Geodesic distance numerical computation on compliant mechanical parts in the aircraft industry

N P Popov and T A Pogarskaia

Peter the Great St. Petersburg Polytechnic University, Polytechnicheskaya 29, St. Petersburg 195251, Russia

E-mail: nikpopov@bk.ru

Abstract. The problem of geodesic distance computation in the Aircraft industry is considered in the paper. While simulating the assembly process, in some computational problems the calculation of the shortest distances between different objects is required. The aim of the work is to compare several approaches to geodesic distance numerical computation on test geometry and some of the most common joints of aircraft. The first considered approach represents graph-based algorithms. In the paper, geodesic distances are calculated in the constructed graph using Dijkstra’s algorithm or its modification with binary heap. The second approach is the fast marching method, which is based on the solution of the Eikonal equation and provides efficient geodesic distance computation on triangulated surfaces.

1. Introduction

The necessity for the shortest geodesic paths calculation arises in different areas. A keen interest in image processing and its application greatly contribute to the need. Numerical calculation of geodesic distances often becomes an important intermediate step of complicated research in medicine [1], engineering [2], image and shape recognition [3, 4], and other applications.

During aircraft assembly process simulation, calculation of the shortest distances between objects, such as drills, fasteners, holes or simply over a joint surface, is required in some algorithms. At the same time, assembled parts are usually large-sized complicated constructions and the junction can represent disconnected areas or non-convex surfaces [5, 6]. Since the usual Euclidean distance is not appropriate for measurements on convex surfaces and three-dimensional objects as the considered parts are, alternative methods must be used. In this paper, geodesic distance is proposed as a method that resolves the mentioned problems.

A similar case was considered in [7], where geodesic distance was calculated via finite element nodes over an industrial composite aircraft wing model for further analysis. In the study, the authors used continuous Dijkstra’s algorithm to propagate the shortest paths. Classical Dijkstra’s algorithm was also implemented in [3], where geodesic distance was used as a measure of the geometric structure for 3D object matching. Another popular approach of geodesic distance calculation, the fast marching method, was applied to evaluate similarities between parts of CAD models in automotive and aerospace industries. It was introduced as a numerical method for solving boundary value problems of the Eikonal equation [8].
The mentioned above approaches are most commonly used for geodesic distance numerical calculation [4, 8, 10]. Meanwhile, Dijkstra’s algorithm alone has many different modifications. Thus, for example, in [9] priority queue modifications for algorithms of distance calculation are implemented and provide significant improvement. In [10] an adaptive priority queue was used to improve graph-based algorithms and was tested on polyhedra. It was shown that the approach can be successfully implemented.

However, the choice of an algorithm often is just postulated, being a part of another extensive research. In contrast to computer graphics, we are limited with pre-defined finite element meshes of the models. That raises the issue of the choice of the best algorithm for geodesic distance calculation due to the particular geometry specificities of the regarded parts and for further implementation in aircraft assembly optimization. Consequently, the fast marching method, Dijkstra’s algorithm and its modification with binary heap are considered. The algorithms are tested on finite element models of some common joints of aircraft.

2. Sample and experiment

The aim of the work is to find the best algorithm for further implementation in aircraft assembly optimization and temporary fasteners rearrangement. This circumstance imposes several restrictions on further work. First of all, we are limited with predefined finite element meshes of the models. Commonly used models involve thousands of nodes and meshes are mostly regular. Each model includes several separate parts that have to be joined and each part has its computation nodes. Total covering surfaces of joining regions are of interest where all the calculations are done. Further application of the shortest distance calculation procedure implies its use at the same model for several times while time-consuming optimization. It is therefore important that distance calculation does not lead to total time increase.

The preliminary calculations are needed since graph-based algorithms quite often suffer from approximation errors and the found optimal path sometimes is not actually it even if the distance length has a correct value. Nature of a wrong path appearance is not connected with the mesh quality but only with an approximation of distances between graph vertices which is considered as corresponding edges of the Euclidean distance value [4].

Different ways to solve the problem were proposed and they are mostly aimed at modification of the graph construction procedure: edges weight can be modified in order to specify a direction or more graph vertices can be connected, not only the neighboring ones. The last approach is used in the paper since it satisfies the requirements of the considered problem where high accuracy needed in image processing, for example, is not required.

The implemented algorithms are first tested on a simple surface in order to quality compare the numerical result with the paper [6] where the problem was discussed. The considered surface is described by the following equation

\[ z = 0.5 \sin(4\pi x) \sin(4\pi y), \quad x \in [0,1], y \in [0,1]. \]  

(1)

First of all, contour lines of the calculated distances were checked in order to confirm that the shortest paths between distant points are measured correctly. Distances from the central point (0, 0) to all other points were calculated. The results of both algorithms are qualitatively in agreement with the results presented in [8] and the shortest paths are calculated properly, see figure 1.
Figure 1. Qualitative comparison of Dijkstra’s algorithm, Dijkstra’s algorithm with binary heap and the fast marching method with results of [8], contour lines

Computational time depending on number of vertices is computed for nested meshes and presented in figure 2. Dijkstra’s algorithm with binary heap is the fastest for meshes more than 5000 elements.

Figure 2. Computational time of the algorithms

3. Algorithm complexity estimation
We will consider two algorithms of geodesic distance calculation based on the graph theory. Dijkstra algorithm is a graph algorithm that solves the problem of finding the shortest paths between the given start vertex and all other vertices in an edge weighted graph.

The running time of Dijkstra algorithm depends on the number of edges $|E|$ and the number of vertices $|V|$. The exact time complexity depends on the data structure of the set where the unvisited vertices are stored. If $T_{dk}$ and $T_{em}$ are the complexities of the decrease-key and extract minimum operations in this set then total complexity is $O(|E|T_{dk} + |V|T_{em})$. The simplest implementation of the set of unvisited vertices is array. For this case the time complexity is $O(|E| + |V|^2)$. 
For sparse graphs Dijkstra algorithm can be implemented more efficiently by storing the unvisited vertices in the binary heap. Then algorithm requires $O((|E| + |V|) \log |V|)$ time floating point operations.

The fast marching method is very similar to Dijkstra algorithm. The difference is that the distance values in the fast marching method are calculated using the distance values of more than one adjacent vertex. The possible updated value in the vertex $x$ is

$$d(x) = \min d_{xyz} \ y, z \in \text{Neigh}(x); z \in \text{Neigh}(y)$$

where

$$d_{xyz} = \min \left( t d[y] + (1 - t) d[z] + || t y + (1 - t) z - x || \right), t \in [0, 1].$$

---

Algorithm 1: The fast marching method

1. for $v \in V$
2. $l[v] := \text{Far}$
3. $d[v] := \infty$
4. end
5. $l[s] := \text{Front}$
6. $d[s] := 0$
7. while $F \neq \emptyset$
8. $u := \arg \min_{u \in F} d[u]$
9. $F := F \setminus u$
10. for $v \in V: (v, u) \in E$
11. if $l[v] = \text{Far}$
12. $l[v] := \text{Front}$
13. $F := F \cup v$
14. if $l[v] = \text{Front}$
15. for $w, w' \in V: (w', w), (w, v), (w', v) \in E$
16. $d^* = \min_{t \in [0, 1]} (t \cdot d[w'] + (1 - t) \cdot d[w] + || t w' + (1 - t) w - v ||)$
17. if $d[v] > d^*$
18. $d[v] := d^*$
19. end
20. end
21. end
22. return $d$
As it was mentioned, \( V \) is the set of vertices of the graph, \( E \) is the set of edges of the graph, \( w(u; v) \) is the weight of the edge \( uv \), \( s \) is the vertex, the distances from which are sought, \( d \) is an array: \( d[v] \) is equal to the length of the shortest path from \( s \) to \( v \) after the algorithm has finished. \( l[v] \) is the label of the node \( v \). It may take values \( \text{Far}, \text{Front} \) or \( \text{Computed} \). Also the vertices labeled as \( \text{Front} \) will be stored in binary heap \( F \). The pseudo-code of the algorithm will be as follows (algorithm 1).

The worst case numerical complexity of this algorithm is \( O(|V| \log |V|) \) because all vertices must be tagged \( \text{Computed} \) only once and selection if node with minimum distance from the set of \( \text{Front} \) vertices requires \( O(\log |V|) \) as this set is implemented as a binary heap.

4. Results and discussions

The algorithms are tested on three different joint models. Each model, except the first one, includes several separate parts that have to be joined and each part has its computation nodes. Covering joint surface becomes disconnected as far as it includes areas of several models, each of which can be non-convex. The first considered test case is a test model. It consists of 3124 vertices and has no disconnected areas. The second model is a model of a pylon to skin joint. It is a part of an airframe designed to carry jet engine. The model includes two meshes consisting of 634 and 7554 vertices (the second one was built by remeshing the first one) and has 16 disconnected areas. The last model represents a part of the main aircraft part - wing to fuselage joint. It consists of wing, triform and central wing box and includes two meshes of 1547 vertices each.

![Figure 3](image)

**Figure 3.** The considered models for geodesic distance computation colored by distance values starting of white points. (a) test model, (b) pylon and (c) wing to fuselage joint

To calculate the geodesic distances on the test cases Dijkstra algorithm with binary heap was chosen. The computational time for figure 3(a) is 0.77 ms, for 3(b) is 3.74 ms and for 3(c) is 1.05 ms.

Since it is necessary to calculate geodesic distances over disconnected areas, a corresponding graph will have as many connected components as many disconnected areas are. To overcome the problem,
some edges must be added so that the shortest paths to the leaves will be appropriate. For this the algorithm 2 was proposed.

Algorithm 2: edge adding algorithm

1. **while** \((V, E)\) is not connected
2. \(d_{\text{min}} := \min \text{dist}(v, v'), \ v, v' \in V: (v, v') \notin E\)
3. **for** \(v, v' \in V: (v, v') \notin E\)
4. **if** \(\text{dist}(v, v') \leq k \cdot d_{\text{min}}\)
5. \(E := E \cup (v, v')\)
6. **end**
7. **end**

\(k \geq 1\) is an addition coefficient. Based on the results of experiments, \(k = 2\) was chosen to solve the problem of finding the geodesic distance on figures 3(b) and 3(c), because these figures consist of several disconnected areas.

**Conclusions**

In this work the algorithms for calculating geodesic distance were considered. For the aircraft elements a sufficiently high quality result was achieved. Based on the study, it is possible to conclude that Dijkstra algorithm with binary heap as a priority queue for storing a set of unvisited vertices is the fastest working. The cases with disconnected graphs will be researched in further studies.

**References**

[1] Bi J and Liang J 2007 Multiple Instance Learning of Pulmonary Embolism Detection with Geodesic Distance along Vascular Structure *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*

[2] Ne P, Fishe A and Martin R J 2016 *Rational Mechanics and Analysus* 222 2

[3] Hamza A and Krim H 2006 *Geodesic matching of triangulated surfaces. IEEE Transactions on Image Processing*. 15 2249

[4] Gabriel P and Cohen L D 2009 *Computational Methods in Applied Sciences* 13 29

[5] Petukhova M, Lupuleac S, Shinder Y, Smirnov A, Yakunin S and Bretangol B 2014 Numerical approach for airframe assembly simulation *Journal of Mathematics in Industry*. 4 8.

[6] Lupuleac S, Shinder Y, Petukhova M, Smirnov A, Yakunin S and Bondarenko D 2013 Development of numerical methods for simulation of airframe assembly process. *SAE International Journal of Aerospace* 6 1.

[7] Scarth C et al 2018 *Methods of Applied Mechanics Engineering* 345 1

[8] Bronstein AM, Bronstein MM and Kimmel R 2007 *Journal of Computational Physics* 225 771

[9] Xin S Q, Wang G J 2009 *ACM Trans. Graph.* 28 4

[10] Schwartz W R, Rezende P J and Pedrini H 2015 Faster Approximations of Shortest Geodesic Paths on Polyhedra Through Adaptive Priority Queue. *10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory*