Terrain Sensitive Routing

Context Survey

George Pestell (200007413)

October 12th 2024

Contents

1 Route Planners

Typical route planners aim to allow users to specify two points on the earth's surface as latitude and longitude coordinates, and then determine and display a suggested route across earth's surface between those two points. The most popular consumer applications such as Google Maps, Apple Maps, and Open Street Maps rely on a graph where edges between nodes represent road/path sections, and nodes represent intersections where someone could change the path they are on. Basic implementations weight the cost of edges based on the distance between the two points represented by the nodes.

Modern advances in specialist applications such as Google's Terrain View, and OS-Maps consider elevation of the paths and roads, offering walking difficulty scores and ranking for hikers. Services such as Ways can consider both permanent , and transient data through community sourcing. This includes speed-cameras, road-works, congestion, and more. However, this data is still limited to improving the suggestions of roads and paths between two points.

2 DEMs

The topology of the terrain is a crucial factor in determining the speed and feasibility of routes. Digital elevation models (DEMs) are raster data structures representing this data. It contains a set of points in a given space, along with their elevation. Regular DEMs have elevation readings evenly spaced along a grid. [1]

3 Graph Representation / DTMs

Typical routing algorithms such as Dijkstra's algorithm and A* require a graph structure where nodes represent points on a 2D surface, with connecting edges representing the possible connections between them.

One method to construct this graph is to first construct a digital terrain model (DTM), which is a mesh from a DEM, so the elevation data is represented in the topology of the mesh [1]. Considering the center of each face a node, adjacent faces can be connected with edges.

There are various methods for creating meshes which will be discussed here. A popular approach is to create a triangle irregular network (TIN). TINs use a triangulation algorithm such as Delaunay triangulation to create a mesh where each face is a triangle [2]. They can very accurately represent different topological features, and so are used widely in 3D modelling [1].

Another method for mesh derivation is the square-based mesh-derivation. This creates a mesh consisting of square/rectangle faces. This approach is particularly suited to regular grid DEMs, because the data is equally spaced, and so the resulting mesh consists of regularly sized square faces.

Alternative mesh shapes can also be used for mesh derivation, UBER's H2 positioning library shows the potential to create a mesh from regular hexagons [3]. Like with regular square meshes, this has the benefit that each face is the same size, but has the added benefit that the distance between hexagons is uniform in all directions, unlike square meshes which have a different distance for adjacent and diagonal faces, and unlike TINs which can have three different distances: one for adjacent faces, and two different distances for the diagonal faces.

The mesh that will be generated from the DEM data will be 2.5D, as each point on the DEM grid gets a single elevation value.

4 TIN Derivation

There are an infinite number of way to use triangles to represent a surface, and so an infinite number of TINs that would represent the same DEM. For regular DEMs, the simplest way to triangulate points is to connect each point as a grid, and then add a single edge in each square to create evenly sized triangles. For irregular DEMs, or those with missing points, a more sophisticated algorithm is required: Delaunay triangulation.

Delaunay triangulation is the process of taking a set of 2D points and forming a triangular mesh. It aims to limit very thin and long triangles, in favour of more equilateral, evenly sized triangles [4].

2.5D Delaunay triangulation generates these meshes by conducting a 2D Delaunay triangulation on the X and Y positions of the points given, and then applies the Z values to each of the points afterwards [5]. This method is popular for TIN mesh derivation as the DEM data is given as 2.5D data.

3D Delaunay Triangulation is an adaptation to the basic algorithm, creating 3D structures from 3D tetrahedra instead of 2D triangles [6]. The is especially useful for accurate 3D modelling where rich data is available. Unfortunately, it is often not required to create terrain representations as the DEM data is often limited to 2.5D.

Constrained Delaunay Triangulation extends the algorithm through the addition of constrained edges. Constrained edges cannot be removed or altered in the triangulation algorithm, allowing edges such as the boundaries of features (e.g. sides of rivers, buildings, etc.) to be maintained [7, 5].

4.1 C++ Libraries

A substantial amount of work has been put into creating libraries which extract the complexities of creating an efficient Delaunay implementation. The following is an exploration into C/C++ libraries designed for Delaunay triangulation.

- **CGAL** [8] - This is a complete suite for computational geometry. It is able to do 2D, 2.5D, and 3D Delaunay triangulations, and also offers a multitude of simplification and smoothing tools. In addition to constrained Delaunay triangulation.

A drawback to this library is the amount of tools included can make it difficult to determine the correct one to use, and the amount of configuration options available can be difficult to understand. This is also a benefit, as it reduces the requirements for additional dependencies.

However, the benefits are that the documentation is extensive, and covers many examples. In addition, the CGAL library itself is licensed using the GPL v3+ open source licence giving the freedom to use the software for any purposes.

- **tin-terrain** [9] - This open-source tool has been specifically created for the generation of terrain TINs. It is no longer in active development.

A drawback to this library is the lack of additional tools for TIN processing. It's main benefit is that it has a permissive GPL licence, giving access to use it for any purposes.

- **Fade25D** [10] - This set of tools is especially aimed at Delaunay triangulation of DEMs. It offers simple abstractions for simplifying and smoothing TINs, as well as the ability to add constraints through "breakpoints".

The benefits to using this library is the lack of explicit configuration required. The amount of tools available specifically for terrain TIN derivation makes it quick and easy to integrate. The documentation is also complete. However, it doesn't contain a permissible open source licence. A student licence is available, enabling restricted use of it's tools for research and education, but one of the restrictions imposed on this licence is a limit on the number of vertices and faces the TIN contains.

5 GIS Software

The amount of data that is reasonable to represent on a map is immense. Geographic Information System (GIS) software is a category of technologies used to create, visualize, and analyse this data. Data is represented by layers which can include raster or vector data. This organization allows users to easily manage and work with their data [11].

6 Coordinate Systems

There are two primary categories of coordinate systems that can be used to represent points on the earths surface [12].

- **Global Coordinate Systems**
- **Projected Coordinate Systems**

A global coordinate systems (GCS) defines where data is located on the earth's surface. The World Geodetic System from 1984 (WGS84) uses latitude/longitude values referencing the relative angle away from the equator (0 to 90 degrees of latitude) and from the prime meridian (0 to 180 degrees of longitude). A projected coordinate system (PCS) defines how to flatten the positions of the 3D globe onto a 2D surface such as a map. Many projections exist, each with their own characteristics.

One of the drawbacks of a PCS is distortion: the pseudo Plate Carrée projection distorts all angles, shapes, and distances to some degree. This means that many PCS are unsuitable for distance calculations or other spatial analysis tasks. However, WGS84 UTM (Universal Transverse Mercator) is a projected coordinate scheme based on WGS84 aiming to accurate project latitude longitude coordinates to regular x and y coordinates where 1 x,y value is equivalent to 1 meter . This makes it much more suitable for distance calculations. The only exception to this uniformity is on the southwest coast of Norway over open water, and around Svalbard.

7 Shortest Path Algorithms

The goal of shortest path algorithms is to calculate the “shortest” path along a graph data structure from a source node, across edges connecting nodes, to a target node. Famous examples include Dijkstra’s algorithm and it’s successor A*. These algorithms use cost functions to weight the edges of the graph, determining how “good” it is to travel across that edge. This “goodness” is defined by the specific cost function used. At its most basic, the cost function can reference the edge’s distance in real life. However, the cost function can utilize any quantifiable characteristic, and model complex relationships between different characteristics.

These algorithms begin from the source node (or alternatively the target node in some implementations) and add all of the adjacent nodes to a search queue. Then, a current and total cost score is calculated for that node - current being the result of the cost function, and total being the sum of the previous costs along the route. It’s neighbours not already on the route are then pushed to the queue.

The original Dijkstra’s algorithm [13] had no way to rank potential next steps based on their likely goodness, and so had to complete a full search of all nodes before finding an optimal path making it inefficient. However, it is guaranteed to find the best path.

A*’s main contribution was the introduction of heuristics - computationally easy characteristics which estimate the cost. With an admissible heuristic (i.e. one that never over-estimates the true cost), A is guaranteed to find the optimal path first, saving much unnecessary computation.

Whilst A* is effective in static environments, where features are known and do not change, it fails to adapt to changing information after it has begun. Dynamic A* (D*) was developed [14] to account for this possibility - allowing the path to be updated without recalculating from scratch where required. This lead to Field D* [15], which removed the restriction of following exactly between nodes and edges. Instead, the environment is represented as a continuous field, allowing cost functions to be defined for traversing across a face of the graph.

Field D* is used extensively in applications such as Robotics and real-time navigation. In Robotics in particular, there is a requirement to prioritize terrain the robot is capable of following. Specific research has used Field D* for application over TINs [6].

8 W. Evans’ Terrain Sensitive Routing

Previous work on creating a terrain sensitive routing application was done by Will Evans for their dissertation project [16]. Much of their initial exploration into the topic serves as the groundwork for this one and a summary of their work’s findings will be given here.

Their routing algorithm was a basic implementation of the Dijkstra’s algorithm as they identified no admissible heuristic. They represented the terrain using a regular square mesh. The characteristics considered in the cost functions were limited to: binary features such as deep water or paths, which are guaranteed to be untraversable/traversable; and topological elevation/gradient data which affects how traversable an edge is based on predefined cost curves defining the effect of certain gradients on walking speed.

Evans' approach was successful in generating routes between two points. Unfortunately, no attempt was made to evaluate the quality of the routes generated.

Many areas for potential improvements to improve the performance and overall design of their code. Given that Dijkstra's original algorithm must complete a full search before the optimal path is determined, implementation of heuristics would greatly improve it's efficiency.

Additionally, Evans identified improvements to the cost functions to improve the quality of routes generated. First, dynamically adjusting the impact of certain features based on the form of transportation (e.g. cycling over some terrain is faster/slower than when on-foot). They identified weather as a potentially important factor for determining which routes are fast/safe. Additionally, instead of a simple effect curve for the impact of gradient on walking speed, maintaining a gradient history may model a users level of cardio-vascular stamina remaining.

Another major improvement identified was in the mesh itself. Their approach used a regular square grid, which resulted in unnecessary computation for large areas with a consistent gradient. Evans suggests a dynamic mesh, similar to TINs, could be used to add more nodes where there is significant gradient change, and reduce the number of nodes where there is not.

Finally, they suggest natively supporting vector formats for paths and boundaries, reducing the likelihood of a lack of resolution causing holes in the feature boundaries.

References

[1] Marc van Kreveld. "Digital elevation models and TIN algorithms". In: *Algorithmic Foundations of Geographic Information Systems*. Ed. by Marc van Kreveld et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 37–78. ISBN: 978-3-540-69653-7. DOI: 10.1007/3-540-63818-0_3. URL: https://doi.org/10.1007/3-540-63818-0_3.

[2] Robert J. Fowler and James J. Little. "Automatic extraction of Irregular Network digital terrain models". In: *SIGGRAPH Comput. Graph.* 13.2 (Aug. 1979), pp. 199–207. ISSN: 0097-8930. DOI: 10.1145/965103.807444. URL: https://doi.org/10.1145/965103.807444.

[3] Uber. *Uber H3*. Last accessed: 12/10/2024. URL: https://h3geo.org/docs/.

[4] Der-Tsai Lee and Bruce J Schachter. "Two algorithms for constructing a Delaunay triangulation". In: *International Journal of Computer & Information Sciences* 9.3 (1980), pp. 219–242.

[5] L Rognant et al. "The Delaunay constrained triangulation: the Delaunay stable algorithms". In: *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*. IEEE. 1999, pp. 147–152.

[6] Simon Perkins et al. “Field D* path-finding on weighted triangulated and tetrahedral meshes”. In: *Autonomous Agents and Multi-Agent Systems* 26.3 (2013), pp. 354–388. ISSN: 1573-7454. DOI: 10.1007/s10458-012-9195-8. URL: https://doi.org/10.1007/s10458-012-9195-8.

[7] L. P. Chew. “Constrained Delaunay triangulations”. In: *Proceedings of the Third Annual Symposium on Computational Geometry*. SCG ’87. Waterloo, Ontario, Canada: Association for Computing Machinery, 1987, pp. 215–222. ISBN: 0897912314. DOI: 10.1145/41958.41981. URL: https://doi.org/10.1145/41958.41981.

[8] The CGAL Project. *CGAL User and Reference Manual*. 6.0. Last accessed: 12/10/2024. CGAL Editorial Board, 2024. URL: https://doc.cgal.org/6.0/Manual/packages.html.

[9] heremaps. *tin-terrain*. Last accessed: 12/10/2024. Github. URL: https://github.com/heremaps/tin-terrain.

[10] Geom Software. *Fade25D*. Last accessed: 12/10/2024. URL: https://www.geom.at/fade25d/html/.

[11] David J Maguire. “An overview and definition of GIS”. In: *Geographical information systems: Principles and applications* 1.1 (1991), pp. 9–20.

[12] E Lynn Usery, Michael P Finn, and Clifford J Mugnier. “Coordinate systems and map projections”. In: *M. Madden,(Ed-in-Chief), The Manual of Geographic Information Systems. American Society for Photogrammetry and Remote Sensing, Bethesda, Maryland* (2009), pp. 87–112.

[13] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Edsger Wybe Dijkstra: his life, work, and legacy*. 2022, pp. 287–290.

[14] Maxim Likhachev et al. “Anytime dynamic A*: An anytime, replanning algorithm.” In: *ICAPS*. Vol. 5. 2005, pp. 262–271.

[15] Dave Ferguson and Anthony Stentz. “Field D*: An interpolation-based path planner and replanner”. In: *Robotics Research: Results of the 12th International Symposium ISRR*. Springer. 2007, pp. 239–253.

[16] Will Evans. “Terrain Sensitive Routing”. BSc Dissertation Project. The University of St. Andrews, 2023.

Development Plan

	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	(XMAS) Week 15	(NYE) Week 16	Week 17	Week 18	Week 19
Context Survey	▓	▓															
Ethics Application	▓	▓															
Create Mesh			▓	▒													
Basic Feature Tagging			▒	▓													
Initial Routing Algorithm				▒	▓												
Initial Pre-Set					▓												
Complex Features						▓	▓										
Pre-Set Creation							░	▓									▒
More Features									░	░							
Visualisation				▒	▒					▓	▓						
Optimisation												▓	░				
Evaluation													▓	▓	▓	▒	▒
Report: Design				░	░	░	▒	▒	▓	▒					░	░	░
Report: Implementation								▒	▒	▓	░	░					
Report: Testing			░	░	░	░	░	░	░	░	░	░	░	░			
Report: Further Improvements															▓	▓	
Report: Conclusion															▓	▓	░
Misc Writing													▓	▓			░

Priority	HIGH	MED	LOW	NON
	▓	▒	░	