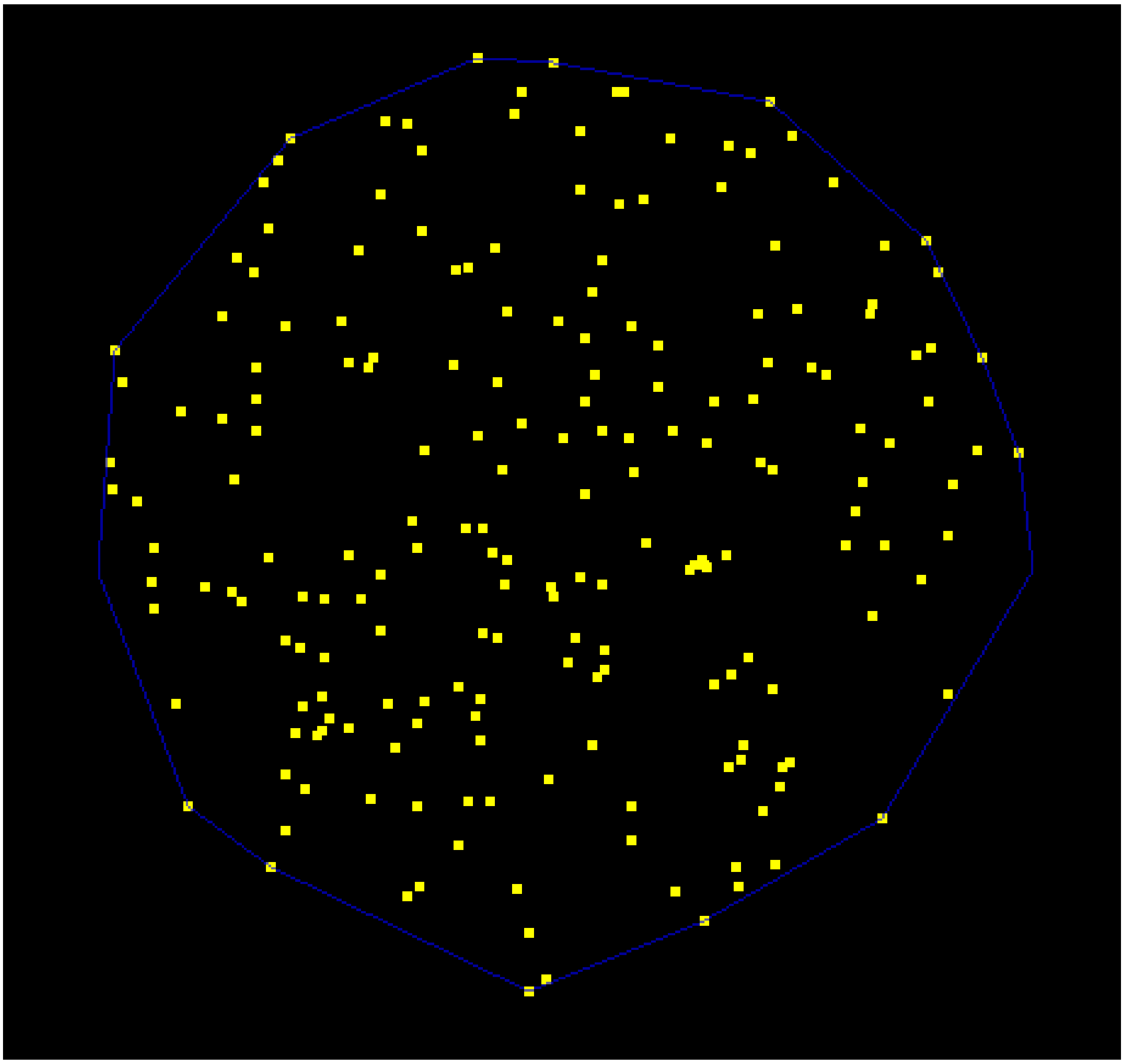
Convex Hull

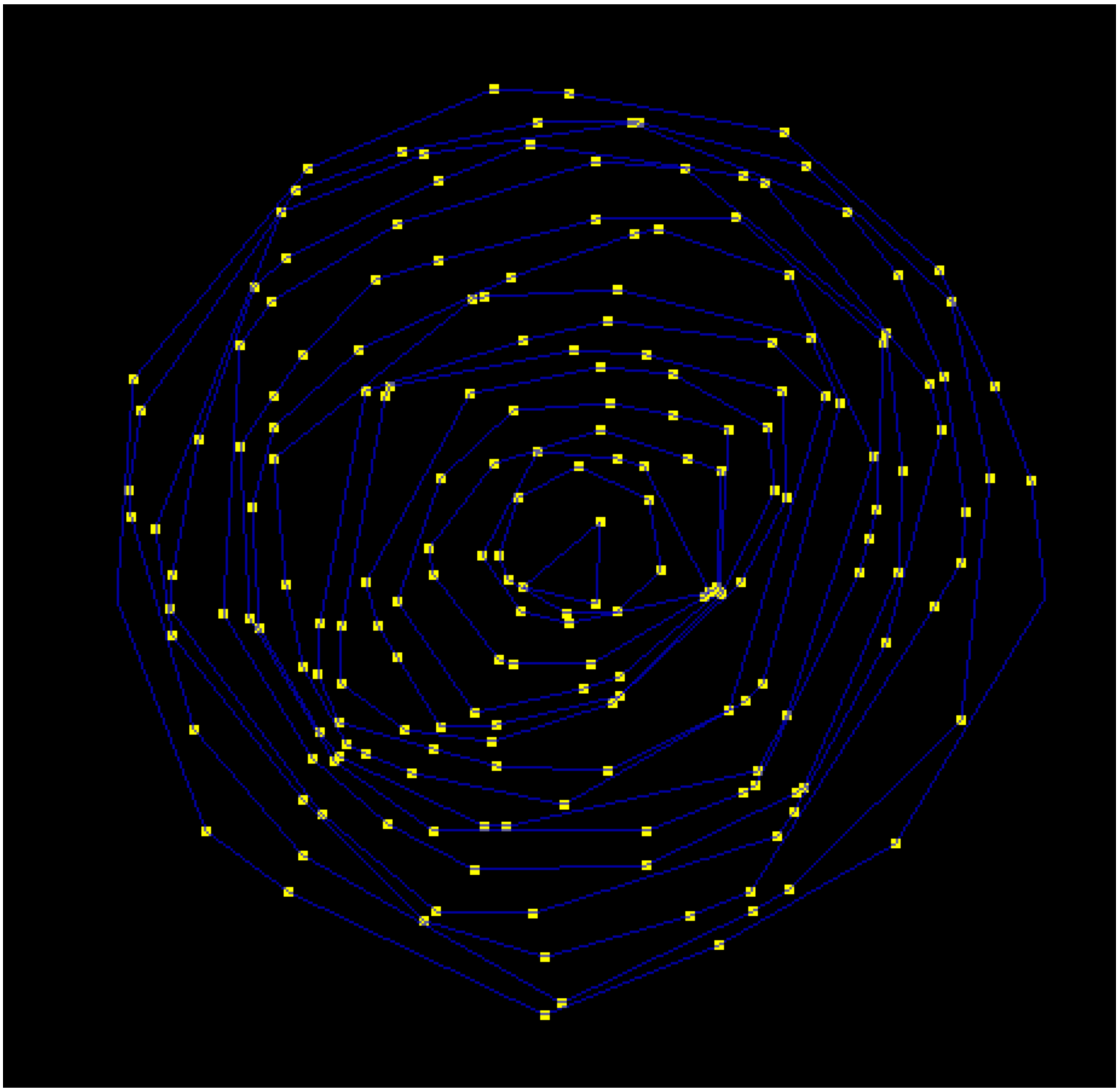
Several algorithms exist to compute the convex hull of 2d and 3d polygons. Of the 2d versions, Andrew's monotone chain algorithm is a particularly simple one to implement. It's also very fast in practice - O(nlogn) for unsorted points and O(n) for sorted points.

The basic idea works as follows:

1. Sort the points lexicographically by the x coordinate (and by y in case of a tie).
2. Build the lower hull: Iterate through the ordered points. On each iteration, while there are at least two points in the hull and the angle between the last two points of the hull and the next point of the ordered set is negatively oriented (clockwise), remove the last point of the hull. Else, add the point.
3. Build the upper hull: Iterate through the ordered points backwards. On each iteration, while there are at least two points in the hull and the angle between the last two points of the hull and the next point of the ordered set is negatively oriented (clockwise), remove the last point of the hull. Else, add the point.
4. Combine both the upper and lower hull into one (and avoid duplicates).

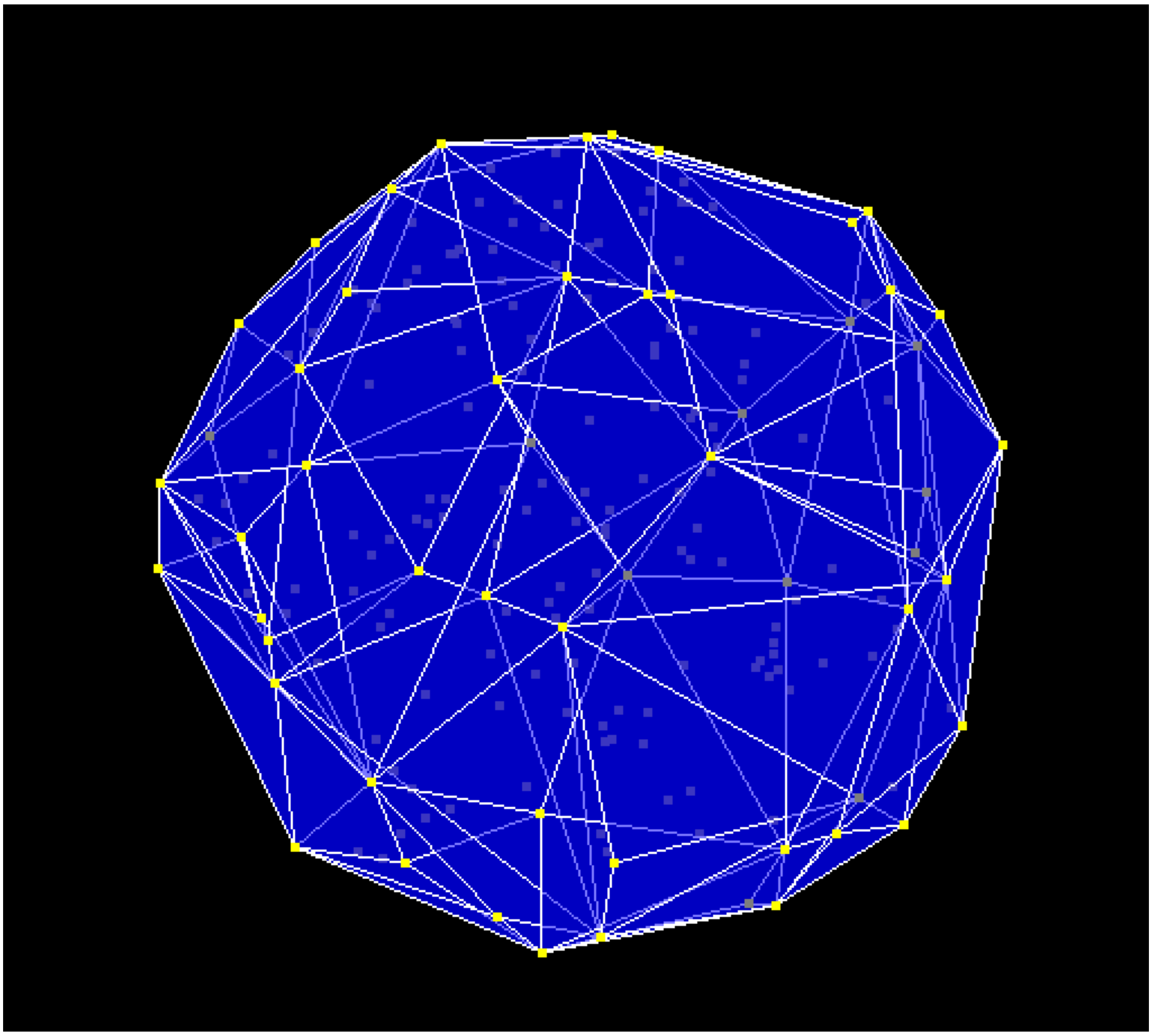


One can also create onion peels by repeatedly using the same algorithm on the remaining point set.



Andrew’s monotone chain algorithm does not carry over to 3d. A simple but slow (O(n^2)) algorithm to construct the 3d convex hull is the incremental hull algorithm. It works as follows:

1. Construct a tetrahedron by connecting the first four points of the input point set and add it to a set of hull faces.
2. For the remaining points
   a. Determine the set of faces visible to the current point. This can be checked by plugging the location (x,y,z) of the current point into the equation of the plane ax + by + c + d. The sign determines the visibility.
   b. Determine the horizon edges for the current point. These edges form the boundary of the faces visible to the point.
   c. For each edge, create a new face by connecting the edge to the current point.
   d. Delete all the visible faces.



Algorithms exist to compute the 3d convex hull in O(nlogn) time but they are considerably more difficult to implement.

Implementation: https://github.com/rohan-sawhney/convex-hull