LiDAR Surface Reconstruction for Battlefield Route Planning

Student Name: Daniel Wilkinson

Supervisor Name: Ioannis Ivrissimtzis

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University

***Abstract* —**

Context/Background

When a military group is moving on foot through an area that they may not know a great deal about, they must consider the difficulty of a proposed route as well as the potential for exposure to an enemy force. Enemy movements are difficult to predict, however strategic advantage can be roughly predicted based on a finite set of factors. A LiDAR scan could be performed at relatively low cost and risk to human life to get an understanding of the terrain, and this terrain scan can be analysed to find a route that is both efficient and safe.

Aims

This project aims to utilise LiDAR data, provided by the environment agency, to find an optimal route through otherwise uncharted ground with respect to the exertion of those travelling the route, as well as the strategic vulnerabilities that the group would be subject to. This route should then be visualised in an intuitive way for non-technically literate users to understand the proposed route.

Method

To achieve these aims, I plan to create a complete reconstruction of the area through terrain reconstruction algorithms, such as cocone, and provide the user with an interface to fly through the reconstruction. To find an optimal route, I plan to develop different route planning algorithms based on Dijkstra's shortest path algorithm and A* search, to find the most efficient method for this problem.

Proposed Solution

I propose a method of using A* search with several small, easily calculable heuristics in a user-defined weighting that can be changed based on situation dependent factors. This can be coupled with relatively basic computer graphics techniques to achieve an intuitive and efficient system to find the best route and brief the user on the selection.

***Keywords* —** Route Planning, Surface Reconstruction.

I INTRODUCTION

A Project Overview

In its most basic form, this project asks "Can a LiDAR terrain scan be used to find an optimal route through an area with regard to enemy evasion as well as traditional route optimising metrics?". In the Afghanistan War alone, the British Armed Forces suffered 453 fatalities, (Ministry of Defence 2016) of which 405 were killed as a result of hostile action. (Ministry of

Defence 2015) In an attempt to reduce these numbers in future conflicts, this project aims to use a low-cost LiDAR scan of an otherwise unmapped area of terrain to find a suitable route between two points that will minimise the exertion on those that are trying to undertake the journey, while maintaining their personal safety by avoiding areas that would give an enemy assailant a significant strategic advantage. If this system works as intended, then the number of engagements where British Forces are at a disadvantage should decrease and when engagements do happen, the group should have more energy to efficiently fight with. To simulate war-time conditions, the LiDAR data of the UK provided by the Environment Agency will be used exclusively. (Environment Agency 2018) The LiDAR scans provided, like most commercial scans, consist of two separate models: a Digital Surface Model (DSM) and a Digital Terrain Model (DTM) both at 2m, 1m, 50cm, and 25cm resolution. A DSM shows the height of the highest surface, whether it be ground or another object above it, whereas a DTM shows the height of the underlying ground regardless of what is above it as is shown in Figure 1. Coverage of England in this LiDAR scan can be seen in Figure 2. High resolution models are only available in certain areas, however as the resolution becomes finer, more locations are null-valued, meaning even areas scanned at a high-resolution will require the use of low-resolution models to complete the terrain.

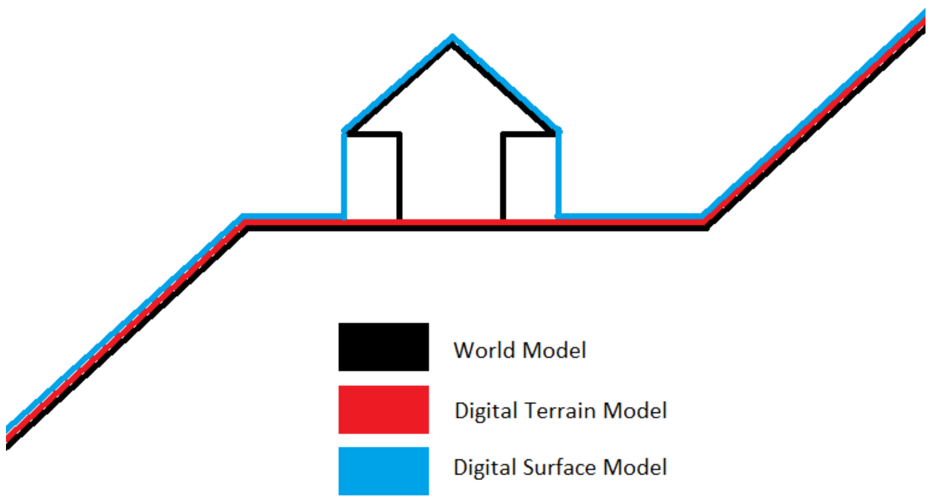


Figure 1: Diagram showing the difference between DTM and DSM

Such a system will require development from many angles. Firstly, the reconstruction of the terrain from the LiDAR data will require investigation of different algorithms that claim to be able to efficiently and accurately create a mesh from point data. Once the mesh has been created, Computer Graphics techniques will need to be used to display the mesh effectively while being computationally cheap enough to run smoothly on a potentially low-powered laptop in poor conditions, such as the ruggedised laptops used by the military. A user interface will be needed that can be used by soldiers who may not be entirely literate or technologically competent, as any misunderstanding or mistakes could cost lives. In 2011, 38% of British Army recruits had below Level 1 literacy (the minimum literacy judged to be required to function effectively in society), which is significantly higher than the UK average of 15%. (House of Commons Business, Innovation and Skills Committee 2014) Finally, and arguably most importantly, the route through the terrain must be created by the system while considering a plethora of factors

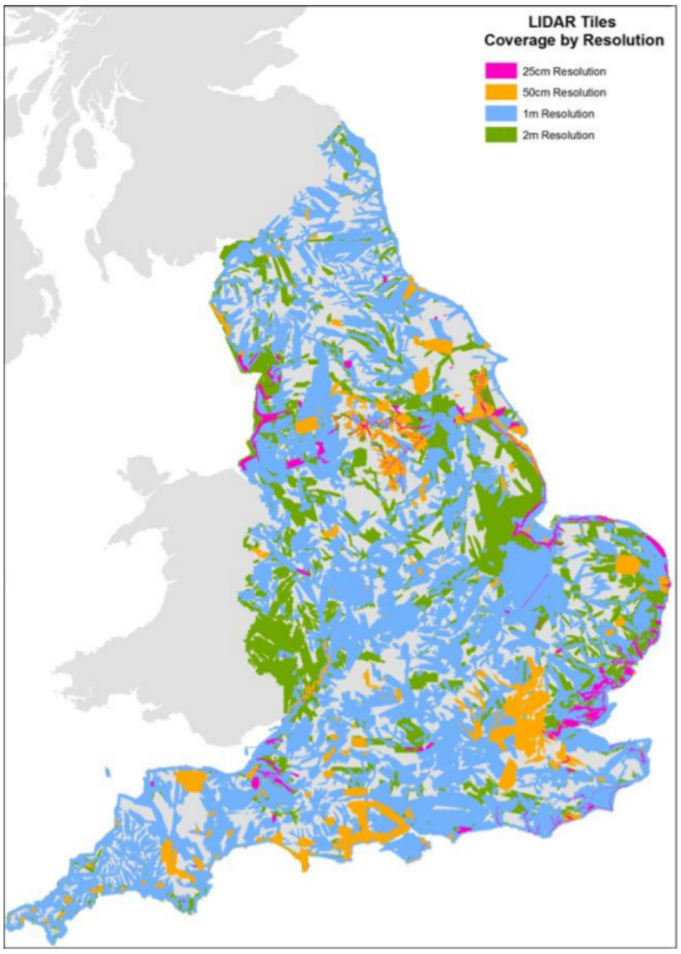


Figure 2: Coverage of LiDAR scan in England, taken from (Environment Agency 2016)

outside of the standard minimum-distance route planning. The route planning element of this project must be thorough enough that it achieves the goals set out earlier, while not being overly computationally expensive, as routes may need to be found in very short timeframes to achieve a mission.

B Stakeholders

This project will directly affect the lives of soldiers on the ground in current and future conflicts, as well as their commanders, who could then spend more time organising the other areas of their mission, while trusting that the system is sorting the navigation aspect for them. From a less direct perspective, this project will hopefully benefit the families of the soldiers who could have otherwise been involved in an ambush. Additionally, the taxpayers of Britain are stakeholders due to increased productivity and efficiency of our Armed Forces making taxes go further.

As the developer of this system, the outcome of the project will affect me personally as it is an opportunity to increase my breadth of skill, and at the same time is my way of using my time at Durham to do something good. Once the project has come to an end, depending on my results, I may submit my findings to the Defence Science and Technology Laboratory (DSTL) to offer the system to them for further development. Therefore, while there are no current connections to

DSTL, they may become stakeholders in the project at a later date.

If everything goes to plan, then certain aspects of this system could be improved and adapted to be used in other applications. With a different set of parameters in the route planning section, the system could be adapted to work well in any situation from low-level flight planning to sheep herding through mountainous terrain.

C Deliverables

C.1 Minimum

1. Collect data from the Environment Agency and build a document parser for their file format.
2. Organise the collected data to be able to find all readings for same location (i.e. Surface Model, Terrain Model, multiple resolutions).
3. Generate and render a surface mesh from the organised data.
4. Model the surface as a graph for path generation with edge weight related to the exertion required to travel between the vertices.
5. Generate a minimum exertion path that will take into account distance and elevation change.

C.2 Intermediate

1. Recognise different types of terrain based on difference between neighbouring vertices as well as difference between surface and terrain readings.
2. Recognise potential for cover in each area based on terrain type and features.
3. Implement a route finding algorithm that will take cover analysis into account.

C.3 Advanced

1. Model shadows on generated surface based on sun's position throughout the day.
2. Evaluate potential for silhouetting due to sun's position throughout day.
3. A selection from:
    - Exit planning from high risk positions.
    - Analysis of distance from given position to nearest safe place, or cover.
    - Interactively input and store known enemy locations to influence probability of enemy presence in nearby areas.

D Related Work

Research has been done into possible parameters that can be used as heuristics when planning a route with potential enemy presence. Ruuben and Kreison suggested a set of 10 parameters which adequately allow route planning for convoys of military vehicles over well mapped areas. (2013) The proposed parameters for roads are: length, width, ground type, environment, road type, road infrastructure, maximum speed, maximum bearing capacity (how heavy a vehicle the road can take), units hostility (how bad the people in the area are), and threat (how likely the bad people are to attack). While some of these features, such as road type and maximum bearing capacity, do not apply to this project as we cater more towards an infantry group that are moving on a (potentially off-road) foot patrol, other features are highly transferable, such as units hostility, threat and ground type.

With the possibility of using a genetic algorithm for the route planning section, Gintaras Vaira's work on a genetic algorithm for the vehicle routing problem would be of great use to this project. The thesis discusses popular heuristics for the problem and proposes operators for mutation and breeding between generations and concludes that when breeding, it is more important to find common route sections from the parents and create a child from there than to take parts that appear in a single parent. Efficient pruning methods are also proposed that in testing, nearly half the number of floating point operations required. (Vaira 2014)

Regarding the LiDAR surface reconstruction, work has been carried out to investigate methods of reconstructing unorganised point clouds, such as the Cocone Algorithm, (Dey et al. 2011) which works excellently, providing a reliable accurate reconstruction with boundaries. The industry standard for terrain reconstruction with organised data seems to be regular grid patterns, and work has been done to recognise buildings based on the differences between adjacent points and points on DTM and DSM. Haala, Brenner and Anders worked on urban reconstructions from LiDAR and 2D maps, with excellent results despite reducing the resolution of their LiDAR data. (1998)

The routes planned in this project aim to not only protect the user from enemy interference, but also to minimise the exertion they must give to get to their destination. Minetti *et al* have studied the average energy expenditure of humans while walking up or down different gradients. (2002) Their results can be used to approximate an expenditure function based on the difference in height between two locations. If it is possible to reliably deduce ground type based on the difference between DTM and DSM values and the difference between neighbouring values in each model, then a coefficient can be applied based on ground type to this energy expenditure to give a suitable estimate for all situations (e.g. walking uphill at 5% gradient through forest vs walking downhill at 5% gradient on a road).

II SOLUTION

A Overview

The development of this project can be broken into 4 separate modules which can be build separately, to be used together in the final system. These are:

1. Document parser to load data from bespoke file format used by The Environment Agency.

2. Render engine to display the LiDAR data as a surface for the user to understand.

3. User Interface (UI) to allow the user to specify any route properties that are necessary.

4. Route generator to model data as a graph and find the best route to take.

It was decided that Java would be used as the primary development language for this project, as it provides an appropriate balance between speed of operation and simplicity to use, while also offering a high degree of cross-platform compatibility as most operating systems can run a Java Virtual Machine (JVM). As will be discussed later, extra libraries are required to enable 3D graphics rendering and UI design.

B Document Parser

The data supplied by The Environment Agency uses a header to supply relevant metadata to understand the rest of the file, followed by a space separated value format where each line represents a row of the point lattice, each space denotes a column, and each value is the height of that position in metres. The header specifies the size, location, resolution, and null value of the scan, as can be seen in the example in Figure 3.

```
ncols 1000
nrows 1000
xllcorner     340000
yllcorner     520000
cellsize      1
NODATA_value  -9999
232.7 234.55 233.16 238.89 239.29 234.1 ...
232.22 231.93 232.54 239.06 237.43 234.08 ...
...
```

Figure 3: Header from a LiDAR data file

The parser reads the file line by line, and stores the header values in private variables. Once the header has been read, a 1 dimensional vertices array is created where each set of 3 elements denotes a single point in the grid. Also created is a 1 dimensional indices array where each set of 3 elements denotes a triangle that can be easily rendered later. This method of storing the data was chosen to work easily with the render engine, as it is the most speed critical aspect of the system. Once the vertices have been loaded into the render engine, it would have been possible to create a 3D array for the data, however the calculation to find the index of the value that we want to find is simple and did not warrant the extra development and computation time to do this structure conversion.

C Render Engine

To render the LiDAR data, a graphics library was required to provide a 3D world in which to display the terrain. This decision was part of the process of choosing a development language for the project. The choices were to use a high-level game engine, such as Unity, or a low-level graphics library, such as OpenGL. A game engine would make development easier as most aspects of the rendering would have already been written to a higher standard than could be written in the timescale of this project, however unnecessary extra features would also most likely be running that may slow the system down on the potentially underpowered target machines. A low-level graphics library could be more difficult to create, as all aspects would need to be made from the ground up, but the resulting system would only be running components that are entirely necessary to the project. From a personal standpoint, a low-level library would also push towards deeper understanding of graphics concepts. Consequently, it was decided that a low-level graphics library would be the better choice. OpenGL is an extremely common graphics library, however it is designed to run on Windows systems through C/C++. This project aims for a high degree of platform independence, so the LightWeight Java Game Library (LWJGL) was chosen as the library of choice. The LWJGL is a Java port for OpenGL methods, allowing similar low-level development that can be expanded on by most graphics developers if the project is continued, while running on any JVM.

The Render Engine requires the vertices and indices arrays generated by the file parser, as well as points on the route generated by the rest of the system and any known enemy locations. The vertices array defines each point of the data, and the indices array specifies which points should be joined into a triangle when rendering. A visual representation of the relationship between these arrays can be found in Figure 4.

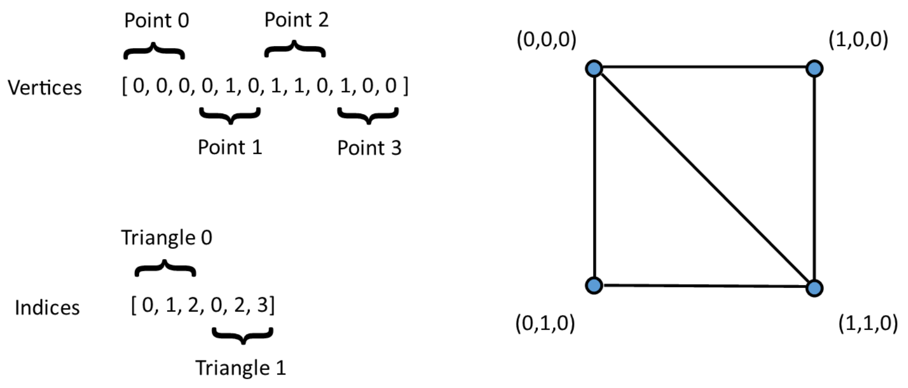


Figure 4: Relationship between vertices and indices array in rendering

The Environment Agency LiDAR scan is presented in a regular grid form, so the use of advanced reconstruction algorithms such as the Cocone Algorithm do not provide any useful improvement in output for their extra cost in computation, since it is designed to create a surface from unorganised point clouds, whereas the provided dataset uses organised points in a grid. As a result, a dot to square to triangle approach was used as is shown in Figure 10. Edges are created between neighbouring points to form a regular square grid across the model. Next, for each square in the grid, opposite corners are compared, and the diagonal with the lowest difference in height is selected to turn the square into two easily-rendered triangles while also providing the most natural bending point for the difficult to render 4-point polygon.

Once the surface has been initially created, the colour of the vertices is decided by the use of a colour map to the vertex height. The highest point of the terrain is coloured red and the lowest point is coloured blue, transitioning through the hues as the height varies. Other methods considered for colouring were gradient colour mapping, which looked cluttered for DSM renders and sparse for DTM renders, and texture mapping, which had the potential to be misleading as it could suggest that the ground is a certain type when we do not know that it is. Texture mapping

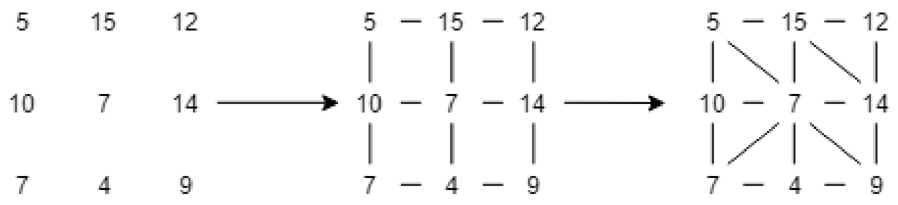


Figure 5: Visualisation for dot-square-tringle mesh generation

also made slight variations in height difficult to see. Height colour mapping seems to be the optimal choice for ease of understanding of terrain, although when flying through the scene close to the ground, it can be difficult to distinguish between different objects on the DSM render.

After the initial colouring, the points from the route are coloured white and any known enemy locations are coloured black. This allows the user to easily tell what the proposed solution from the system is.

D User Interface

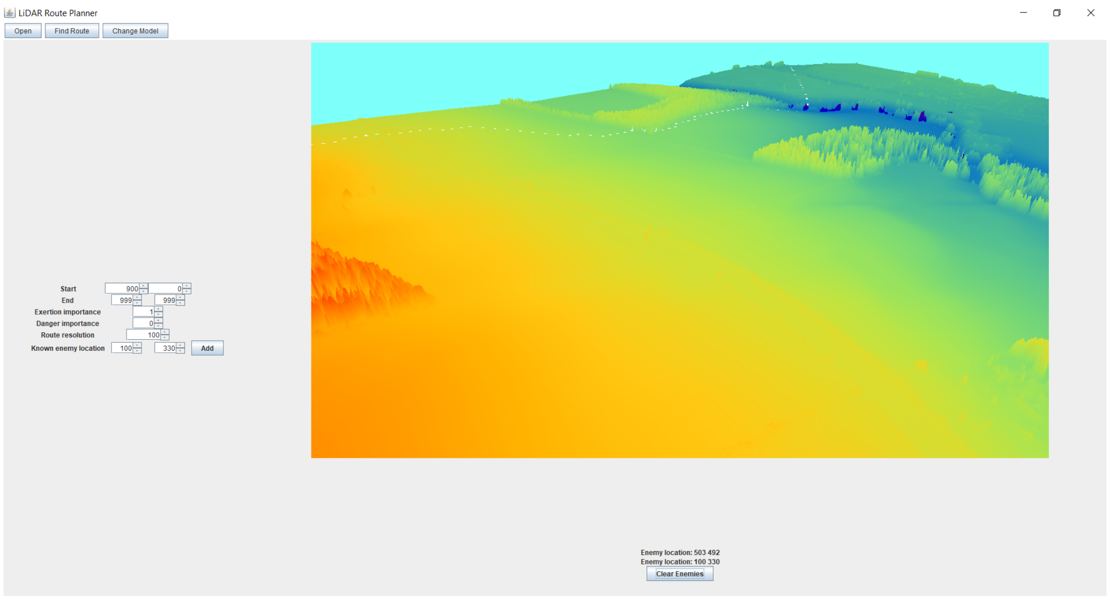
Given the time scale of this project, the primary objective when selecting a User Interface creation library was to be able to rapidly develop a front end that will suffice for testing. If the project were to be put into production then much more time would need to be spent designing an interface that can be used with limited literacy skills due to the limitations discussed earlier. There is already a huge amount of research in UI design, so the focus on this project will be more towards the algorithmic side yet to be discussed. Consequently, Java Swing was chosen as the UI library as it is incredibly efficient in development time to create a working interface, and there is plenty of scope for later development into a more polished product.

The design of the UI can be broken down into 4 main areas. These are the toolbar, the option panel, the route information panel and the render display. Using swing it is possible to create each panel independently and tie them all together into a single coherent display. The toolbar contains all of the buttons that control the main functions of the system. The option panel is required to specify all parameters for route selection, including the start and end point, importance of exertion and danger mitigation, resolution to use, and known enemy positions. Swing offers inputs that validate integers dependent on limits and increments specified in the code. This allows validation to be performed before any other functions are called. The render panel contains a canvas that LWJGL can access, allowing for the rendering to be in the same window as the rest of the panels. A view of the current user interface can be found in Figure 6, where the toolbar is at the top, the option panel is on the left, and the rendering canvas is on the right with the route information panel beneath it.

E Route Generator

The main area of focus in this project was the Route Generator. Other areas of development have helped to support the usability of the system, however in an academic environment, the requirement for strong algorithmic progress outweighed the need for a pretty front end.

Firstly, I needed to create a class that can take the vertices array given by the file interpreter and form a graph from the data. To do this, I chose to store the vertices array as is, since a 1 dimensional array has no excess data to store and is fast to read from, as opposed to instances of



Start 900 0
End 999 999
Exertion importance 1
Danger importance 0
Route resolution 100
Known enemy location 100 330

Enemy location: 503 492
Enemy location: 100 330

Figure 6: User Interface of system

a custom position class. I then wrote methods that will calculate any desired attributes, such as the height of a position, any neighbouring points at a specified distance, and the cost of moving from one position to another, as will be discussed later.

The next point for development was the route itself. I chose the A* algorithm as the basis of my route finding algorithm, as it is proven that with admissible heuristics (never overestimates cost to reach goal) then A* search will return an optimal solution, and with a consistent heuristic (value for current position is no greater than step cost plus value for successor position), A* is optimally efficient.(**?**) I started to develop my algorithm by making an implementation of Dijkstra's shortest path algorithm where the weight of each edge is the flat ground distance covered plus the elevation change. When keeping track of which positions have been discovered and finalised, I chose to use ArrayLists instead of arrays set to the size of the world, as space complexity is an issue in this system, where the target computers are likely to be limited in RAM and CPU power. I decided that the potential time tradeoff was worth the extra free memory. Admittedly, this does cause a drastic change in computation time for complex routes, so I decided to implement dynamic scaling of the graph based on the 'as the crow flies' distance between start and end points. While this cannot guarantee reasonable computation times as complex routes, where the total cost of the optimal path is high, still require a large number of points to be visited before we can be sure that we have the correct route.

Once the Dijkstra implementation was working properly, I converted the method to use a separate cost function and use a heuristic function when deciding which node to visit next. These cost and heuristic functions could then be changed easily as the project progressed. For the heuristic function, I chose to use the 'as the crow flies' distance between the current position and the end point. For a minimum exertion path, this heuristic can be admissible with an appropriate cost function, however it is not quite consistent.

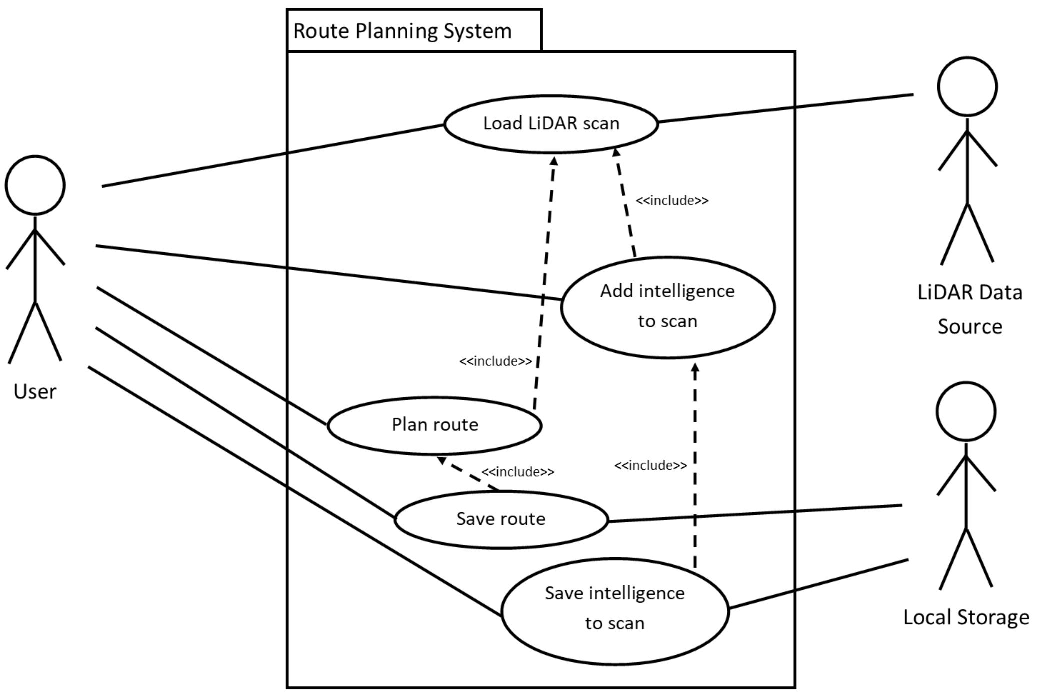


Figure 7: Use case diagram for project system

III DESIGN

A Requirements

After analysing the deliverables set out above and considering the possible use cases of the system, as can be seen in the use case diagram in Figure 7, this section will define the system's functional requirements in Table 1 and non-functional requirements in Table 2 as well as ranking each requirement on its importance to the final product.

Table 1: Functional Requirements

ID	Requirement	Priority
FR1	Environment agency data can be interpreted	High
FR2	Terrain mesh is generated from LiDAR data	High
FR3	Terrain is rendered to display variations in height	High
FR4	Valid route can be found between specified points	High
FR5	Route can be plotted onto rendering of terrain	High
FR6	Terrain features can be categorised into types	Medium
FR7	Potential for cover is added into rendering	Medium
FR8	Likelihood for enemy attack is accounted for in route	Medium
FR9	Shadows and silhouettes are rendered onto terrain	Low
FR10	Advanced route parameters are taken into account	Low

Table 2: Non-Functional Requirements

ID	Requirement	Priority
NFR1	The system can run on a low-powered laptop without stuttering	Medium
NFR2	Users can interact with the system intuitively and with limited literacy	Medium
NFR3	Routes suggested avoid danger as far as possible	High
NFR4	Routes suggested are efficient in terms of exertion	High

B Equipment Used

To get the initial LiDAR data, the Environment Agency used an airborne LiDAR scanner, however if the system were to be used in its intended purpose, there are commercially available LiDAR pods to be fixed to standard commercial drones as well as manned aircraft. This provides choice and limits cost to gain an accurate representation of the terrain. The LiDAR files provided have a header detailing the size, resolution and location of the area represented. After the header is a space / newline separated grid of height values to 2 decimal places.

The primary development machine will be a laptop, due to its availability and its more realistic likeness to the situation that the system is likely to be used in. On the field, the worst-case scenario (which the system must be built around) will be a low-powered laptop without a dedicated graphics card, similar to the development machine. As a result, the project will naturally steer towards graphical efficiency, due to the inconvenience to the development if the rendering has a low framerate.

To prototype a document parser quickly and check basic functionality, Python was used due to its ease of rapid development. While Python is great for simple proof-of-concept development, it was not designed to perform efficient graphical rendering. The options for a final development language were either C++ or Java; C++ offers unrivalled performance on Windows machines due to its low-level compilation and direct access to OpenGL libraries, while Java can be run on all major platforms with simple setup and is arguably easier to develop on. After considering the options, Java was selected as the language that the system would be built on as versatility and the prospect of fewer blockers in the development cycle appeal more than marginal efficiency gains for this style of project.

Once a language was finalised, the next choice was which library to use to create the rendering of the terrain. The choice basically boiled down to a low-level graphics library, such as OpenGL or a high-level game engine, such as Unity. A low-level library would allow development of only what is absolutely necessary for the system however would be more complex to implement as all aspects of the engine would need to be built from the ground up. A game engine would streamline the development process, however the end result could end up being bloated with many unnecessary additions, which would require a more powerful machine to run. The LightWeight Java Game Library (LWJGL) was selected as the graphics library that offered the best compromise, as it allows Java to use OpenGL functions, which will keep the computer load to a minimum assuming the correct software engineering practices are followed, while providing opportunity to implement any possible features that the system could require.

C System Architecture

The main requirements for the system are to be able to take in LiDAR data files, render the files for viewing, find an appropriate route through the terrain, and interact with the user. To read the

LiDAR data files, a custom parser class will be created for the format of the data supplied by the Environment Agency, however this class will be decoupled as far as possible from the rest of the system so that if the system is deployed after this project then it would be simple for a software engineer to write an equivalent class to parse the file format that they choose to use. On top of the stock LiDAR data, the system will be capable of storing additional intelligence from the user, so the next class in the system will be responsible for pairing the intelligence file with its respective location in the LiDAR scan. This does not need to be as lightly coupled as the LiDAR scan parser as the file format will be defined in this project and should not need to be modified extensively if the LiDAR source changes format.

Once we have all relevant data within the system, we need to create a terrain mesh to be rendered from the LiDAR points. To do this, a mesh generator class will be written to convert the points into easily-rendered triangles, and pass the mesh to a render engine package to colour the vertices and the fragments using OpenGL functions through the LWJGL. Initially, no extra graphical processing will be done so that computational load is kept to a minimum, allowing the system to be run smoothly on any modern machine. When available, multiple scan resolutions can be used to limit the intensity of rendering by allowing nearby areas to be rendered accurately and far away areas to be less detailed. Later in the information flow, the render engine will be given a route to add to the terrain rendering, which will be represented as a change in colour on the surface of the terrain. While this may cause issues with users that have colour-blindness, the alternative when displaying the route would be to create an elevation change in the model, however doing so would cause a misrepresentation of the land and false information could be taken from viewing the landscape as a result.

The route-finding element of the system will require a weighted directed graph with weights based on the exertion for a group to move between the locations. The computation required to convert the render mesh into the graph is comparable to starting again from the raw LiDAR points, however the latter it is a shorter, more easily understandable pipeline. Consequently, a graph generator class will be created to take the point cloud and pass the subsequent graph to a route handler class. This route handler will act as an organiser for multiple route finding algorithm classes, conveying start and end points as well as other route preferences, while choosing which algorithm to use. The algorithm classes will purely be concerned with finding the best possible route based on a number of heuristics concerning the enemy evasion aspect of the problem.

Finally, the system will need a user interaction handler to allow the user's commands take precedence over all other computation. This handler will be able to convert the commands into actions within the system and send messages across the classes to wherever they need to be acted on. The user interface will require careful consideration due to the limited literacy assumption being made in this project. Any user will need to be able to load a scan, input intelligence such as known enemy locations, known roads, tried and tested safe routes, friendly locations, etc. as well as setting start and end points for the route, and other parameters such as enemy threat and time criticality. Once intelligence has been entered, it should be able to be saved.

Once a stable build of the system is completed, a single .jar executable can be created, which will allow usage over a broad range of machines on any common Operating System. A diagram showing information flow between entities in the system can be found in Figure 8.

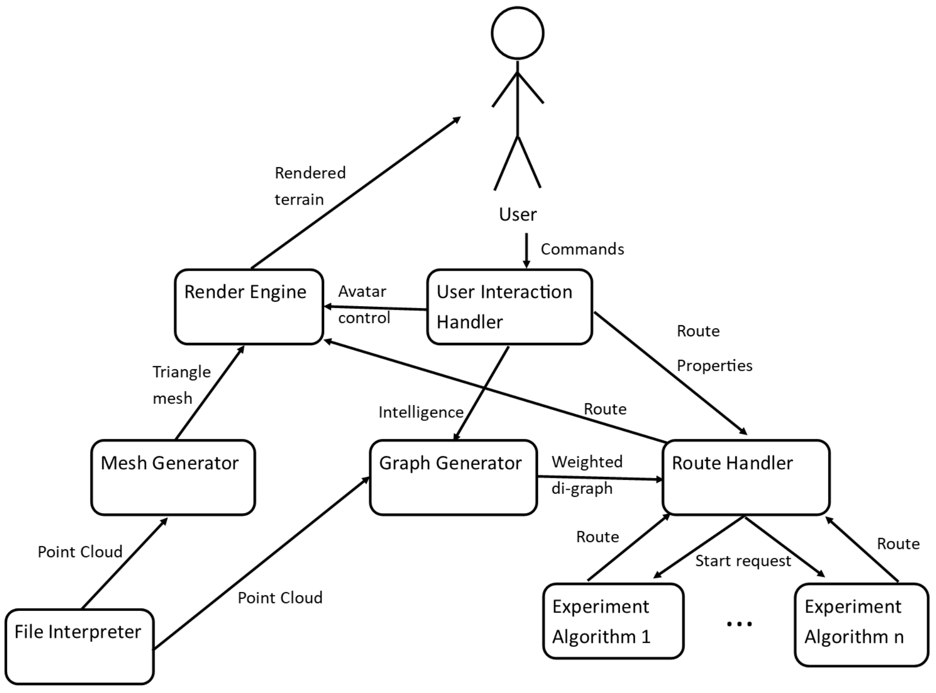


Figure 8: Diagram of system components and information flow between them

D User Interaction

The system will first require the user to load a LiDAR data file. This will be converted into a mesh on loading, which will not be saved to save storage space, as currently a 1 metre resolution scan of 1 km^2 is roughly 6MB. Therefore, a full scan of the UK could be estimated at 1.4TB of data, meaning that a complete offline copy of a theatre of operation could feasibly be carried in an external hard drive, however the cost for storing a mesh over a point cloud would tip this possibility over the edge of reasonable.

Once the mesh has been loaded, the user will be able to access additional intelligence input tools, which will have optional parameters such as intel expiry time for situations such as if an enemy has been temporarily sighted in an area to avoid. When the user is happy with the intelligence on the system, they will be able to save all permanent intelligence into a separate file within the system's data storage. Alternatively, if there has previously been intelligence on the area loaded to render, then the system will be able to match the files and create a complete picture from the start.

With the terrain updated to show the latest intelligence, the user will be able to assign start, end, and waypoints to their route as well as optionally tweaking parameters detailing what weight some simplified parameters should be given (threat level, time constraint, etc.). The system will then be set computing an optimal route based on all information at its disposal. Upon completion, the route will be rendered onto the terrain in a colour to make it stand out from the model and the user will be able to walk through the virtual world using the WASD and arrow keys, as in Figure 9.

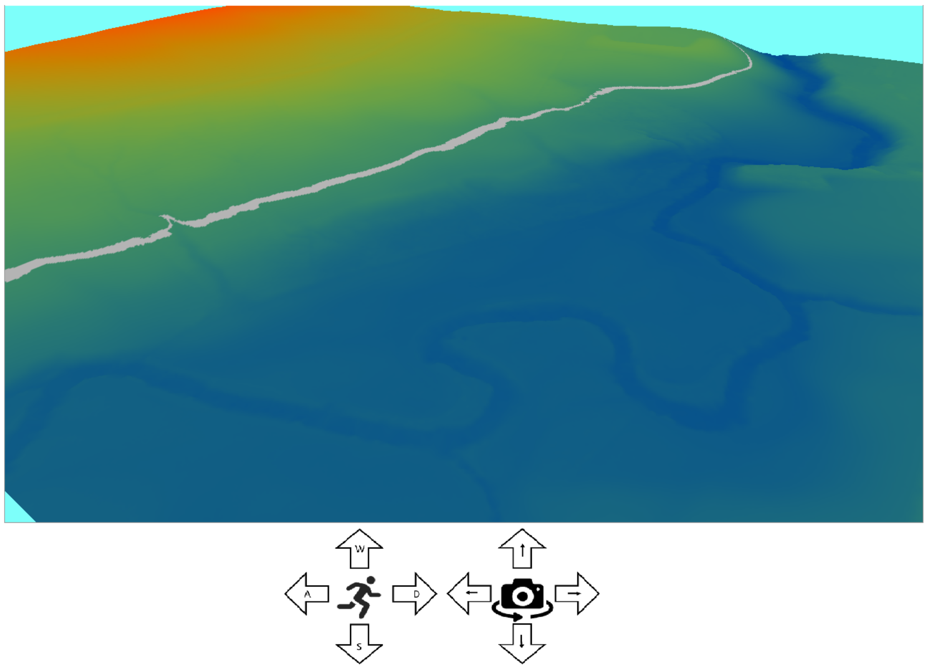


Figure 9: Reconstruction from proposed system indicating path to follow in white with basic user instructions

E Algorithms

For a successful project, algorithms will need to be created and optimised to generate a surface mesh from the point cloud data and to find a suitable route through the terrain.

The Environment Agency LiDAR scan is presented in a regular grid form, so the use of advanced reconstruction algorithms such as the Cocone Algorithm do not provide any useful improvement in output for their extra cost in computation, since it is designed to create a surface from unorganised point clouds, whereas the provided dataset uses organised points in a grid. As a result, a dot to square to triangle approach will be used as is shown in Figure 10. Edges are created between neighbouring points to form a regular square grid across the model. Next, for each square in the grid, opposite corners are compared, and the diagonal with the lowest difference in height is selected to turn the square into two easily-rendered triangles while also providing the most natural bending point for the difficult to render 4-point polygon.

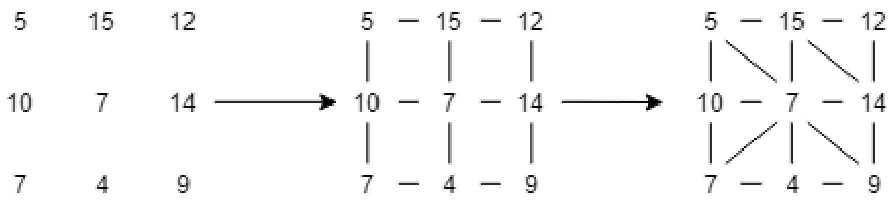


Figure 10: Visualisation for dot-square-tringle mesh generation

In the dataset, there are many locations (particularly in the DSM) that are null-valued. In an attempt to limit the impact of these null areas, the reconstruction algorithm will find the surface area of the null patch and compare it to a threshold. If the surface area is above the threshold, then to assume anything about that location would be dangerous as an advantageous feature for an enemy to occupy or an impassable object could be there however we would have no way of knowing. The safest thing to do is label this area as a potential risk and attempt to avoid coming close to it. If, on the other hand, the surface area is below the threshold then it should be safe to assume that no significant threat can reside in such a small area and so it would be better to interpolate likely values for the null-valued area to aid route planning.

The algorithm that will be used to generate a route could be deterministic, such as A* search, or non-deterministic, such as a genetic approach. A* search would be provably optimal as long as we use admissible heuristics, (Hart et al. 1968) however the algorithmic complexity would mean that calculating a route over large distances would take an unreasonable amount of time. Genetic algorithms do not guarantee an optimal, or even a good solution, however probabilistically speaking, with enough generations, they are likely to provide a near-optimal solution while taking a much shorter time to compute than A* search. Additionally, if an A* based algorithm is the only one developed, then a change to the specification of the algorithm could be very difficult to implement. Genetic algorithms are extremely good at handling all kinds of problems, so adding extra features should not require much work.

Traditionally, these algorithms are used to find shortest path solutions to directed or undirected graphs. The use case for this system requires that the algorithms are modified to use a number of different analytical parameters, each with a weight that can be changed by the user to suit the situation of the mission.

Without being able to test the performance of each algorithm, there is no way to choose a single route that will certainly be the best. Both algorithms will be developed in parallel and once both are working as well as possible, they can be compared in terms of route optimality, memory used and computation time to decide ultimately which will be most effective in the final system, or if both have situations that they excel at and it would be worthwhile to keep both algorithms in the final product.

IV TESTING AND EVALUATION

The quality of the system will be quantified by metrics that need to measure the accuracy of reconstruction, the effectiveness of the route planned, and the burden on the system that runs it.

Reconstruction accuracy can be checked by visualising a possible route through an easily identifiable area, such as the Lake District, and recording a video of a real-world hike through the terrain. By showing both representations side-by-side, we can compare the likeness produced by the system to the real thing.

Route effectiveness would be best quantified by a score awarded by an expert in the field. As the effectiveness is such a subjective topic, the expert will be asked to experiment with the user-defined parameters before giving an opinion so that the system has chance to adapt to their personal preference. Coupling expert opinion with comparisons with existing hiking route finding applications should give a clear indication of the success of the project in creating a good route.

To test system burden, the final product will be run on a laptop without a dedicated graphics card, and the CPU and memory usage will be recorded as well as the system temperature. If the laptop can run the system continuously for a specified period of time, while maintaining reasonable resource usage and temperature, then the system will pass the test. Comparisons to existing applications may not be possible as trends in computationally expensive tasks are tending towards offering a solution as Software as a Service (SaaS), meaning that the best systems will not burden the local machine with the task of computing the route. From the use case of this project, it will not be possible to host this software in SaaS as reliable internet connections can not be relied on in combat situations.

References

Dey, T. K., Dyer, R. & Wang, L. (2011), 'Localized cocone surface reconstruction', *Computers Graphics* **35**(3), 483 – 491. Shape Modeling International (SMI) Conference 2011.

Environment Agency (2016), 'Environment Agency LIDAR data Technical Note'. `http://environment.data.gov.uk/ds/survey/docs/Environment_Agency_LIDAR_Open_Data_FAQ_v5.pdf`.

Environment Agency (2018), 'LiDAR Survey of the United Kingdom'. `http://environment.data.gov.uk/ds/survey/#/survey`.

Haala, N., Brenner, C. & Anders, K.-H. (1998), '3D urban GIS from laser altimeter and 2D map data', *International Archives of Photogrammetry and Remote Sensing* **32**, 339–346.

Hart, P. E., Nilsson, N. J. & Raphael, B. (1968), 'A formal basis for the heuristic determination of minimum cost paths', *IEEE transactions on Systems Science and Cybernetics* **4**(2), 100–107.

House of Commons Business, Innovation and Skills Committee (2014), 'Adult Literacy and Numeracy'. `https://publications.parliament.uk/pa/cm201415/cmselect/cmbis/557/557.pdf`.

Minetti, A. E., Moia, C., Roi, G. S., Susta, D. & Ferretti, G. (2002), 'Energy cost of walking and running at extreme uphill and downhill slopes', *Journal of applied physiology* **93**(3), 1039–1046.

Ministry of Defence (2015), Operations in Afghanistan. `https://www.gov.uk/government/fields-of-operation/afghanistan`.

Ministry of Defence (2016), 'Op Herrick casualty and fatality tables to 28 February 2015'. `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/503022/20160222_Op_Herrick_Casualty_Tables_Feb_16_REVISION.pdf`.

Ruuben, T. & Kreison, O. (2013), Route planning in asymmetric military environments, *in* 'Future Generation Communication Technology (FGCT), 2013 Second International Conference on', IEEE, pp. 73–80.

Vaira, G. (2014), Genetic Algorithm for Vehicle Routing Problem, PhD thesis, Vilnius University.