Off-road routing methods based on visibility graphs

Denis Kozub, denikozub@gmail.com

Abstract. The problem of off-road routing and terrain discretization with the help of visibility graphs is considered. A polygon approximation method is proposed as applied to constructing a visibility graph on a plane, as well as pathfinding without constructing a complete graph for solving the problem of off-road navigation. The developed algorithm for finding supporting lines to a convex polygon on a plane is described. The ways of implementing a hierarchical approach to building a visibility graph are analyzed.

Keywords: navigation, computational geometry, visibility graph

1. Introduction

The article discusses approaches to solving the problem of building routes over rough terrain. All currently existing mapping and navigation services provide the ability to build a route between given points either along roads and sidewalks, or along country roads. At the same time, the need for off-road navigation arises, for example, among tourists, hikers and rescuers. The underlying algorithms can also be used to analyze ways to make the most efficient use of territorial and natural resources: in particular, in the construction of roads and railways, oil and gas pipelines, and power lines.

2. Terrain discretization

Although geographic data is inherently continuous, there are two main ways to discretize it - vector graphics and raster graphics. In the first case, terrain data is stored on a plane as a set of polygons and polylines, each of which is assigned some attributes that describe geographical properties. In the

second case, the minimum unit of information is a pixel, which also stores attribute data. Raster data is obtained using satellite imagery, and vector data is obtained by processing satellite images by a person or a program using computer vision algorithms. This article discusses the vector format of data representation.

Separately, we describe the specifics of setting the problem of constructing routes over rough terrain. Further, we will consider geographic data in two-dimensional space (the position of each point is given by a pair of values in some coordinate system). Objects (natural or infrastructure) on the plane are given by simple polygons and broken lines and contain information about their type. In what follows, without loss of generality, we will consider only the set of polygonal objects $P = \{P_i\}$, $|P| = h$, whose vertices are set counterclockwise.

When it comes to routing, the natural and most commonly used data structures are the graph and the grid. The grid, in the case of geographic data, is represented by a raster image (grid map), and by default defines 4 possible directions of movement from each cell - left, down, right, up (or 8 if diagonal routing is enabled). The graph, in turn, can be built in several ways from vector data.

The first way is a road map based on the concept of a trapezoid map [1, p. 122]. This approach is often used in robotics and computer games, but cannot be used to solve our problem. He considers polygons as insurmountable obstacles, which in our case is not always true and will depend on the type of natural object.

The second way is the visibility graph [1, p. 323]. By definition, for a set of polygons on a plane, the vertices of the visibility graph are the vertices of the polygons, and an edge between two vertices exists if the segment on the plane connecting them does not intersect any of the edges of the polygons. This definition can be trivially extended for a set of polylines. We will consider weighted undirected graphs, while the topic of determining the weights of edges

deserves a separate article, and will not be discussed in this paper. It should be borne in mind that in order to solve a problem in which polygons are natural objects, the visibility graph must be supplemented with all diagonals of polygons, the construction algorithm of which is also not considered in this article.

3. Building a visibility graph

It was proved in [1, p. 326] that the shortest path between two points on the plane for a set of polygonal obstacles passes along the edges of the visibility graph. Moreover, it is clarified that in this case the visibility graph can be reduced - its edges are common tangents for a pair of polygons.

Let $n = \sum_{i=1}^{h} n_i$ where n_i - number of vertices in the polygon P_i. Then n is the total number of vertices in the visibility graph. The brute-force algorithm for its construction will have an estimate of the time complexity $O(n^3)$. In [1, p. 326], an algorithm is given that uses the sweeping line principle and has a time complexity of $O(n^2 \log n)$. To construct a visibility graph for a set of segments on a plane, an algorithm [2] was proposed that runs in $O(n^2)$ time and uses the principles of duality of segments and points in space. An output-sensitive algorithm for constructing a reduced visibility graph using triangulation methods and having a time complexity of $O(n + h^2 \log n)$ was proposed in [3].

Compared to other algorithms, the sweeping line method has several advantages: it is fast on small inputs, it is easy to implement, and it also builds visibility edges for each query point q separately in O(n log n) time. The last property is of particular interest, since in the case of solving the problem of navigation over rough terrain, the construction of the entire visibility graph becomes a computational problem. Heuristic algorithms for finding the shortest path on a graph in most cases do not discover all the edges of the graph, which makes it possible to search for the shortest path in the graph without its

preliminary construction. To do this, it is necessary to be able to search for edges incident to a given vertex - query point q (visibility edges). This possibility is provided by the sweeping line algorithm. It should be noted that the described approach also makes it possible to implement the search for the shortest path in a dynamically changing graph (for example, when weather conditions or time of day change, the weights of the graph edges also change) without its complete rebuild.

4. Polygon approximation

We have proposed an algorithm for approximating polygons in solving the problem of navigation on rough terrain by constructing a visibility graph for a set of polygons defining natural objects. Let us introduce the concept of a supporting line, which is closely related to the concept of visibility. A line is called supporting to a figure on the plane if it has common points with it, but does not contain internal ones. Fig. 1 shows examples of supporting lines (q_1l_1, q_1r_1, q_2l_2, q_2r_2).

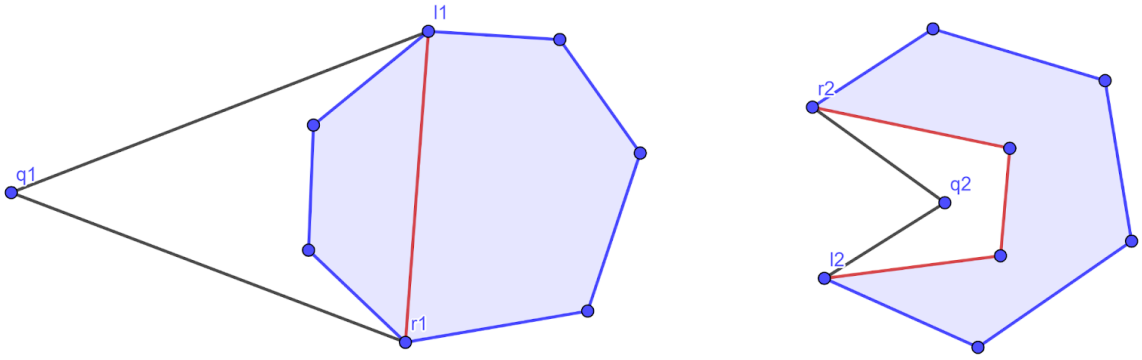

Fig. 1. Supporting lines to polygons passing through given query points q_1 & q_2.

Let at some point in time it is necessary to construct the edges of the visibility graph incident to a given vertex q. We will construct supporting lines passing through q to each of the polygons P_i. Consider the case when $q \notin \overline{CH}$ (P_i), where CH(P_i) - convex hull of P_i. In this case, the supporting lines ql and qr

to P_i will coincide with the supporting lines to $CH(P_i)$. The preprocessing of P_i consists in constructing its convex hull, which can be done in $O(n_i \log k_i)$, where k_i is the size of the hull, using Chen's algorithm [4]. Without loss of generality, we consider the case of P_i being convex. We aim to achieve time complexity $O(\log n_i)$. To do this, we use the property of a convex polygon: the half-planes whose intersection defines the polygon are sorted by the inclusion indicator of any point of the plane. In the case under consideration, the half-planes uniquely defined by a pair of vertices are divided into two non-empty subsets C and NC (containing and not containing q).

Checking whether a line is supporting to a convex polygon using cross products takes constant time. To organize a binary search over the polygon vertices to find the supporting points l and r, you need to find two vertices m and M contained in C and NC. In 1979, an algorithm for finding them in $O(\log n_i)$ was proposed [5]. It is based on a recursive descent through a modified AVL tree that stores the vertices of a polygon ordered by their polar angle. At each iteration, it is checked whether the leftmost leaf of the tree and its root are the desired points. The disadvantages of this algorithm are its recursiveness and a fixed data structure for storing a polygon.

We have proposed an alternative algorithm for finding the points m and M. Let us draw a line through q and the vertex of the polygon m, which is the first in the record of the vertices of the polygon. Let's find the vertex or edge at which the given line intersects the polygon for the second time (Fig. 2). To do this, we use another property of a convex polygon: its vertices are ordered relative to polar angles calculated from some internal or boundary point of the polygon. P_i preprocessing includes calculating these angles from point m in $O(n_i)$ time. Using binary search on these corners, we localize the crossed vertex or edge. In the latter case, we take any of the vertices defining the edge. If the found vertex matches m, then m is one of the supporting points, and the binary search is organized over the remaining vertices. Otherwise, we have found the

points m and M, and the binary search is organized independently over the subsets into which m and M partition the vertices of the polygon. Logarithmic complexity is achieved.

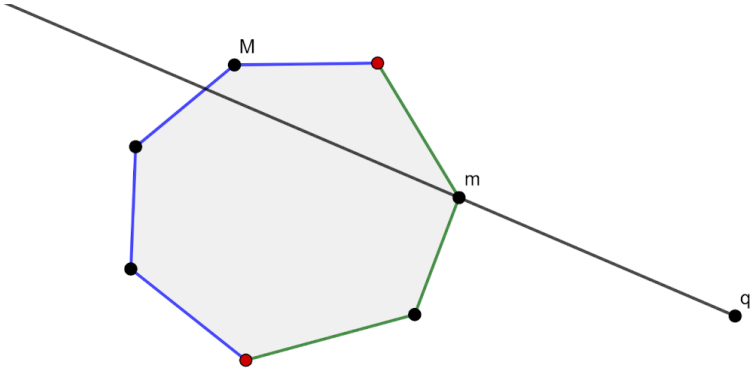

Fig. 2. Method for finding m and M contained in C (blue) and NC (green)

In the considered case, the polygon is approximated by the found segment lr. To find the edges of the visibility graph incident to the query point q, using the sweeping line algorithm in O(h log h) time, visibility edges are constructed for q and a set of h segments $l_i r_i$ built from the supporting points of each of P_i. The localization problem for q in P_i can be solved in $O(\log n_i)$ [6]. Using the inequality $\sum_{i=1}^{h} log\, n_i < h * log(\sum_{i=1}^{h} n_i) = h * log\, n$, we arrive at an estimate of the time complexity Ω(h log n) of the algorithm for constructing supporting lines and Ω(h log nh) of the algorithm for constructing incident edges.

If $q \in \overline{CH}(P_i)$ finding supporting points takes quadratic time, however, due to the specifics of the problem, such cases are extremely rare in practice. In this case, the polygon is approximated not by a segment, but by a set of segments connecting the points l and r (Fig. 1, right). It should also be noted that there are configurations in which the use of this approach does not allow one to obtain all the visibility edges for the set P correctly; additional checks are

required for the complete occurrence of one of the polygons in the closure of the convex hull of another polygon.

For the point q, which is a vertex of the polygon P_j, for the correct construction of visibility edges for the set of segments $l_i r_i$, the concept of a restriction angle α is introduced. The ends of the segments that do not fall into α will not be connected to q by an edge. If $q \in CH(P_j)$, $\alpha > \pi$ is set by q and its convex hull neighbors. Otherwise, $\alpha = \measuredangle (l_j q r_j) \leq \pi$. Checking whether the point belongs to α using vector products is performed in constant time.

5. Hierarchical approach

Suppose we need to build a car route several thousand kilometers long. The road graph of a territory can contain such a large number of vertices and edges that the user will have to wait for the result of calculations for more than one hour, especially if the calculations are performed on a local machine. To solve this problem, a hierarchical approach is used on the existing road graph. Its essence is as follows: the road graph is divided into local clusters, the optimal distance to overcome which is calculated in advance and cached. The search for the shortest path between clusters is performed using one of the pathfinding algorithms. In this case, local clusters can be combined into clusters of a higher level, so that the number of hierarchy levels is not limited. For example, building a route thousands of kilometers long will be divided into three parts: the path from the starting point to the highway, the path along the highways, the path from the highway to the end point. It should be noted that the constructed route will not always be the shortest. The hierarchical approach to pathfinding is also used in computer games [7].

In the off-road navigation problem we are considering, the number of visibility graph vertices will be thousands of times greater, and the number of edges - millions of times, since not only roads, but also natural objects will be considered in the indicated territory. In this case, the hierarchical approach also

finds its application: visibility graphs can act as separate clusters in the territories between the start and end points and the road graph - it is in these areas that the task of navigating over rough terrain is most acute. At the same time, there is no need to build the entire visibility graph - only the geometry in the indicated areas will be considered. The above two clusters, in turn, can also be divided into smaller local clusters - in places where the largest number of natural objects is concentrated. This solution can be used to expand the capabilities of existing navigation services.

In the previously considered approach to finding the shortest path along the visibility graph without directly constructing the entire graph, the hierarchical approach finds its place in working with the geometry of natural objects. Depending on the size of the territory where the search for shortest paths is carried out, or depending on the size of the clusters into which the visibility graph is divided, the geometry of objects can be simplified. To simplify polygons and broken lines, for example, the Ramer-Douglas-Peucker algorithm [8] can be used, to simplify the geometry in the form of a graph, the edge contraction algorithm can be used, and objects that are too small relative to the size of the territory can be removed. In this case, to search for the shortest path at lower levels of the hierarchy, the geometry of the object can be restored.

References

1. M. de Berg, O. Cheong, M. Kreveld, M. Overmars «Computational Geometry» // Springer, 2008
2. T. Sevilmiş, B. Mizrahi «Implementation of the visibility graph construction for a set of polygons for 2D in O(n2)» // Bilkent University, 21p. 2021
3. H. Rohnert «Shortest paths in the plane with convex polygonal obstacles» // Inform. Process. Lett., pp.71–76, 1986.

4. Timothy M. Chan «Optimal output-sensitive convex hull algorithms in two and three dimensions» // Discrete and Computational Geometry, Vol. 16, pp.361–368. 1996.
5. F.P. Preparata «An Optimal Real-Time Algorithm for Planar Convex Hulls» // University of Illinois, 4p. 1979
6. M. l. Shamos «Computational geometry» // Yale University, pp.92-95, 1978
7. Botea A., Muller M., Schaeffer J. «Near Optimal Hierarchical Path-Finding (HPA*)» // Journal of Game Development, N3, pp.1-30. 2004
8. Urs Ramer, «An iterative procedure for the polygonal approximation of plane curves» // Computer Graphics and Image Processing, 1(3), pp.244–256. 1972