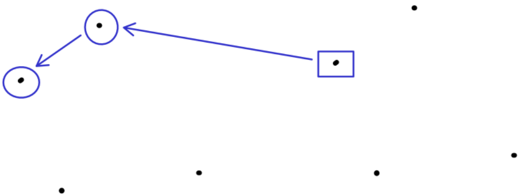
MergeHull

Merge



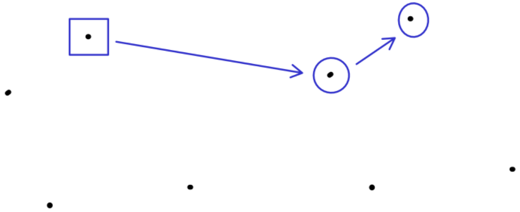
MergeHull

Merge



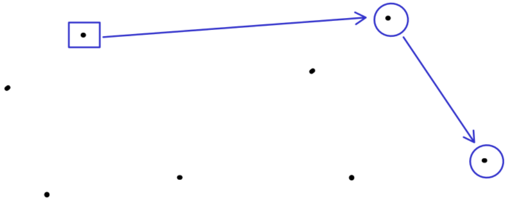
MergeHull

Merge



MergeHull

Merge



Parallel MergeHull

Problems

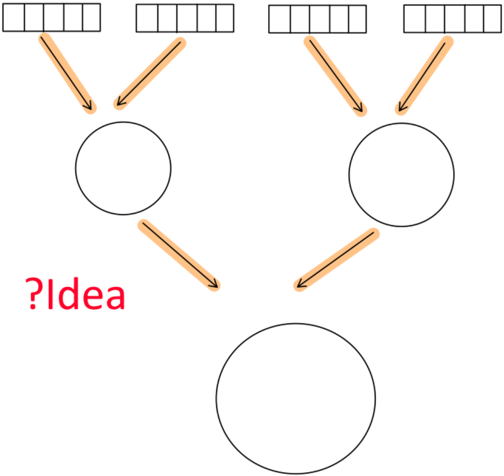
We want to decrease the number of merge operations.

Original MergeHull problems:

- The smallest subset size is not fixed.
  It can only be bound.
- Two arbitrary sets can not be merged.
  They have to be convex hulls.

BottomUp MergeHull

Idea



BottomUp MergeHull

Algorithm

```
for (int sz = d_size; sz < size; sz = sz + sz)
        for (int lo = 0; lo < size - sz; lo += sz + sz)
                        ...
```

- No merge yet.

BottomUp MergeHull

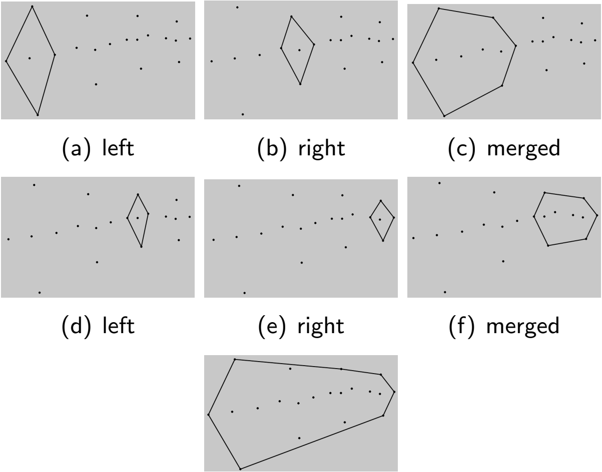
Algorithm

```
for (int sz = d_size; sz < size; sz = sz + sz)
        for (int lo = 0; lo < size - sz; lo += sz + sz)
                if (subset_size == dsize)
                        jm(subset);
                merge(...);
```

- OK.
  First compute convex hulls.
  (e.g Jarvis March)

BottomUp MergeHull

Example



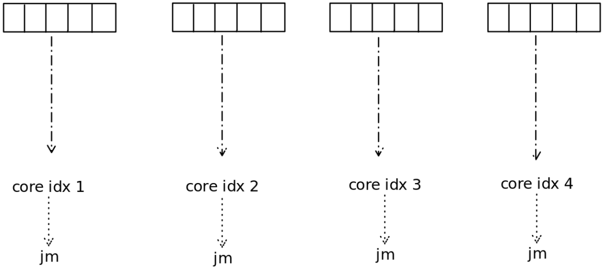
(a) left (b) right (c) merged

(d) left (e) right (f) merged

(g) convex hull

Parallel MergeHull

GPUA



Parallel MergeHull

GPUA

```
jm_gpua();
for (int sz = d_size; sz < size; sz = sz + sz)
        for (int lo = 0; lo < size - sz; lo += sz + sz)
                merge(...);
```

Results

Preliminary

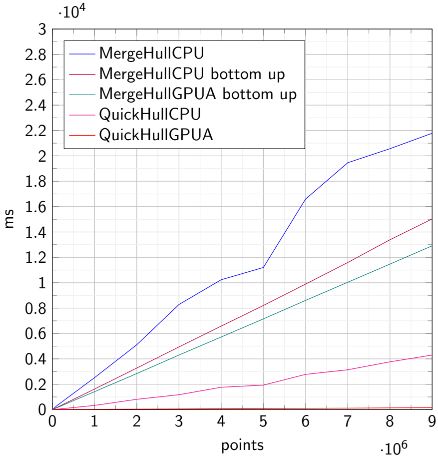


Figure: Execution on I7 4790k GTX980

Results

Parameter change

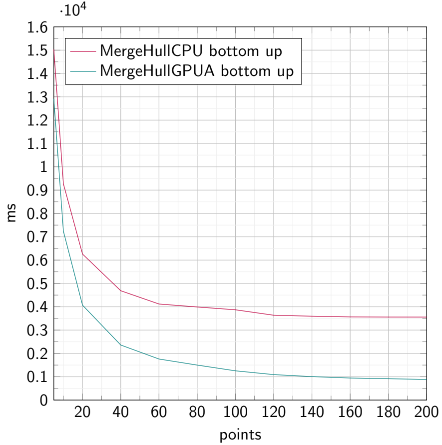


Figure: dsize change over 9 million points

Results

MergeHull vs QuickHull

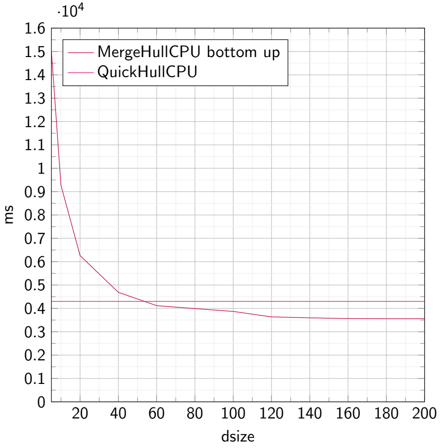


Figure: dsize threshold

Results

MergeHull vs QuickHull

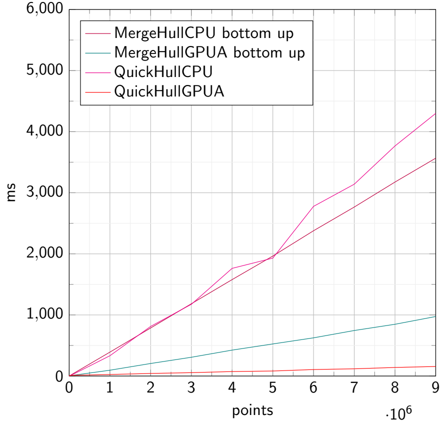


Figure: Execution on I7 4790k GTX980 with dsize=160

[1] MYO NeuralNet. *Learning MNIST with GPU Acceleration - A Step by Step PyTorch Tutorial*. 2017. URL: http://makeyourownneuralnetwork.blogspot.com/2017/05/learning-mnist-with-gpu-acceleration.html.

[2] F.P. Preparata and Se Hong. "Convex Hull of a Finite Set of Points in Two and Three Dimensions". In: *Communications of the ACM* 20 (Feb. 1977), pp. 87–93. URL: https://www.researchgate.net/publication/234809559_Convex_Hull_of_a_Finite_Set_of_Points_in_Two_and_Three_Dimensions.