Concave Hull Algorithms Description

Yaghyavardhan Singh Khangarot

March 26, 2019

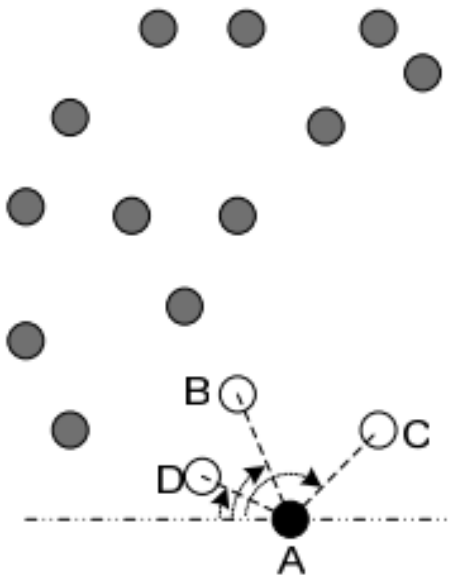
1 CONCAVE HULL:A K-NEAREST NEIGHBOURS APPROACH

The goal of the algorithm is, given an arbitrary set of points in a plane, to find the polygon that best describes the region occupied by the given points. The algorithm is taken from the paper **CONCAVE HULL: A K-NEAREST NEIGHBOURS APPROACH FOR THE COMPUTATION OF THE REGION OCCUPIED BY A SET OF POINTS (Adriano Moreira and Maribel Yasmina Santos)**.

1.1 K-Nearest Neighbours Approach

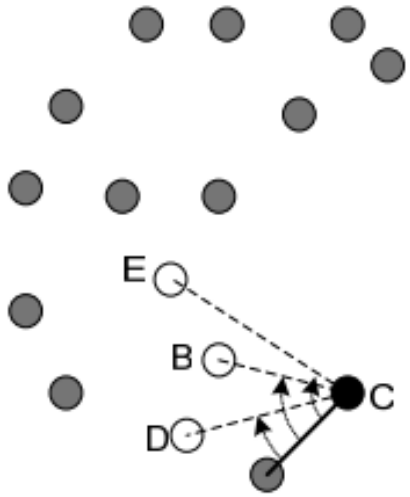
1. The first step of the process is to find the first vertex of the polygon (point A in figure 1) as the one with the lowest Y value.

Figure 1:



2. In the second step, the k points that are nearest to the current point are selected as candidates to be the next vertex of the polygon (points B, C and D in figure 1, for k=3). In this case, point C is selected as the next vertex of the polygon, since it is the one that leads to the largest right-hand turn measured from the horizontal line (xx axis) that includes the first point (point A). Since C is now a vertex of the polygon (as well as A), it must be removed from subsequent steps while searching for the k-nearest neighbours.

Figure 2:



3. In the third step, the k-nearest points of the current point (point C) are selected as candidates to be the next point of the polygon (points B, D and E in figure 2). In this case, the point that results in the largest right-hand turn, corresponding to the largest angle between the previous line segment and the candidate line segment, is selected (point E in figure 2). As before, point E is now part of the polygon and will never be considered in the next steps.

4. The process is repeated until the selected candidate is the first vertex. For the first vertex (point A) to be elected as a candidate, it must be inserted again into the data set after the first four points of the polygon are computed (before that, if the first point is selected as the best candidate, a triangle is computed). By the end of the process, the polygon is closed with the first and the last point being the same (point A).

1.2 Special Cases

1. When the selected candidate results in a polygon edge that intersects one of the already computed edges.In figure , the candidate that results into the largest right hand turn is point B. However, this candidate leads to a polygon edge that intersects one of the existing edges and, therefore, should be discarded. In cases like this, the next candidate should be considered (point G). If none of the candidate points (the k-nearest neighbours) is acceptable, then a higher number of neighbours must be considered, by increasing the value of k and starting again.

Figure 3:

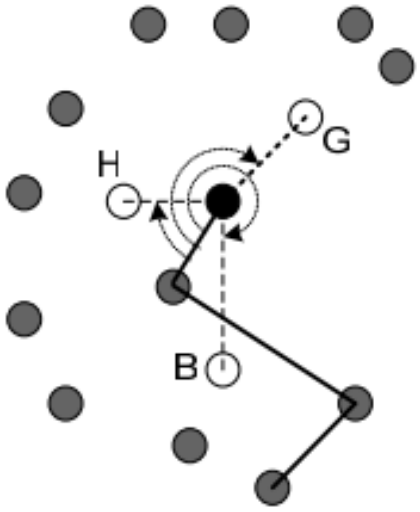
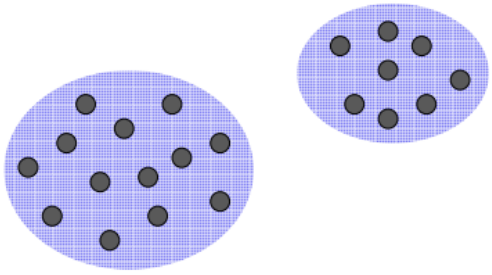


Figure 4:



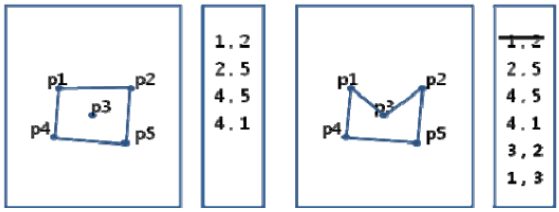
2. When the spatial density of the initial set of points is not uniform. In this case, the first point of the polygon is in the lower-left region (the point with the lowest Y value) and, therefore, the process starts by looking for candidates that are near this first point. However,since the points in the upper-right group are too far away from the points in the lower-left group, they are never considered as candidates if the number of neighbours (value of k) considered in each step of the process is small. As a consequence, the points in the upper-right group are left out of the polygon. To solve this issue, a higher number of neighbours must be considered.

2 2-Dimensional Concave Hull Algorithm

The strategy used in paper (**A New Concave Hull Algorithm and Concaveness Measure for n-dimensional Datasets* JIN-SEO PARK AND SE-JONG OH**) to produce a set of concave hulls was to identify a convex hull using a known algorithm, and 'dig' it to produce a concave hull with an appropriate depth. The digging level was determined by the threshold value N. Using a smaller N induces a sharper surface (see Fig. 2). If N has a sufficiently large value, our algorithm does not dig, and will return a convex hull, instead of a concave hull.

1. First, a set of convex hull edges is selected (Fig.5 (a)) and the threshold value N is chosen.

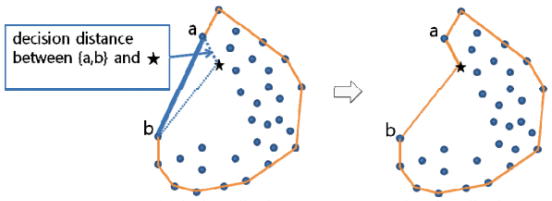
Figure 5: Example of Concave list



(a) *convexList* before digging. (b) *concaveList* after digging.

2. Second, the nearest inner points from the convex hull edge are identified, after which the shortest distance between the nearest inner point and the edge's points is identified. This distance is called as a 'decision distance' (see Fig.6 (a)).

Figure 6: Example of decision distance and digging



(a) before digging. (b) after digging.

3. Third, a decision to dig or not is made by comparing N with the decision distance. If (length of edge)/(decision distance) $>$ N, then we execute digging process (see Fig.6 (b)).

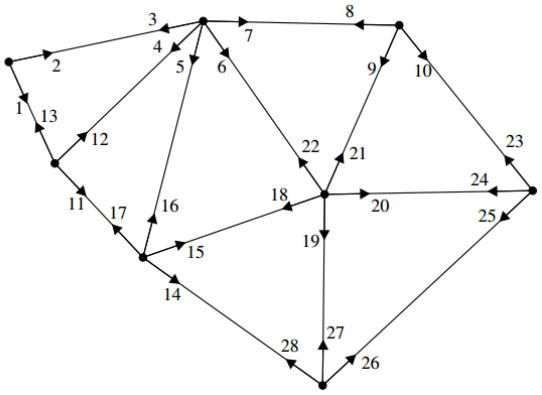
4. Fourth, the second and third procedures are repeated until there is no inner point for digging. The final output of this algorithm is a set of edges that forms a concave hull for the given dataset (see Fig.5 (b)).

3 The χ (chi) algorithm

The χ -shape algorithm is based on "shaving" exterior edges (edges that bound only one triangle) from a triangulation of the input point set in order of the length of edges and subject to a regularity constraint. The algorithm itself has a time complexity of $O(nlogn)$, where n is the number of input points. The algorithm is described in the paper **Efficient generation of simple polygons for characterizing the shape of a set of points in the plane (Matt Duckhama,$*$, Lars Kulikb, Mike Worboysc, Antony Galtond)**.
Summary of the algorithm for an input point set P and a length parameter l.

1. Generate the Delaunay triangulation of the set of input points P and the list of boundary edges B sorted in descending order of edge length.Determining whether a particular edge is a boundary edge can be achieved in constant time by checking for 3-cycles of darts in the combinatorial map , also determine whether a particular vertex is a boundary vertex.

Figure 7: Example of triangulation structured as combinatorial map



2. Remove the longest exterior edge from the triangulation such that:
   (a) the edge to be removed is longer than the length parameter l and
   (b) the exterior edges of the resulting triangulation form the boundary of a simple polygon ,i.e edge belongs to B.
   (c) When an edge e is removed, the two new boundary edges that are revealed by the removal of e are added to the list of boundary edges B, respecting the edge-length ordering of B
3. Repeat 2 as long as there are more edges to be removed.
4. Return the polygon formed by the exterior edges of the triangulation.