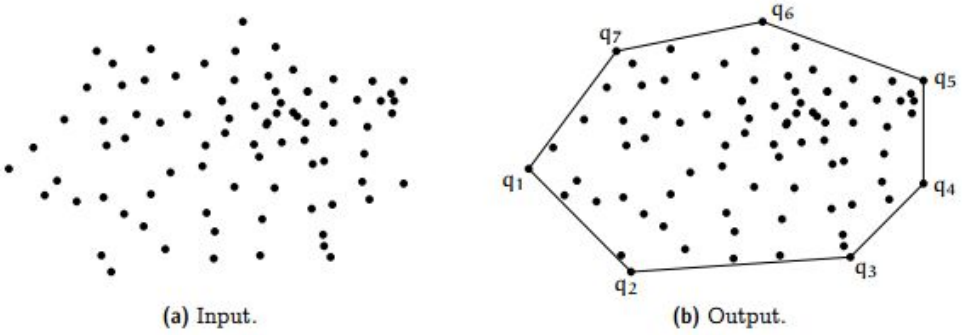
Convex Hull

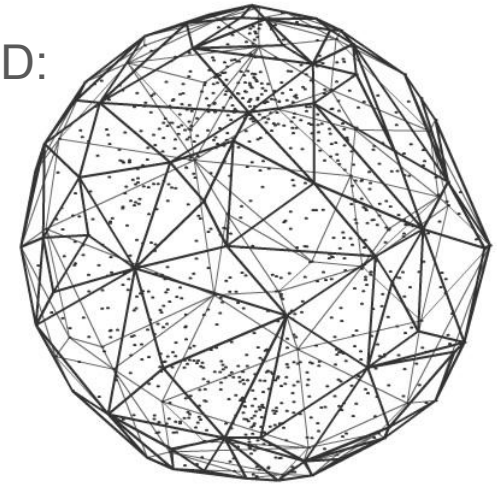
Claudia Cortell, Kyle Edwards, Avighna Suresh

Introduction

In 2D:



In 3D:

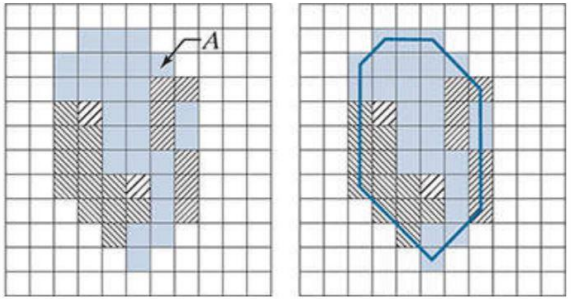


In a set P of n points in a plane, the **convex hull** of P is the **smallest convex polygon containing the points** and the **largest convex polygon whose vertices are points in P**.

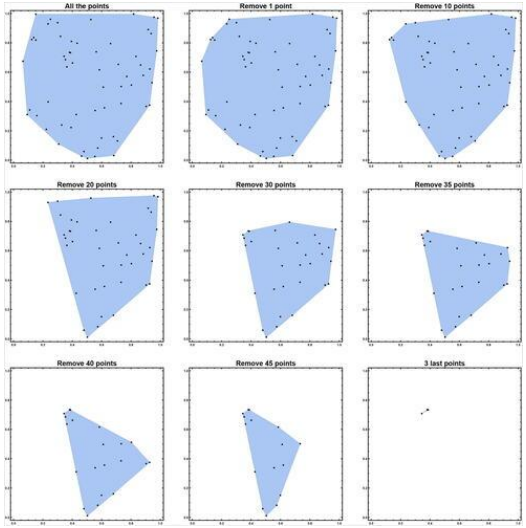
A **3D convex hull** is the **smallest convex polyhedron** that completely encloses a set of points in three-dimensional space

Applications

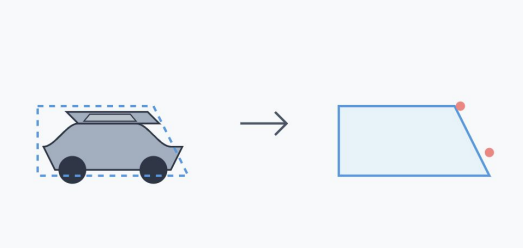
Image Processing:



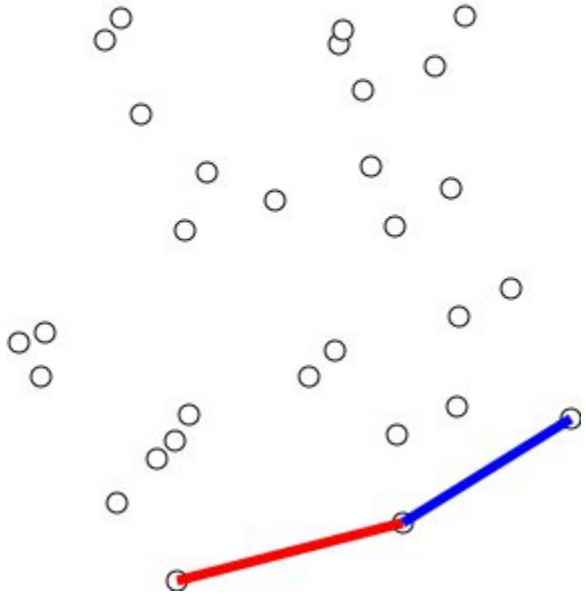
Epidemiology:



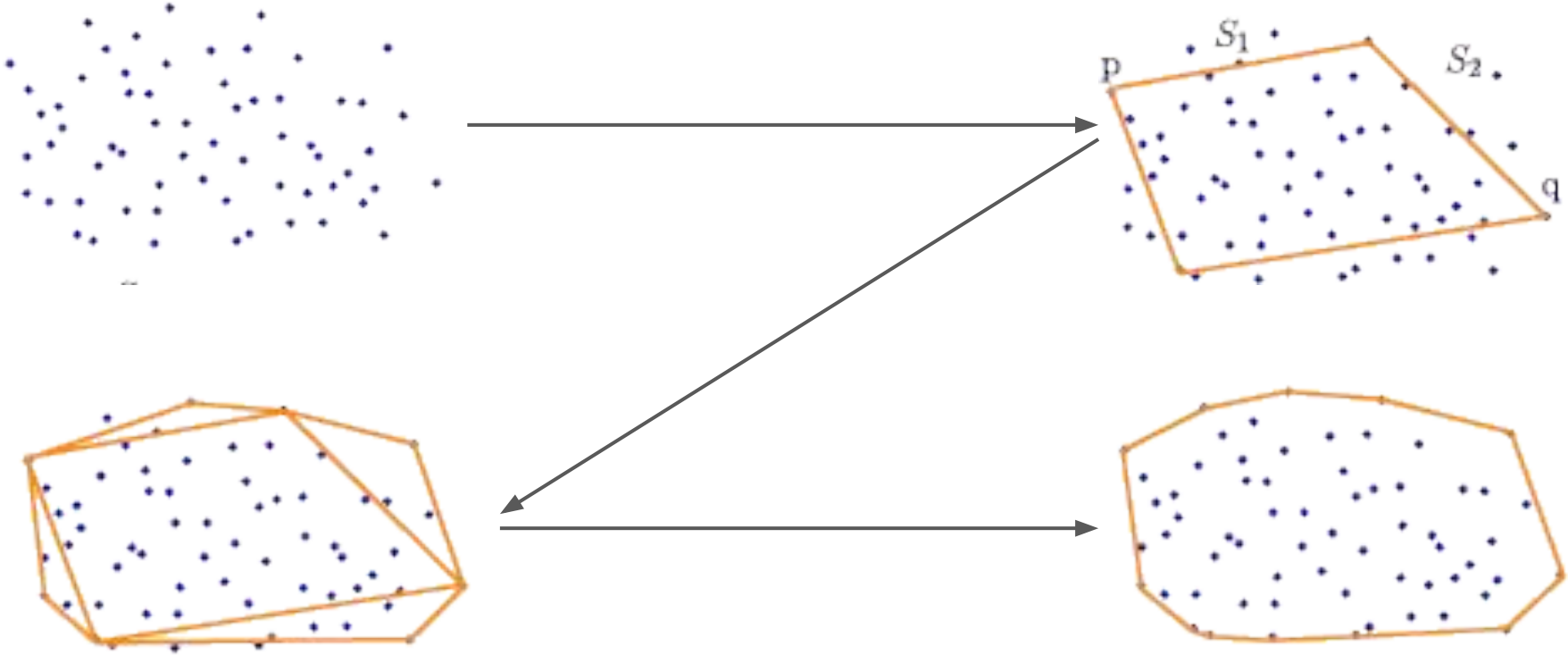
Collision Detection:



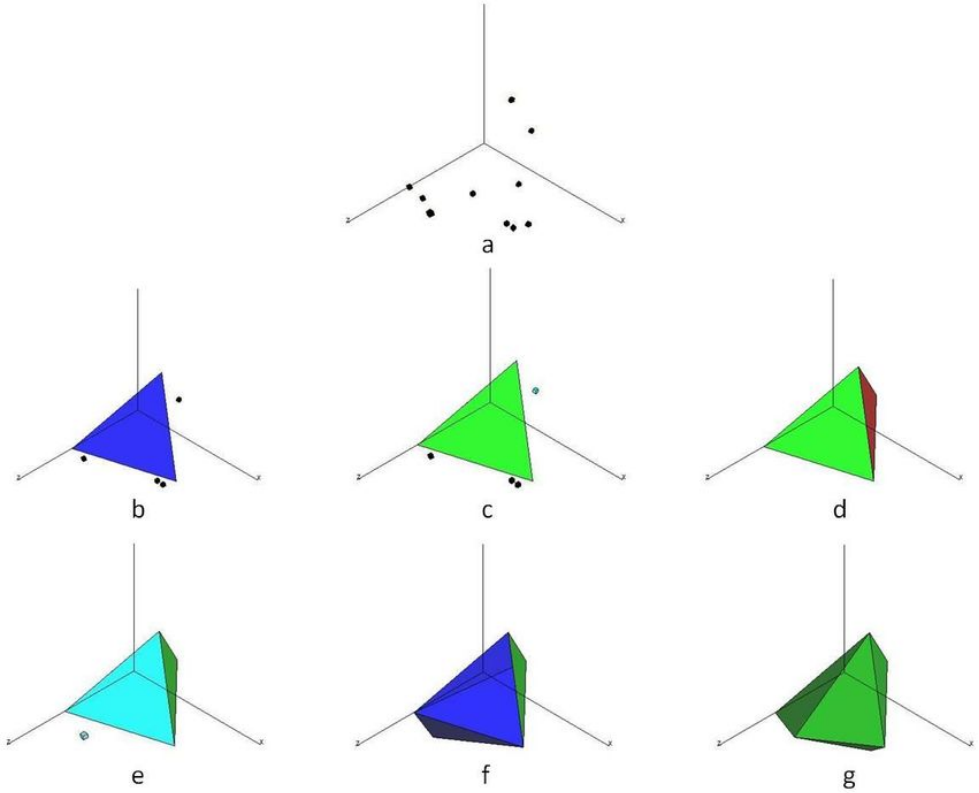
Algorithms: Graham Scan



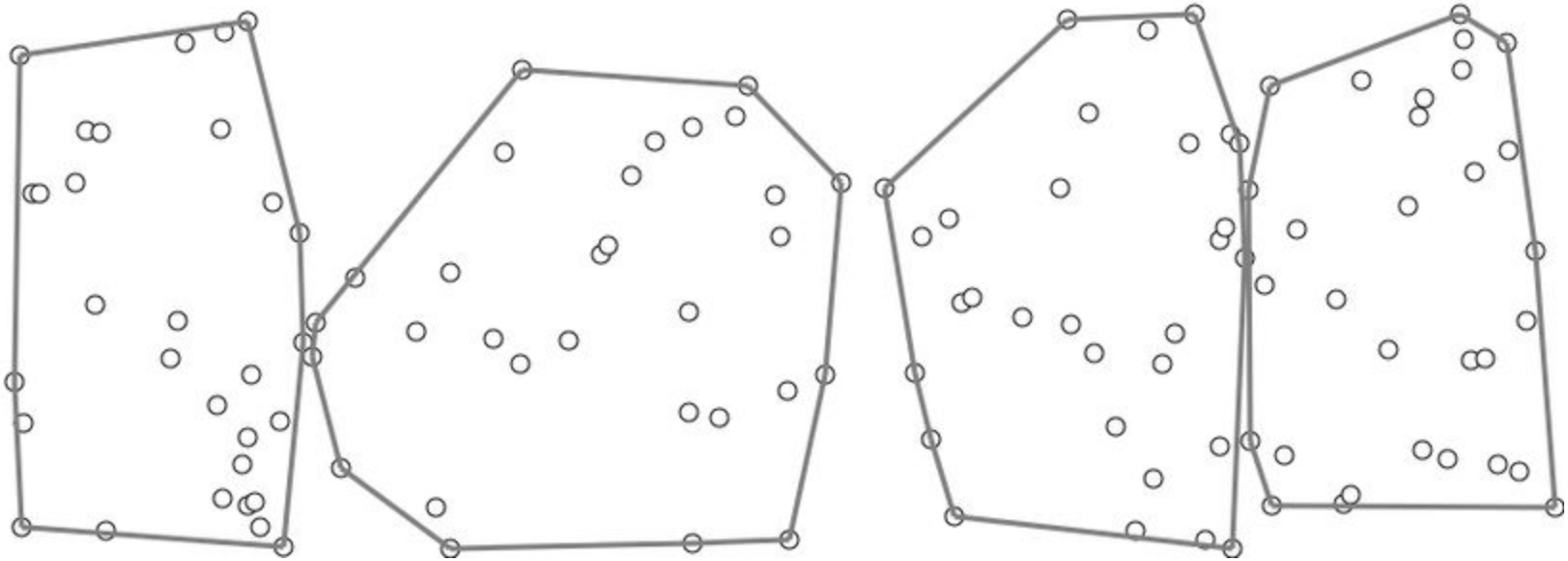
Algorithms: QuickHull



Algorithms: QuickHull (3D)



Algorithm: Chan’s



Parallelizing QuickHull

2D

```
concat (map (quickHull2Par) lines `using` parList rdeepseq)
```

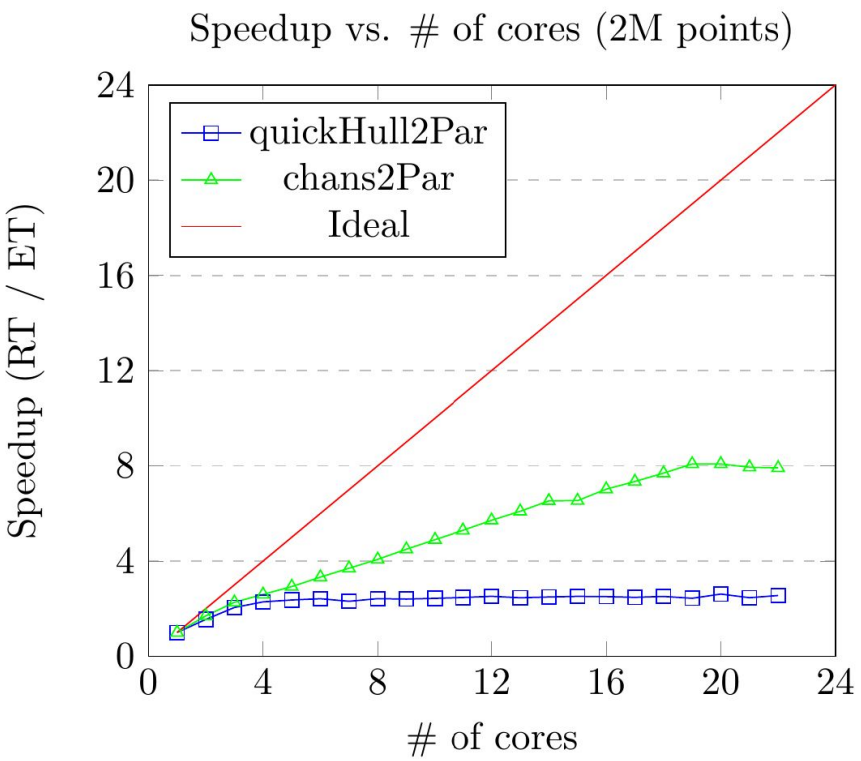
3D

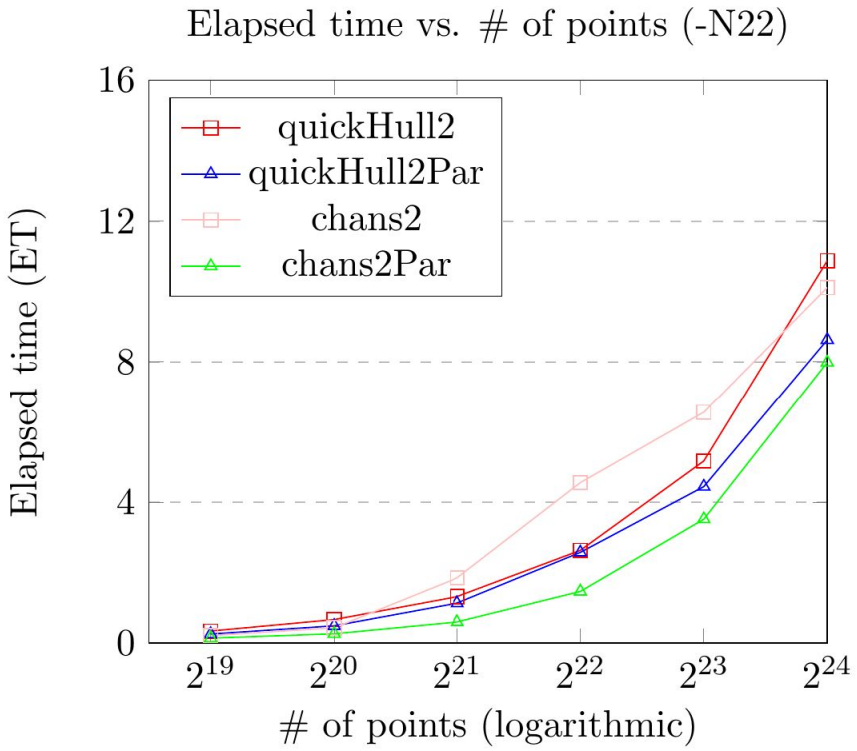
```
concat (map (processPointsParallel 0 epsilon points)
        initialFaces`using` parList rdeepseq)
```

Parallelization: Chan's Algorithm

```
subHulls = map findHullSeq subPoints `using` parBuffer 32 rdeepseq
```

Results


Speedup vs. # of cores (2M points)
quickHull2Par
chans2Par
Ideal
Speedup (RT / ET)
# of cores
0
4
8
12
16
20
24


Elapsed time vs. # of points (-N22)
quickHull2
quickHull2Par
chans2
chans2Par
Elapsed time (ET)
# of points (logarithmic)
0
4
8
12
16
2^{19}
2^{20}
2^{21}
2^{22}
2^{23}
2^{24}

Threadscope

QuickHull

Chan’s Algorithm

Bottlenecks: File I/O

- Initial Implementation: reading points from a file
- Problem: algorithms became I/O bound
- Solution: generate random points instead

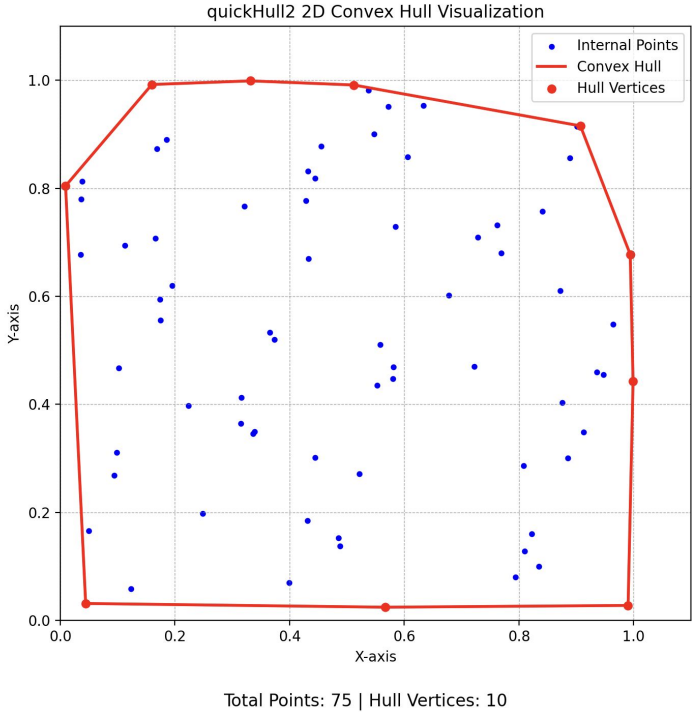
Bottlenecks: List Length Calculation

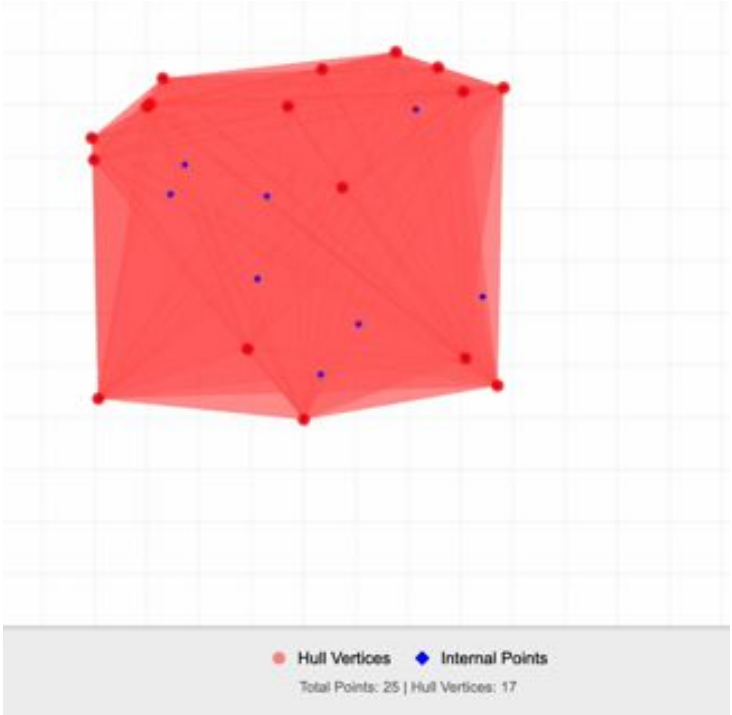
- Initial Implementation: `length ps` in Chan's sub-hull calculations
- Problem: took forever to do
- Solution: pass n as a parameter

Design Choices

- QuickHull:
  - Limit parallel depth (e.g. nfib4)
- Chan’s
  - List -> Vector for binary search
  - Approximating size of hull
  - parBuffer over parList

Visualizations





Thank you!
Any questions?