Mesh Extraction from a Regular Grid Structure Using Adjacency Matrix

David Tully¹, Abdenmour El Rhalibi^{1,2(✉)}, Zhigeng Pan²,
Christopher Carter¹, and Sud Sudirman¹

¹ School of Computing and Mathematical Sciences, Liverpool John Moores University, Liverpool, UK

dtully@2008.ljmu.ac.uk, {A.ElRhalibi, c.j.carter,
S.Sudirman}@ljmu.ac.uk

² Digital Media and HCI Research Center, Hangzhou Normal University, Hangzhou, People's Republic of China
zgpan@hznu.edu.cn

Abstract. Crisis management is a modern phenomenon brought about by natural disasters and acts of terrorism. Building a modern crisis management response program needs a multi-disciplinary architecture and accurate, up-to-date, real-world data. The creation of virtual environments depicting critical infrastructure buildings and conduits between these highly interconnected man-made structures is a complex procedure. The crossover between games technology and use of real-world map data for real-world simulations is becoming more common with the advancements of computer hardware and software, and the accuracy of real-world map data. However, there are many problems with using real-world map data for simulation due to the large potential of missing and error prone data involved in this big data. Within this work we use three types of data sets; Ordnance Survey data, LiDAR data, and OpenStreetMap data to provide accurate map and 3D environment information for crisis management systems. Combining these large data-sets can reduce errors and retrieve missing data for use within a modern game engine for visualization analysis. We propose a novel technique for data extraction using adjacency matrices for custom model generation corresponding to real-world structures such as landscapes, buildings, road systems, area boundaries, or a combination of these at different resolutions.

Keywords: Crisis management · Adjacency matrix · Computer games · Visualization

1 Introduction

For the creation of a modern crisis management tool to aid crisis managers, as proposed by [1], needs accurate up-to-date data and advanced visualization techniques for the interacting and displaying of large complex layering of multiple data-sets to the crisis manager. A modern visualization system depicting up-to-date, accurate information is needed when dealing with large interconnected real-world scenes. For example, the first responders of a crisis event would be helped if they had a 3D visualization of the

infrastructures surrounding the affected area such as buildings and especially the conduits connecting a highly dense area [2–5]. Having a system which can combine multiple data-sets and produce accurate terraformed scenes for crisis managers to interact and understand unseen scenarios and landscapes is critical for making the best possible plans to solve the crisis. The use of a modern computer game engine can provide high level API's for the generation of large, highly realistic visualized real-world scenes. The use of this hardware and software is relatively cheap in current time, and destined to become even cheaper. In contrast, the availability of accurate real-world data is often extremely expensive and difficult to obtain. Combining games technology and real-world large data-sets is a difficult task in itself, due to the large differences between the data-sets and the choices of hardware and software available. Another issue is the high level of training needed to use this hardware and software. Generating large realistic scene within virtual environments needs custom algorithms to produce scenes which can run at a minimum of 30 frames-per-second. Triangulation techniques such as Delaunay triangulation has been used to reduce the potential vertex and triangle generation in models in computer games, thus reducing calculations. Using adjacency matrices and uniform 2D grids of accurate data, we can remove the use of complex triangulation algorithms to reduce highly dense shaped objects.

Within this paper we discuss procedures needed to produce accurate complete data-sets for the extraction of custom model meshes obtained by the use of adjacency matrices with use with uniform 2D grids. This technique combines 3 data-sets; Ordnance Survey data UK, LiDAR data, and OpenStreetMap.Org data.

The structure of the paper is as follows: In Sect. 2, we discuss the concept of crisis management. In Sect. 3 we present some of the data we use in our approach based on Ordnance Survey and LiDAR data. In Sect. 4, we introduce the map data we use based on OpenStreetMap data. In Sect. 5, we discuss the Interpolation and combination of data-sets. In Sect. 6, we introduce our technique based on artefact extraction using adjacency matrices. In Sect. 7, we present the effect and accuracy of different interpolation techniques of on our extraction approach when changing the resolution of the map data. In section we conclude the paper.

2 Crisis Management

Critical infrastructures surround our daily lives; be it transportation, financial transactions, water, gas, electricity, the internet. They are the backbone to our way of life. If upset is caused to one of these highly interconnected infrastructures, it can have unpredictable disastrous consequences which can cascade through all connected infrastructures. A train carrying hazardous materials crashed in 2001 in the USA [6]. The cascading effects consisted of multiple fires, a burst water pipe which flooded local streets which in turn short circuited local electricity supplies, preventing the use of local communication systems such as telephones. 1200 homes were affected for many days.

Having a modern crisis management tool capable of displaying connections and visualize highly accurate real world urban environments with minimal errors within data-sets can provide crisis managers with specific in sites into unfamiliar territory. Our techniques proposed can allow the extraction of specific map data which can then be

used in a layered fashion allowing the user to see; landscapes, buildings, road systems, area boundaries, or a combination of these at different resolutions.

3 Ordnance Survey and LiDAR Data

Ordnance survey (OS) data is acquired through the UK government web portals. The data in use by our procedures represents height points on the terrain of the UK. Each point is contained in text file. Each file covers 10 km² containing 200² points which are 50 m apart. This large distance in sampling data provides large potential for increased errors rates and missing artefacts from scenes. Light detection and ranging (LiDAR) tries to improve this by sampling height data obtained by projecting light beams from a low flying plane and records the time the light beam takes to bounce from the ground to the plane. This creates height points accurate to within 5 to 15 cm margin of error. LiDAR data is provided in two formats; digital-terrain-model and digital-surface- model. LiDAR has trouble reflecting off water and penetrating bio-material such as trees and shrubberies which introduces error. Another issue with LiDAR is the expense of obtaining accurate data and the large amount of data produced. An estimated 113.2 terabytes of storage needed to store a complete set of 4 spatial resolutions of LiDAR maps; 2 m, 1 m, 0.5 m, and 0.25 m sampling points, covering the UK. Interpolation techniques can be utilized to up sample data to generate missing data points but will introduce small amounts of error. An added benefit of this will be lower storage facilities will be needed. The maps produced can be used by a modern game engine to create accurate terraformed landscapes. Within this work we propose a novel mesh extraction technique for the extraction of artefacts regular complete data sets. The large data-sets of real-world data is a difficult challenge for a multiple of reasons, be it data storage; LiDAR coverage for the UK in 4 spatial resolutions; 2 m, 1 m, 0.5 m 0.25 m point sampling. The data we use **has been** donated by Geomatics-group¹. They process the raw data with multiple techniques to produce two separate models; the digital-terrain-model (DTM) and the digital-surface-model (DSM). A DTM is the terrain of a map not including structures such buildings, trees, pylons etc. while a DSM contains all assets of an environment. The DSM maps have a distinct problem of capturing movable objects such as birds which introduce high spike of error within a scene. This creates visually inaccurate scenes but also, if spatial analysis is performed on the DSM map, this can produce cascading errors within procedures.

4 OpenStreetMap Data

OpenStreetMap² (OSM) is a worldwide open source platform for mapping local areas. The user generated data is often more up-to-date than the likes of Google maps³ which can be 4 years out of date for rural areas. This is a problem for generating accurate

¹ <https://www.geomatics-group.co.uk>.

² <https://www.openstreetmap.org>.

³ <https://www.google.co.uk/>.

virtual scenes. OSM provides object types which specify the longitude and latitude locations of points on the Earth's surface. OSM contains 4 main types of object: *node* which represents a single location; *way* which represent an open or closed list of *nodes* which is used for building boundaries and road systems; *relation* which adds information between a multiple of *ways* or *nodes*; and *tag* which is a key-value object which states what the *node* or *way* represents. Using this data we have created large visualizations of real-world cities. Figure 1 shows our procedurally generated city of Manchester in UK covering 1,277 km² area.

Fig. 1. Left OpenStreetMap.Org screen shot of part of Manchester UK. Right procedurally generated navigable virtual scene of the complete Manchester OSM file 1,277 km²

5 Interpolation and Combination of Data-Sets

To generate complete maps used for mesh extraction, complete data sets are needed. In this section we will cover problems which are common within these data-sets and procedures needed to reduce or remove these errors. For building mesh extraction we will be using the DSM maps of the LiDAR data-set. As stated this map type can have multiple errors within it, from flocks of birds producing large spikes of error data, to missing sections of data due to the light beams not being able to bounce off water or produce accurate readings through bio-material. Combining data from the DTM with the DSM maps can fill in missing data but if data is missing from the DTM map then data can be retrieved from a lower resolution map. The OS maps are the lowest resolution maps available. Interpolation must be done to produce a resolution according to the DTM/DSM map we are creating it for, in this case it will be OS map at 50 m resolution to LiDAR 2 m resolution. The use of multiple interpolation techniques to interpolate to a higher resolution is discussed in a later section.

Working with DSM maps which have missing data spanning a buildings edge is a special case for interpolation. Using OSM building boundary data, the edges of the buildings can be determined within the LiDAR maps. Using these boundaries we can interpolate or duplicate data contained within the boundary, or use a smoothing technique commonly found in image processing techniques [7]. If no data is within a boundary, because either data is missing from the LiDAR data or the LiDAR data is out of date, then this is process which is needed to be completed for future work.

It is needed to be stated that OSM provides longitude and latitude locations projected in EPSG:3857⁴ schema. Working with GIS systems, this projection schema is not a problem, but combining it with the ordnance survey national grid reference scheme uses by OS and LiDAR data, is an issue due to their projection differences. However, tools are available to convert from one to another⁵.

6 Artefact Extraction Using Adjacency Matrices

The DSM LiDAR maps contain all assets within an environment which are captured at data acquisition time, be it pylons, cars, birds, buildings, and other artefacts. A DTM map visualized within a modern game engine at the highest resolution of 0.25 m covering 1 km² contains 16 million vertices. The need of an extraction process is required to extract specific artefacts from a DSM to then use to place it on top of the DTM. If the DTM is visualized at the same resolution, this will use 16 million vertices, and plus the vertices added from the extraction process. The user can state at what resolution the DTM needs to be viewed at. There are many level of detail algorithms for large polygon reduction rates for use with large open scenes [8–10]. These algorithms reduce the polygon count but for intricate artefact structures such as churches, high detail is needed for realism and also spatial boundary analysis. Extracting mesh data for specific object, we can remove potential erroneous movable objects. Categorized map layers can be placed on top of the DTM to create a more accurate representation of real-world scenes which reduce model imperfections.

After maps have been preprocessed to check completeness, artefact extraction can take place. The following statements will focus on building extraction, but can be used for any boundary positions. Using OSM data to map the boundaries of buildings, we can use adjacency matrices to extract specific data points from the regular 2D grid of the LiDAR map. Adjacency matrices are multi-dimensional array stating the adjacency connections of objects⁶.

The algorithm proceeds as follows:

Convert the building or artefact boundary points we wish to extract from OSM projection to OS national grid reference schema.

Create a 2D array of same width and height of the LiDAR map which will contain 0 s and 1 s populated by the next process.

Loop through all the points of the LiDAR map and if the point is contained within the boundary if the converted OSM artefact, then within the empty 2D array at the same index, insert a 1, else insert a 0. This will create a new array containing only 0 s, and 1 s containing only the LiDAR data points obtaining to that artefact.

The next process is to convert those points to a model mesh. This is done by looping through the 2D array of 0 s and 1 s.

⁴ <http://wiki.openstreetmap.org/wiki/EPSG:3857>.

⁵ <http://www.movable-type.co.uk/scripts/latlong-gridref.html>.

⁶ <http://mathworld.wolfram.com/AdjacencyMatrix.html>.

For simplicity and explanation in accordance with the diagrams, we will depict the starting index of the 2D array as the top-left point of the map. This is how the data points are formulated with the LiDAR, which is confusing due to OS national grid reference starts at the bottom left of maps.

Looping through, the algorithm will check if each point is either a 0 or 1. If it is 1, then the adjacency matrices will be generated. Two matrices will be generated. The first matrix will contain the points to the left, to the bottom, and diagonally bottom left. The second will contain the right, bottom right, and bottom surrounding points. Figure 3 shows the adjacency matrices generated.

To generate triangle within the vertex array used within a game engine or model for a CAD drawing, we need to create a technique which can loop through points and decide to generate a triangle in a certain position and orientation as to always extract complete sets without missing potential points. This is a complex task. For example, given 4 vertices to make a polygon there are 2 ways to orientate the triangle. For the procedural extraction of points the triangles need to be orientated and positioned in a specific order as to eliminate future complications. Figure 2 shows the triangle orientations needed. The left side shows a purple triangle (top left) generated if the first matrix contains points top left, top right, bottom right. If only the bottom left, bottom right and top right then the yellow triangle will be generated. If all points are within the matrix then the purple triangle will be generated. See the right side of Fig. 2 for the configurations needed to generate the other triangles.

Fig. 2. Triangle orientation and placement

The matrices contain 0 s and 1 s which are converted to a binary sequence. The sequence will be ordered as stated in Fig. 3. Within the figure the binary sequence will be 0010 which equals 2.

Figure 4 shows the combinations of binary sequence values for matrices generated for the left and right matrices. The algorithm runs the value through an if-else statement to generate the corresponding triangles. For example, if the left matrix is equal to 11 or

Fig. 3. Two adjacency matrices generated from specific point. Left matrix contains points to the left, bottom left, and bottom of the point. Right matrix contains points to the right, bottom right and bottom of the point.

15 then generate the purple triangle; else if the value is equal to 14 generate the yellow triangle. For the right matrix, if the value is equal to 13 or 15, generate the orange triangle, else if the value is equal to 7 then generate the green triangle.

Fig. 4. Binary values for the left and right matrices (Color figure online).

Figure 5 shows the end product after the process has completed. The left image shows each point being processed and each triangle generated. The right image shows the complete polygonal mesh after the process has finished.

Fig. 5. Example of a complete map after matrix analysis and triangle generation. Left depicts the individual triangles after each process. Right is the end model mesh.

Fig. 6. Procedural city generation of part of New York Manhattan containing buildings, roads, and rail networks

Within each point being checked, if a triangle is generated, it needs to be added to a vertex array for use with creating the model, and also added to an index array. DirectX can rasterize triangle strips by adding each triangle to an array, but this is a huge performance cost, thus the addition of an index array can be used to offset the performance cost. For each point of each triangle, we can check whether it has been added to the vertex array already; if it has been added, then we extract the index value of the vertex in the array, and add that index value to the index array.

Figure 6 depicts a procedural city generation of part of New York Manhattan containing buildings, roads, and rail networks using map data and related OSM, and LiDAR data.

7 Interpolation Experiment Results

Interpolating from an OS map at 50 m resolution to 2 m resolution to match the LiDAR DTM resolution accurately is dependent on the interpolation technique used. The experiment consisted of an OS map at 50 m resolution and a benchmark LiDAR DTM map covering the same 1 km^2 location within the UK. The OS map covers 10 km^2 at 200^2 points, so a subset needs to be extracted to match the same area and coverage of the LiDAR DTM map which is 1 km^2 at 20^2 points. In contrast the LiDAR DTM 2 m resolution map is 1 km^2 at 500^2 points. After this preprocessing has taken place the OS subset map can be interpolated. The comparison will be to determine which interpolation technique introduces the most amount of error when interpolating. There are a multiple of reasons why we need to compare interpolation techniques. Using interpolation to interpolate from a low resolution map to a higher resolution means that less storage is needed. For example, interpolating from 20^2 to 500^2 points is a total of 400 points to 250000 points respectively; which is equivalent to 625 50 m resolution 1 km^2 maps. The calculations of the different interpolation techniques need to be considered. When speed is of the essence, interpolating huge data sets to from a low resolution to a high resolution will take time. When time is not an

Fig. 7. Blue line represent ease-out, orange line represents ease-in, and the red line represents ease-in-out (color figure online)

issue and visual accuracy is, then more complex computational interpolation techniques can be used.

The interpolation techniques we used for our experiments are: Linear, circular, cubic, sinusoidal, quadratic, catmull-rom, exponential, quartic, and quantic. The interpolation techniques were obtained from a tweening library.⁷ A tweening library is a term used for calculating the values in-between two other values, and is often used for animations. The library contains functions for easing-in, easing-out, and ease-in-out, see Fig. 7.

The tweening library does not contain an API for the Catmull-Rom interpolation function. Catmull-Rom interpolation needs additional point data either side of the values being interpolated to generate the tangent angle for entering and exiting the interpolated values. For this, additional tangent points are needed which are extracted from the surrounding maps of the map being interpolated. This experiment has only worked with a map which has complete surrounding maps. To work with missing data values, multiple pre-processing techniques will have to be employed but this is left for future work and is not relevant for this type of experiment.

Table 1. Error rates after interpolation technique is applied to 50 m resolutions at 20² point map to generate a 2 m resolution at 500² point map

Interpolator	Interpolated point map mean average height	Benchmark point map mean average height	Error in metres
Quintic	37.73528	37.79643	0.06115
Quartic	37.73481	37.79643	0.06162
Exponential	37.73467	37.79643	0.06176
Catmull-Rom	37.73452	37.79643	0.06191
Quadratic	37.7342	37.79643	0.06223
Sinusoidal	37.73419	37.79643	0.06224
Cubic	37.73419	37.79643	0.06224
Circular	37.73406	37.79643	0.06237
Linear	37.73349	37.79643	0.06294

⁷ <http://xnatweener.codeplex.com/>.

Fig. 8. Error in meters bar-chart for interpolation experiment

The error rates of the interpolation techniques are generated by first interpolating the 50 m 20² point maps to 2 m 500² point maps. The values are summed together and divided to find the mean average. This average is compared to the complete benchmark LiDAR DTM map which has had the same mean average calculated. The results are shown in Table 1 and Fig. 8. As Table 1 shows, the potential error for all interpolation techniques is 6 cm's. Linear interpolation introduces the most amount of error, where the quantic interpolation function produces the least amount of error, but the computations between the interpolation techniques are considerably different.

8 Conclusion

Within this paper we have presented a technique for a novel mesh extraction technique by iterating through regular complete 2D grids of real-world height map data to be visualized and utilized within a crisis management support system. The extracted meshes will be used in a layered hierarchy of map layers to view combinations of maps. We have also discussed interpolation techniques needed to interpolate from a low resolution to a high resolution point map. The interpolated maps are used for the process of inserting data into the missing data points of the high resolution LiDAR maps. The processes combined are a step closer to creating a tool set for a modern disaster support system.

In the future work, we plan to complete a comprehensive analysis of interpolation techniques used for interpolating low resolution map data to a higher resolution map, categorizing multiple maps containing different landscapes; water and land, land only, and maps with cliff faces. Using ease-in, ease-out, and ease-in-out interpolation functions need to be examined further to specify which function needs to be implemented at specific points. Using the gradient value between points, we can determine which

function to use. If the gradient is over a certain threshold, then we can use a step function to represent the side of a building. This has two positives; the visualization will have improved accuracy but also the accuracy for of spatial analysis used for object clearance based for example on pathfinding algorithms [11].

Acknowledgments. This research is co-supported by the NSF key project with grant no (61332017) and the National project with grant no (2013BAH24F00).

References

1. Tully, D., El Rhalibi, A., Merabti, M., Shen, Y., Carter, C.: Game based decision support system and visualisation for crisis management and response. In: 15th Annual PostGraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (2014)
2. Kwan, M.P., Lee, J.: Emergency response after 9/11: The potential of real-time 3D GIS for quick emergency response in micro-spatial environments. *Comput. Environ. Urban Syst.* **29** (2), 93–113 (2005)
3. Kopylec, J., D'Amico, A., Goodall, J.: Visualizing cascading failures in critical cyber infrastructures. In: Goetz, E., Shenoi, S. (eds.) *Critical Infrastructure Protection*. IFIP, vol. 253, pp. 351–364. (2008)
4. Rinaldi, S.: Modeling and simulating critical infrastructures and their interdependencies. In: *Proceedings of the 37th Annual Hawaii International Conference on System Sciences 2004*, pp. 1–8 (2004)
5. Scarlatos, P., Kaisar, E., Teegavarapu, R.: Modeling and simulation of catastrophic events affecting critical infrastructure systems. In: *Proceedings of the 11th International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering (MMACTEE 2009)*, pp. 334–346 (2009)
6. Dudenhofer, D., Hartley, S., Permann, M.: Critical infrastructure interdependency modeling: a survey of U.S. and international research. In: Report from U.S. Dep. Energy Natl. Lab. Oper. by Battelle Energy Alliance, August, 2006. doi:[10.2172/911792](https://doi.org/10.2172/911792)
7. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 417–424 (2000)
8. Xia, J.C., El-Sana, J., Varshney, A.: Adaptive real-time level-of-detail based rendering for polygonal models. *IEEE Trans. Vis. Comput. Graph.* **3**(2), 171–181 (1997)
9. Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N.: Real-Time, Continuous Level of Detail Rendering of Height Fields 1 Introduction 2 Related Work 3 Motivation
10. Hoppe, H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In: *Proceedings of the Visualization 1998 (Cat. No. 98CB36276)* (1998)
11. Harabor, D., Botea, A.: Hierarchical path planning for multi-size agents in heterogeneous environments. In: 2008 IEEE Symposium on Computational Intelligence and Games, CIG 2008, pp. 258–265 (2008)