Visibility graph for off-road pathfinding

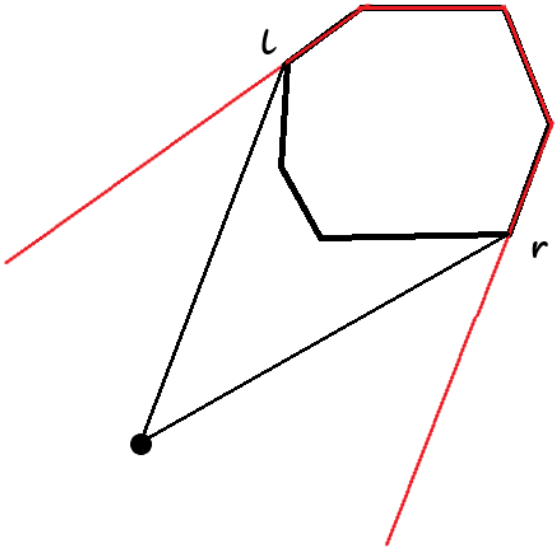
Denis Kozub

1. Given a set of **h** polygons with $\mathbf{n}=\sum_{i=1}^{h}\mathbf{n_i}$ vertices the goal is to build a reduced visibility graph on a given area with **O(nh log n)** time complexity. It is motivated by the desire to build routes in real-time, without the user waiting for computation. Existing algorithms [1] [2] [3] are either slower or do not build a reduced graph. The set complexity can only be achieved by building an estimated graph, which would be as close to the correct one as possible.
2. To achieve real-time speed, Ramer–Douglas–Peucker approximation algorithm [4] is used to simplify polygons and polylines. Since computation time should not depend much on size of the area, hierarchical approach is used. It removes small polygons from consideration and alteres RDP parameter due to the size of the area to keep the amount of graph edges in the same range.
3. Another important moment is the ability to find vertices of the visibility graph, incident to a given point without building the graph itself. This would speed up the pathfinding algorithm.
4. The visibility graph algorithm for each point can be divided into two steps: finding supporting lines to each polygon and building a visibility graph for these lines. Time complexity for both steps should not exceed **O(nh log n)**.
5. Precomputing includes building a convex hull for each polygon using Chan algorithm [5] with time complexity $\sum_{i=1}^{h}\mathbf{O(n_i \log h_i)}$ and finding polar angles to each vertex or a polygon from one of the vertices of a polygon for each polygon which takes **O(n)** time.
6. Finding supporting lines from a point to a convex polygon takes $\mathbf{O(\log n_i)}$ time (see appendix). Finding supporting lines from a point to a non-convex

polygon takes $\mathbf{O(n^2)}$ time (brute force). $\mathbf{O(\log n_i)}$ complexity is achieved every time, when a point does not belong to the polygon's convex hull (in almost all cases). Point localization in a convex polygon takes $\mathbf{O(\log n_i)}$ time [6]. Therefore, step one has time complexity $\mathbf{n \sum_{i=1}^{h} c * \log n_i = O(nh \log n).}$ Building a visibility graph for **2h** supporting lines takes **O(nh log n)** [3]. This means that the goal is achieved.

Appendix

Finding supporting lines from a point to a convex polygon. Algorithm by Denis Kozub. This algorithm works with the quality of a convex polygon in respect to a point: semi-planes forming the polygon are sorted by the fact of containing the point. Therefore 2 subsets are formed, points dividing them are supporting points which are to be found.



To find supporting points using binary search, 2 elements, each one of them belonging to one of the subsets respectively, are required. One of them (point1) can be found by picking a random vertex of a polygon. The second one (point2)

can be found as one of the vertices, forming a segment, which is crossed by line (point, point1). This can be done similarly to locating a point in a convex polygon [6] in **O(log n_i)** time. Having found point1 and point2, binary search can be applied to find supporting points in **O(log n_i)** time as well. A working implementation can be found in geometry/supporting_convex.py.

References

1. Danny Z. Chen∗ and Haitao Wang, “A NEW ALGORITHM FOR COMPUTING VISIBILITY GRAPHS OF POLYGONAL OBSTACLES IN THE PLANE”

2. Subir Kumar Ghosh & David M. Mounts, “AN OUTPUT-SENSITIVE ALGORITHM FOR COMPUTING VISIBILITY GRAPHS”

3. M. de Berg, O. Cheong, M. Kreveld, M. Overmars «Computational Geometry» // Springer, pp. 326-331. 2008.

4. wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

5. Timothy M. Chan. "Optimal output-sensitive convex hull algorithms in two and three dimensions". Discrete and Computational Geometry, Vol. 16, pp.361–368. 1996.

6. Препарата Ф., Шеймос М. Раздел 2.2: Задачи локализации точек. // Вычислительная геометрия: введение. — Москва: Мир, 1989.