Fast Geodesic Distance Approximation using Mesh Decimation and Front Propagation

Joachim Giard And Benoît Macq
Université Catholique de Louvain,
Communications and Remote Sensing Lab.,
Place du Levant 2, B-1348 Louvain-La-Neuve, Belgium

ABSTRACT
Polygon meshes are collections of vertices, edges and faces defining surfaces in a 3D environment. Computing geometric features on a polygon mesh is of major interest for various applications. Among these features, the geodesic distance is the distance between two vertices following the surface defined by the mesh. In this paper, we propose an algorithm for fast geodesic distance approximation using mesh decimation and front propagation. This algorithm is appropriated when a fast geodesic distances computation is needed and when no fine precision is required.

Keywords: Geodesic Distance, 3D Mesh, Front Propagation, Mesh Decimation, Surface Simplification

1. INTRODUCTION
3D modeling and 3D features extraction are powerful tools for various applications from microbiology to manufacturing. For instance, the travel depth of a molecular surface enables to find pockets where chemical reaction could happen. The point signature, in the domain of face recognition, throws light on points to select to compare two faces (e.g., end of the nose, corner of mouth, ...). In watermarking, it is possible to locate robust feature points by using the curvature tensor, or to weight the local deformation by a roughness score. An efficient and widely used tool to model 3D shapes is the polygon mesh, which is a collection of connected vertices, edges and faces defining an external surface in a 3D environment. Meshes are discretely defined and have no analytical expressions. So, algorithms computing features have to be founded on the only available information: vertices positions and connectivity (inside edges or faces). In this context, the geodesic distance, i.e., distance along the surface, is a base for most surface features. This distance can also be used, as it is, for processes like remeshing, segmentation or animation effects (fire, water, ...).

1.1 Related Work
The geodesic distance is the distance between two points, following the surface of an object. It is, for instance, the distance covered by a vehicle on earth without going underground. In the case of 3D meshes, the geodesic distance is also a basis for discrete differential-geometry operators, such as curvature estimators or integrators. Mitchell et al. introduced an exact "single source, all destination" algorithm, running in $O(n^2 \log n)$. This algorithm is close from the Dijkstra algorithm finding shortest paths in graphs. Surazhsky et al. showed that in practise, it runs in sub-quadratic time. They also proposed an approximation algorithm running, in practise, in $O(n \log n)$ with bounded error. Kimmel and Sethian presented the fast marching algorithm running in $O(n \log n)$. All the methods mentioned here, and in the literature in general, are exact or low-error algorithm. But sometimes, it can be needed to get rapidly a rough approximation of the distance.

Further author information: (Send correspondence to J.G.)
J.G.: E-mail: joachim.giard@uclouvain.be, Telephone: +32 10 47 81 24
1.2 Contribution

In this paper, we propose an algorithm for fast geodesic distance approximation (FGDA) using mesh decimation and front propagation. Mesh decimation is a reduction of the number of vertices, a kind of sub-sampling. Front propagation is a technique to compute geodesic distances from a source vertex to every vertices of the mesh. The idea of the algorithm presented here is to decimate the original mesh, compute the geodesic distance between each pair of vertices on the decimated mesh and store it in an array. The distance between two vertices of the original mesh can be retrieved using connections of the decimated mesh as highways. The general idea is to keep in memory information which can be used for several different geodesics computations. It is illustrated in Figure 1. This technique is appropriated when numerous distances have to be computed or even if the same computations must be done several times. The algorithm is a solution for both low-memory and low-time consummation.

1.3 Overview

This paper is organized as follow. In Section 2, the Fast Geodesic Distance Approximation (FGDA) algorithm is described. In Section 3, error and time analysis are presented. In section 4, some experimental results are exposed. Section 5 contains a discussion about the algorithm and its results.

2. ALGORITHM DESCRIPTION

The algorithm described in this paper, the FGDA, is separated into an initialization phase and a purpose phase. During the initialization phase, common information is computed and stored. This information is used during the purpose phase, each time a geodesic distance is required for a final application.

2.1 Initialization Phase

This phase has to be performed only once and prepares ground information to use during the purpose phase. The initialization phase is illustrated in figure 2.
2.1.1 Mesh Decimation

The original mesh (\(\mathcal{OM}\)) is decimated by removing edges and collapsing end vertices of deleted edges.\(^{19,20}\) The edges are consecutively deleted in such a way that a quadric measure error (linked to the distance to the original surface) is minimized. The deletion stops when the desired decimation rate is reached or when topological constraints cannot be respected anymore. The result of this decimation is the decimated mesh (\(\mathcal{DM}\)). The implementation of the algorithm in the \textit{vtkQuadricDecimation} Class of the Visual ToolKit (VTK \(^\ast\)) is used in this work.

2.1.2 Front Propagation

The geodesic distances between each pair of vertices of \(\mathcal{DM}\) are computed by front propagation, starting from a vertex and making grow the zone of computed distances delimited by an isotropic front. This technique is inspired from the Fast Marching Method introduced by Sethian.\(^{18,21}\) A distance vector \(d_i\) is created for each vertex \(i\) of \(\mathcal{DM}\). First, the distance vector is filled with infinite distances except at the starting vertex \(i\):

\[
d_i(j) = \begin{cases} 
\infty & \text{if } j \neq i \\
0 & \text{if } j = i. 
\end{cases}
\]

Then, the vector is browsed from the smallest distance to the biggest one, updating at each step the distances to vertices connected to the active vertex \(k\):

\[
\forall j \in C(k), \quad \begin{cases} 
\text{if } d_i(j) > d_i(k) + |jk|, \\
\text{then } d_i(j) = d_i(k) + |jk|. 
\end{cases}
\]

where \(C(k)\) is the set of vertices connected to \(k\) and \(|jk|\) is the Euclidean distance between \(j\) and \(k\). At the first step, \(k = i\) because it is the only vertex to have a non-infinite distance. After each step, the vertex \(k\) is labelled as visited and the non-visited vertex with the smallest distance value, considering the updates, becomes the new \(k\). Once all vertices are visited, the front propagation algorithm is performed with \(i + 1\) as starting vertex. Vectors \(d_i\) are constituting the columns of a reference array \(D\) containing the geodesic distances between each pair of vertex of \(\mathcal{DM}\). Finally, the \(N\) closest vertices of \(\mathcal{DM}\) are stored in an \(n \times N\) array, in order to retrieve them faster during the purpose phase. The set of \(N\) closest \(\mathcal{DM}\) vertices from a particular \(\mathcal{OM}\) vertex \(s\) is denoted \(\gamma(s)\).

Generally, \(\mathcal{DM}\) is triangulated, so \(N=3\) is a good choice because vertices of \(\mathcal{OM}\) can be projected on triangular faces of \(\mathcal{DM}\). If the polygons of the mesh are not triangles, they can be triangulated in linear time.\(^{22}\)

2.2 Purpose Phase

Depending on the final application, the use can be different. Generally, distances are calculated either from a source vertex to a target vertex (itinerary), from a source vertex to equidistant vertices (geodesic circles), or from several source vertices to all vertices (Voronoi partition).

In the purpose phase, the Euclidean distances between end vertices (source and target) and close \(\mathcal{DM}\) vertices are computed and added to mutual geodesic distances contained in the array \(D\), constructed during the initialization phase. The expression of \(\hat{\delta}_s(t)\), the estimated geodesic distance between \(s\) and \(t\) is the following:

\[
\hat{\delta}_s(t) = \min_{s' \in \gamma(s), t' \in \gamma(t)} \left( |ss'| + D(s', t') + |tt'| \right).
\]

The sets, \(\gamma\), of \(\mathcal{DM}\) vertices close to \(\mathcal{OM}\) vertices have a size of \(N\), which leads to \(N \times N\) possibilities for finding the minimum.

For vertices in a close neighborhood around the starting vertex, a limited front propagation on \(\mathcal{OM}\) is effectuated to avoid large over-estimations that would be generated by going back and forth to the closest vertex of \(\mathcal{DM}\). This purpose phase is illustrated in figure 3.

\(^\ast\)http://www.vtk.org
Figure 2. The original mesh (left), $OM$, containing $n$ vertices is decimated, resulting in a decimated mesh (center), $DM$, containing $m$ vertices, with $m < n$. The geodesic distance between each pair of vertices of $DM$ is computed using a front propagation algorithm. The distances from a particular vertex $i$ to all the vertices of the mesh is stored in a vector $d_i$, which constitutes the $i^{th}$ column of a reference array $D$ (right). Here, the distance is expressed as a percentage of the maximal distance.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
D(1,1) & D(1,2) & \ldots & D(1,m) \\
D(2,1) & D(2,2) & \ldots & D(2,m) \\
\vdots & \vdots & \ddots & \vdots \\
D(m,1) & D(m,2) & \ldots & D(m,m) \\
\hline
\end{tabular}
\end{table}

Figure 3. The purpose is to approximate the geodesic distance between a source vertex $s$ and a target vertex $t$. The $N$ vertices of $DM$ closest from $s$ (resp. $t$) are candidate to be the approximated source vertex $s'$ (resp. target vertex $t'$). The approximated distance is the shortest one amongst the different possibilities, which are the $N \times N$ combinations of $|ss'| + D(s',t') + |tt'|$. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{horse.png}
\caption{The purpose is to approximate the geodesic distance between a source vertex $s$ and a target vertex $t$. The $N$ vertices of $DM$ closest from $s$ (resp. $t$) are candidate to be the approximated source vertex $s'$ (resp. target vertex $t'$). The approximated distance is the shortest one amongst the different possibilities, which are the $N \times N$ combinations of $|ss'| + D(s',t') + |tt'|$.}
\end{figure}
3. ALGORITHM ANALYSIS

The FGDA aims to approximate quickly geodesic distances on a polygon mesh by computing distances on a decimated mesh, which leads to approximation errors and time gain. Here follow a description of the possible error sources and a time complexity analysis.

3.1 Error Sources

In this section, possible errors due to the decimation are exposed. If the front propagation distance is considered to be the reference, there are two major sources of error. First, an over-estimation because the shortest theoretical path could cross \(DM\) edges, whereas the estimation will follow these edges. Second, an under-estimation because a succession of several segments approximating a curve in \(OM\) could be replaced by a single straight segment after the decimation. The decimation algorithm used for this algorithm removes vertices ordered by a local cost function until the desired threshold of decimation is attained. It conserves the topology of \(OM\) so that there only are local and propagated errors, but no topological errors. Finally, in a close neighbourhood, distances are computed by front propagation which locally gives an exact solution.

3.2 Time Complexity

The time complexity of the FGDA, executed \(p\) times, from a single source vertex to all the vertices is \(O(n \log n) + O(m^2 \log m) + O(p(n \frac{m}{n} \log \frac{m}{n})\)), where \(n\) is the number of vertices in \(OM\), \(m\) is the number of vertices in \(DM\). The first term, \(O(n \log n)\), comes from the quadric-based decimation algorithm. The second term, \(O(m^2 \log m)\), comes from the front propagation algorithm for the computation of the geodesics on \(DM\). The third term, \(O(p(n \frac{m}{n} \log \frac{m}{n})\)), comes from the front propagation in a close neighbourhood around a source vertex containing about \(n/m\) vertices. Finding the shortest path between from \(p\) to \(n\) vertices testing the \(N\times N\) possibilities has a complexity of \(O(pm) \in O(p \frac{m^2}{n} \log \frac{m}{n})\). Since distances between vertices of \(DM\) are stored in an array of size \(m^2\), \(m\) is set to a constant, in practice, to avoid memory lacks. In this case the complexity of the FGDA is \(O(m \log m)\), the same than for the front propagation algorithm alone. However, in practice, coefficients are smaller for the FGDA because only local front propagations are needed. This is true only if the computed distances are long, if \(p\) is large or if \(n\) is large. Otherwise the third term, \(O(p \frac{m^2}{n} \log \frac{m}{n})\), becomes comparable to both other terms, \(O(n \log n)\) and \(O(m^2 \log m)\), and the front propagation alone becomes less time consuming.

4. EXPERIMENTAL RESULTS

To illustrate the time saving and to estimate the errors, some comparisons between the FGDA and the front propagation algorithm are presented here. Tests were performed on 3 different meshes: head (11703 vertices), horse (48485 vertices) and molecule (435894 vertices). Distances were computed from one source vertex, all vertices of the mesh being targets. The process were repeated 1, 10, 100, 1000 and \(n\) times with different starting vertices, for each mesh and both methods (\(n\) is the number of vertices in the considered mesh), \(m\), the number of vertices in \(DM\) was set to 500. For the front propagation algorithm, it boils down to repeating the whole unique-source and multiple-target algorithm for every different starting vertex. For the FGDA, only the purpose phase is repeated. Time plots and screenshots of the results are depicted in figure 4. Spent time using front propagation only increases linearly with the number of starting vertices because the algorithm is simply repeated. Spent time using FGDA increased slower when the number of different starting vertices is smaller due to the fixed cost of the initialization phase.

A mean error score was also computed during the tests. The reference was the front propagation distance, evaluating the error generated by the decimation phase. The mean error \(e_i\) for the starting vertex \(i\) is

\[
e_i = \frac{1}{n} \sum_{j=1}^{n} \frac{|\delta_i(j) - \tilde{\delta}_i(j)|}{n \cdot \delta_{i,max}},
\]

where \(\delta_i(j)\) is the geodesic distance between \(i\) and \(j\) using the front propagation only, \(\tilde{\delta}_i(j)\) is the geodesic distance using the FGDA and \(\delta_{i,max}\) is the maximal value of \(\{\delta_i(j) | j \in [1, n]\}\). The mean error was 3.2% for the head, 1.65% for the horse and 2.33% for the molecule.

The whole algorithm was implemented in C++ with VTK. All the tests were performed on a computer with an Intel Core 2 CPU @ 2.00GHz, and with 2 Gb RAM.
Figure 4. Examples of algorithm results (left) and corresponding execution times (right) for the head (top) (about $10^4$ vertices), the horse (middle) (about $5.10^4$ vertices) and the molecule (bottom) (about $5.10^5$ vertices). The times plotted are those of the front propagation algorithm (gray circles) and the FGDA (black squares) for the computation of distances to all the vertices of the mesh from a unique source. On the x-coordinate, the number of different source vertices for which distances were computed. Scales are logarithmic. Times are expressed in seconds.
5. DISCUSSION

This algorithm is appropriated when a fast geodesic distance approximation is needed from several different sources, at a long distance or for large meshes. As shown in section 4, for the example of the unique-source and multiple-target, the fixed time of the initialization phase is longer than the complete computation using the front propagation algorithm if the mesh is too small (see figure 4, for the head and the horse). This disadvantage appears for combination of these factors: short distances computation, few different source/target vertices and small meshes.

The time for $5 \cdot 10^5$ vertices is coherent with the one announced by Surazhsky17 (one minute). The mean error was calculated in reference with the front propagation algorithm used for the tests. It worth noting that any geodesic distance algorithm can be used in this method. So, the error method just takes into account the difference induced by the decimation.

6. CONCLUSION

To sum up, we propose an algorithm approximating geodesic distances on a polygon mesh and reducing considerably the execution time. It is based on a front propagation algorithm applied on a decimated mesh. In the future, this algorithm would be developed to create other tools such as differential or integral operators on meshes.14

ACKNOWLEDGMENTS

Joachim Giard is funded by Nanotic/Tsarine, a project of the Région Wallonne of Belgium (convention number: 516248).

REFERENCES


