D2.1.2 Final Viewshed Component

Phil Bartie, William Mackaness, Morgan Fredriksson, Jürgen Königsmann

Distribution: Public

SpaceBook

Spatial & Personal Adaptive Communication Environment:
Behaviours & Objects & Operations & Knowledge

Deliverable 2.1.2
1 July 2013

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	270019
Project acronym	SpaceBook
Project full title	Spatial & Personal Adaptive Communication Environment: Behaviours & Objects & Operations & Knowledge
Instrument	STREP
Thematic	Priority Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2011 / 36 Months
Security	Public
Contractual date of delivery	M28 = 1 July 2013
Actual date of delivery	1 July 2013
Deliverable number	D2.1.2
Deliverable title	Final viewshed component
Type	Prototype
Status & version	Final 1.0
Number of pages	22 (excluding front matter)
Contributing WP	2
WP/Task responsible	UE, LM
Other contributors	
Author(s)	Phil Bartie, William Mackaness, Morgan Fredriksson, Jürgen Königsmann
EC Project Officer	Franco Mastroddi
Keywords	Viewshed, visibility, line of sight, ray casting, vista space

The partners in SpaceBook are:

Umea University	UMU
University of Edinburgh HCRC	UE
Heriot-Watt University	HWU
Kungliga Tekniska Hogskola	KTH
Liquid Media AB	LM
University of Cambridge	UCAM
Universitat Pompeu Fabra	UP

For copies of reports, updates on project activities and other SPACEBOOK-related information, contact:

The SPACEBOOK Project Co-ordinator:
Dr. Michael Minock
Department of Computer Science
Umea University ˚
Sweden 90187
mjm@cs.umu.se
Phone +46 70 597 2585 - Fax +46 90 786 6126

Copies of reports and other material can also be accessed via the project's administration homepage, http://www.spacebook-project.eu

Contents

1 Overview 2
2 Visibility Modelling Background 4
2.1 Isovist (2D) 5
2.2 Feature of Interest Target Based Visibility based on 3D Ray Casting 5
2.2.2 Targets 7
2.3 Ray Sweep Viewshed 9
3 Implemented solutions 10
3.1 Line-of-Sight Method 10
3.2 Viewshed Method 10
3.2.1 Ray Scan Sequence 11
3.2.2 Vertical Extent 11
3.3 Raw Visible Point Details Recorded with each Visible Cell 12
3.4 Zonal Statistics – Visibility Metrics 13
3.4.1 Point Count 13
3.4.2 Distances 13
3.4.3 Bounding Points 13
3.4.4 Foreground 13
3.4.5 Elevations 14
3.4.6 Areas 14
3.4.7 Angles 15
3.4.8 Cache System 15
3.5 Example of Output 16
3.6 Using the Visibility Engine with the City Model 17
3.7 Examples of How the Output May be Used 20
4 Conclusion and Future Work 21
5 References 22

Executive Summary

Current location based services (LBS) filter information based on proximity, either in Euclidean or network space. One of the novel aspects of the SPACEBOOK project is to include 'vista space filtering', allowing for features to be included or excluded based on their visibility from the user's location. This requires access to detailed digital surface models (DSM) which incorporate topography and surface objects (e.g. buildings, trees).

By knowing what is in the field of view, the system is able to be more efficient in the description of routes, and to describe prominent features, landscapes and landmarks independently of their proximity to the listener. A richer understanding of when and where landmarks fall into view offers exciting intuitive ways of modelling environment interactions and descriptions. It also comes with challenges requiring us to 1) devise a rich set of metrics that enable us to describe where objects are in the field of view, 2) to handle the uncertainty in location and facing direction of the user, 3) precision and accuracy of the DSM and its referencing with the topographic 2D layer.

This paper outlines the ways that visibility modelling can be computed, and the advantages and disadvantages of each approach. The user may request information ('pulling'), but also information will be volunteered by SPACEBOOK to the user ('pushing') based on what is in the field of view. Furthermore relevant landmarks may be identified along a route, or near junctions, to assist in forming more natural wayfinding instructions.

1 Overview

Location Based Services (LBS) typically use proximity to determine geographical relevance when filtering information, for example the closest park determined by Euclidean space, or network space. People, however, often refer to items in vista space (Montello 1993), that is a region of interest defined by visibility.

Figure 1 shows an example of the different regions which would be defined when using a filter based on Euclidean, Network, and Vista space for an observer (green dot) in a city street in Edinburgh. While Euclidean space ignores topography and the city layout it is a particularly useful way to define rapidly a region around the user which may act as a filter for removing less relevant items.

Network space considers the total travel cost along defined pathways (e.g. roads, tracks). The cost is usually based on distance or time, but may consider any numerical variable. It is useful in giving the user an estimate of how difficult or easy it is to reach a feature, but requires greater computation and is not suitable in all cases (e.g. finding the nearest tree in a park).

Vista space reflects the surrounding regions which can be viewed from a specified location, and is more fragmented in nature. It is the most computationally expensive of the three spaces described here, as a high resolution city model including elevations is required. For this SPACEBOOK has access to a LiDAR sourced Digital Surface Model (DSM) and a Digital Terrain Model (DTM) for the pilot study region in Edinburgh.

Euclidean space (200m radius)

Network space (200m travel distance)

Vista Space (all visible items)

Figure 1: Spatial filter examples

Elevation data for the Edinburgh City model is sourced from Light Detection and Ranging Data (LiDAR) which capture both the topography and surface objects at a high spatial resolution. An example of the elevation data and a sample perspective view are shown below.

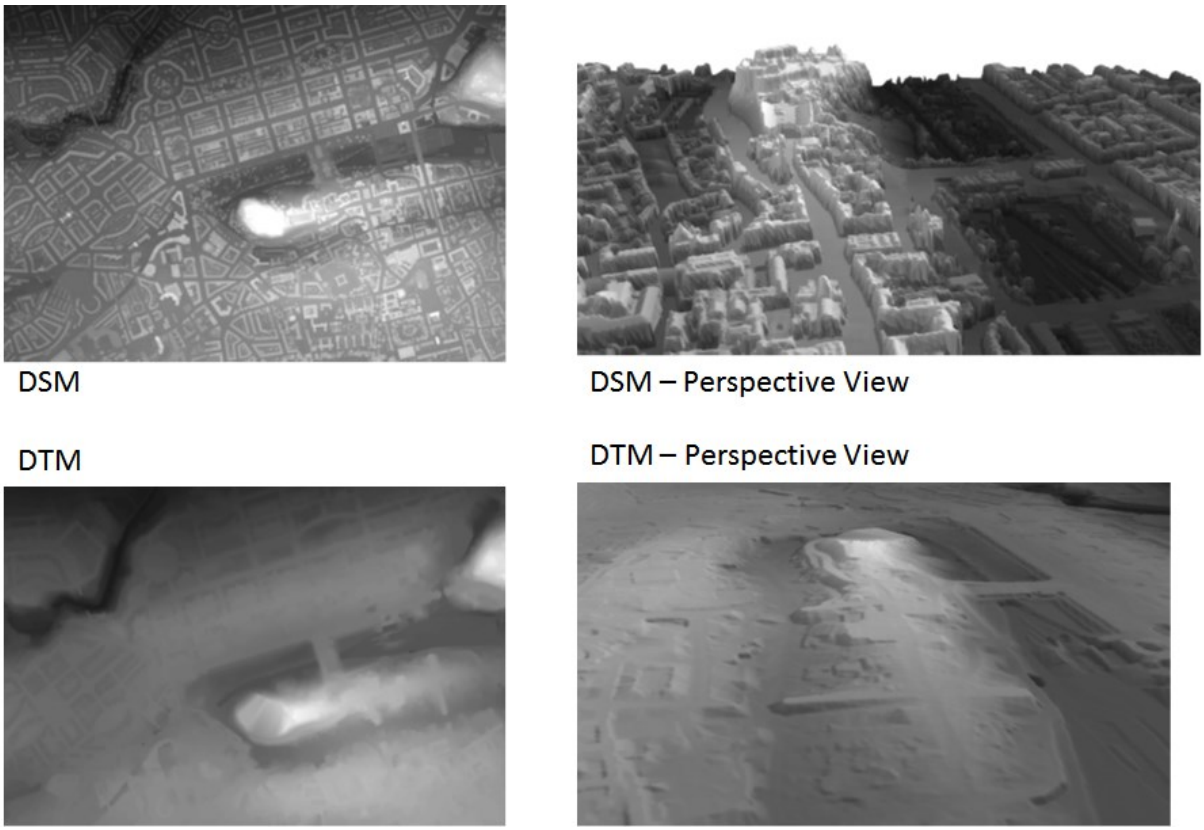

Figure 2: Digital Surface Models

This document describes the visibility engine component and the metrics made available to the rest of the SPACEBOOK system.

2 Visibility Modelling Background

Visibility modelling can be used to determine which regions may be viewed from a specified location. In urban areas this has tended to be based on isovist theory (Tandy 1967; Benedikt 1979) using building plans to determine the limits of view, without consideration for topography or building height. Recently 3D isovist modelling has become a possibility by accessing city elevation models which include the urban form (Morello and Ratti 2009).

In rural regions viewsheds are used to determine what is visible from a vantage point by calculating which cells of an elevation model are visible to an observer (Tandy 1967; Lynch 1976) using a line of sight algorithm (Fisher 1993). Elevation data may be stored in rasters or as vector Triangular Irregular Networks (TIN) (De Floriani and Magillo 1994) and uncertainty may be reflected in the results (Fisher 1994; Fisher 1995). They have been used in a wide range of applications including finding the most

hidden route or those with the best views (Lee and Stucky 1998), studying the visual impact of wind farms (Kidner, Sparkes et al. 1999), and in landscape planning (Fisher 1995) as well as urban contexts (Bartie, Reitsma et al. 2010; Chisholm 2011). Performance has also received a lot of attention as processing can be computationally intensive, especially across high resolution elevation surfaces (De Floriani, Magillo et al. 2000; Rana and Morley 2002; Ying, Li et al. 2006).

There are many ways that visibility modelling can be implemented, a brief summary of those investigated are summarised below. The solution implemented is detailed in Section 4.

2.1 Isovist (2D)

Traditionally urban models have lacked height information for surface features (e.g. buildings, trees) and have therefore used a 2D Isovist approach, whereby all buildings are considered to be infinitely tall and block the view of all more distant objects. This allows for a fast approximation of the closest viewable features using only planimetric map data.

An example of the Isovist space around a user for Edinburgh is shown in Figure 3, where no consideration is given to the height of Buildings A and B, and even if B was very tall and visible over the top of A it would be still considered out of sight using an isovist approach.

Figure 3:Isovist Region (2D) for an Observer in Edinburgh

While this provides some understanding of the visible space it has a number of shortcomings, and with the availability of LiDAR based DSMs, such as those available for Edinburgh (Figure 2), more sophisticated approaches which considers building elevation and views across rooftops can now be considered.

2.2 Feature of Interest Target Based Visibility based on 3D Ray Casting

The simplest way to calculate the visibility of a feature with consideration for surface elevation is to use a ray casting approach, whereby rays are sent from the user towards pre-designated targets on each feature of interest (eg Old College). The user's elevation is read from the DSM and the elevation values between the user and target are sampled to determine if the target is blocked from view. If the target viewing angle is steeper than the view to all cells in between, then the target is considered

visible. This is repeated for all the cells within the Feature of Interest (FOI) to give a score of how much of that object is visible. Using this approach it is possible to work out which parts of features are in view (eg Old College - Figure 4).

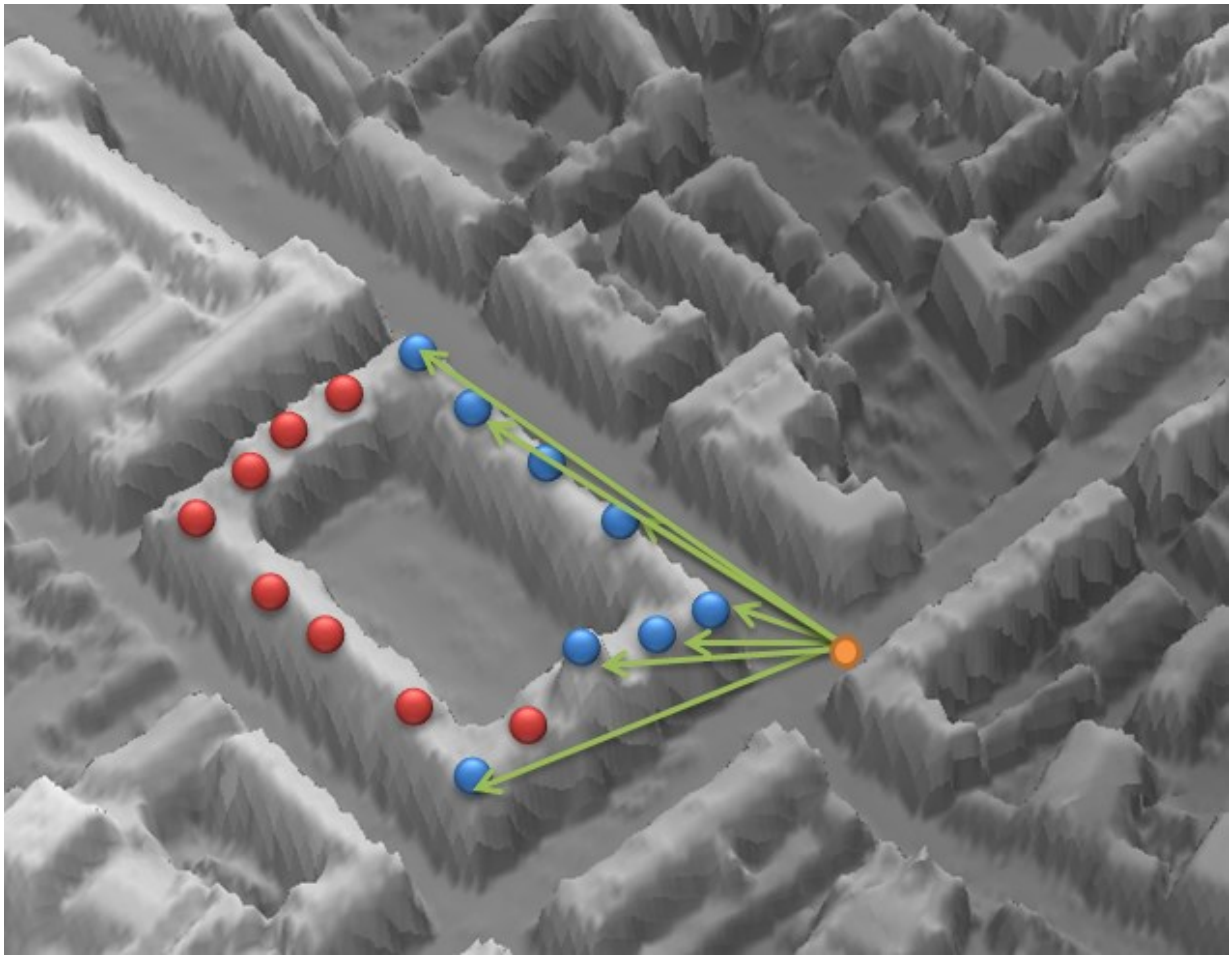

Figure 4: Ray Casting shown on LiDAR Sourced DSM
(user shown in orange, visible target shown as blue, non-visible target shown as red)

The DSM is sampled along the ray line, cast from the observer to the target, using either a vector or a raster based approach.

2.2.1.1 Vector Ray

A vector line between a designated start (observer) and end (target) point can be turned into a set of sample points at a given interval (Figure 5). The sample locations can be determined easily by calculating the difference in x and difference in y coordinates, and applying this as an offsets from the starting location until the end location is reached. This has a side effect that many sample points may occur within a single DSM raster cell, but due to the simplicity of the task the sample points can be generated very efficiently. It is possible to determine distinct cell sampling locations by adding a filtering stage to the end of the process.

2.2.1.2 Raster Ray

Using Bresenham's line algorithm raster cells can be selected in order along a path from an observer to a designated target (Figure 5). This has the advantage over a vector ray that

samples are determined at cell level, meaning each cell is only sampled a single time. However trials revealed that the extra steps and conditional statements of the code cause it to operate at a slower sample rate than the vector ray, even including the processing time to remove any duplicate cell sample locations from the vector ray output.

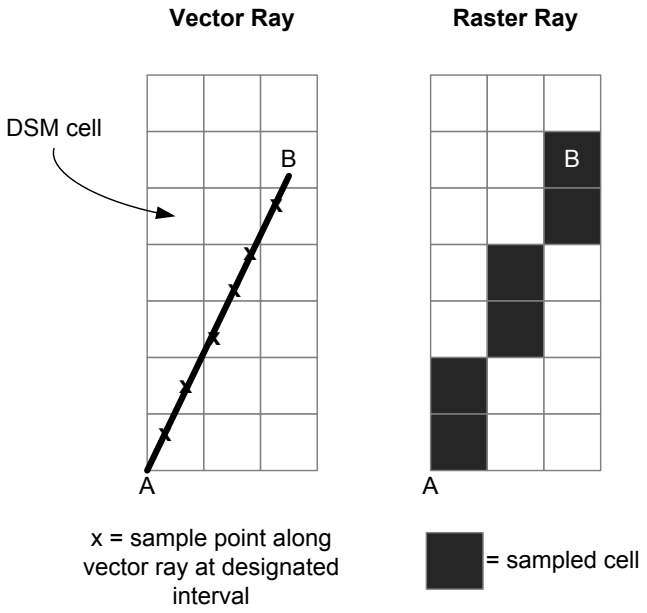

Figure 5: Ray path across DSM from A to B using Vector or Raster Method

2.2.2 Targets

The calculation of visibility on high resolution DSMs for a large number of FOIs is computationally expensive. A number of ways to reduce the processing time were investigated.

2.2.2.1 Selecting Very Important Points (VIP)

When considering the visibility of a designated FOI it is possible to restrict the search to a list of fewer targets – those targets which are considered to be the most visually important in defining the building.

There are a number of ways that the most visually significant parts of a feature can be rapidly identified, and thereby reduce the number of points which need to be scanned. These include producing a Triangulated Irregular Network (TIN) and selecting the most prominent nodes, or by producing a slope raster and selecting the cells with the steepest angles (eg above 40 degrees), as shown in Figure 6. The VIP targets for each FOI were pre-calculated and stored with each FOI.

Original Digital Surface Model Slope Raster Nodes from TIN

Figure 6: Reducing Target Points per FOI

While this approach improves performance by reducing target numbers, it raises a question over the reliabililty of the visibility result, especially when an FOI is reported as not being visible as this may be a result of the target selection process, and the user may see part of the building that is not reflected in the chosen set of VIPs.

2.2.2.2 Filtering by Aspect

An alternative approach is to calculate the slope aspect for each cell, which describes the direction in which that cell faces (eg North, East). These values may then be used to dynamically select which cells can be removed from the target list for an FOI, based on the user's location.

For example Figure 7 shows a pre-computed map of slope aspect, based on the DSM. For an observer based south of the Old College the visibility engine would not need to consider targets places on the cells with a North aspect, thereby limiting the number of rays required.

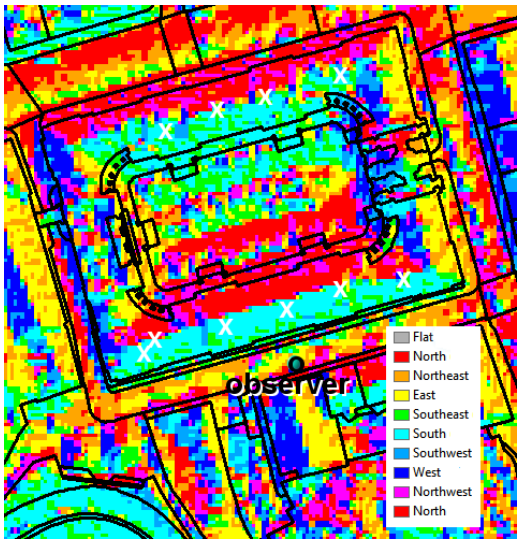

Figure 7: Aspect Based Target Filtering

This approach generally halves the number of targets which need to be scanned.

2.2.2.3 Can See any Part of a Feature – Boolean Response

A further reduction can be made if only the Boolean visibility of a feature is required (can see, cannot see), then as soon as any target on the designated feature has been calculated as visible the rays to all other targets for that FOI are not required, and the next feature may be scanned. This approach gives a big performance boost but yields far less detailed results on FOI visibility.

2.3 Ray Sweep Viewshed

Another approach is to calculate the visibility of each cell within a designated distance using a radial sweep around a location, and then to correlate the results from that against city entities (see SpaceBook D3.3.2), such as buildings, roads, and parks. This can be conducted either from the observer looking out across the city, or from each FOI. Figure 8 shows these two approaches whereby in (Figure 8A) the visibility of the city is calculated from a pedestrian accessible location, in this case on the road at location X. The visibility of all cells within a designated distance is then calculated, and either stored as a set or used to produce summary statistics based on defined zones. For example building A is in view with 7 visible cells, while building B is completely hidden from view. The process needs to be repeated for each location that the pedestrian can access (eg the cells of roads r1,r2,r3,r4). The alternative approach (Figure 8B) works from calculating the view of the city from targets placed on a FOI (in this case building B). The view of the city from each target is calculated to define the visible regions, and these may be summed to define all locations from which the building may be viewed. When the observer enters one of these locations the building is considered to be in view.

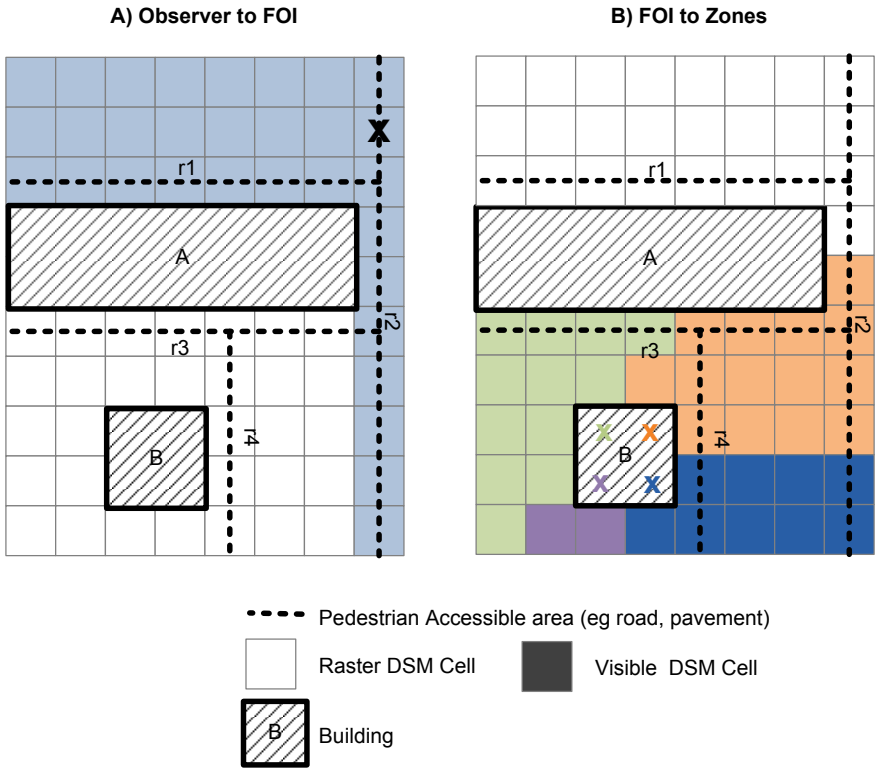

Figure 8: Ray Sweep Viewshed

The advantage of running the calculations from each FOI is that the computational effort is focused directly on knowing which parts of the city can see objects considered to be worth knowing about, and from only a few sample locations (eg the VIPs on an FOI) the visibility across the whole cityscape for those features will be known. The output can be stored with each FOI, so that any time an observer enters an associated cell then the FOI will be considered visible. This lends itself well to datasets where the set of FOIs is limited, and was used in the EARS system (Bartie and Mackaness 2006).

The alternative approach is to calculate from each pedestrian accessible location the visible parts of the entire city. This also means that the designated FOI list can be defined at any time, including redefining boundaries. However there are many millions of locations at a 1 metre scale that the observer may occupy, meaning this approach required either an optimised live visibility engine, or a significant preparation time to pre-calculate and store the viewshed results.

3 Implemented solutions

We have implemented two solutions which are designed to work with various levels of spatial data according to data availability. In some regions LiDAR datasets are available enabling urban viewshed calculations to be made, while in other regions this level of detail is not yet available or too costly. In those regions an OpenStreetMap dataset can be used with a Line-of-Sight method to determine visibility. The following section outlines both visibility engine approaches.

3.1 Line-of-Sight Method

The LM Visibility Engine is a module that maintains the mapping from the logical representation of the city (in terms of buildings, streets, etc.) to the algebraic representation (in terms of polygons, lines, and coordinates). This module can be called at any time to perform to perform visibility calculations to find out whether there is a free line-of-sight between two given points. The Spatial Model automatically generates its polygon representation of the city from an export from OpenStreetMap (Haklay and Weber 2008). A minimal bounding box is computed for each polygon in order to speed up visibility calculations, as it is faster to compute whether a line intersects a rectangle than an arbitrary polygon. If the Dialogue Manager needs to find out if B is visible from A, a request is sent to the Spatial Model, which first computes whether the line AB intersects any bounding rectangle in the entire city representation. If not, there is a clear line of sight from A to B. If the line intersects a bounding rectangle, a second more expensive calculation is carried out to check whether AB intersects the polygon inside the rectangle.

3.2 Viewshed Method

The ED Visibility Engine requires access to DSM and DTM surface models, and uses an egocentric sweep algorithm (as outline in Section 2.3) using the Bresenham raster ray. To ensure responsiveness the sweep algorithm was optimised to run in parallel across all available cores on the server, with each ray being assigned to a separate thread. The sweep scan covers a 360 degree region of 5000m radius around the user (Figure 9), as it was considered worthwhile to calculate the visibility of all objects both in front of and behind the observer, so that SPACEBOOK could draw attention to any interesting features (whether to the side, front or behind the user).

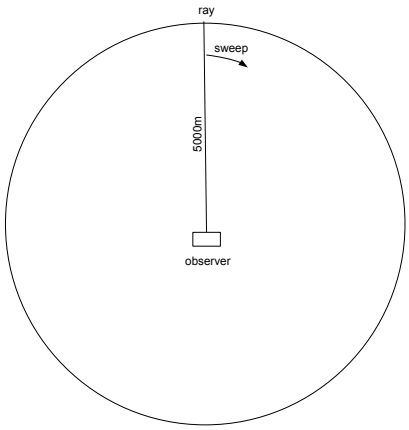

Figure 9: Sweep Viewshed approach (360 deg)

3.2.1 Ray Scan Sequence

The Bresenham's raster line algorithm was used to determine ray paths for a 5km zone around a coordinate of (0,0). To improve performance the sample points were stored in order as a predefined set of rays which could be very rapidly retrieved rather than calculated. This scan pattern may be applied to any location by offsetting the values of x and y appropriate to the observer's location. The result is a very fast set of cell sampling points for the supplied DSM, where each cell is sampled only once, which out performs the vector ray method.

The ray is fired from the DTM elevation plus an offset to accommodate the observer's height, and compared against the DSM to determine which cells are in view. This ensures that tree top views are not calculated when the user walks under vegetation canopies.

3.2.2 Vertical Extent

Traditional viewshed algorithms only record the Boolean status of cell visibility, or the angle of the incoming ray. However to truly model the feature visibility it is necessary to record the vertical extent of a feature which is visible. This means that the ray's interception elevation is required such that the extent showing above that can be recorded. To separate the objects from the topography it is necessary to reference both the DSM and DTM. When the line of sight ray intercepts at ground level then the building's visible extent is recorded as the difference between the DSM and DTM. However when the ray intercepts above ground, based on a previous close horizon value, then the building's height extent showing is the difference between the DSM and the ray's elevation. An example of this is shown in Figure 10 for a ray from the observer towards buildings A and B. Here the full vertical extent of building A is in view, while only a small portion at the top of building B can be seen. To calculate the vertical visible extent for building A it is necessary to reference both the DSM and DTM, while for building B the vertical extent is calculated from the DSM and intersection of the projected ray.

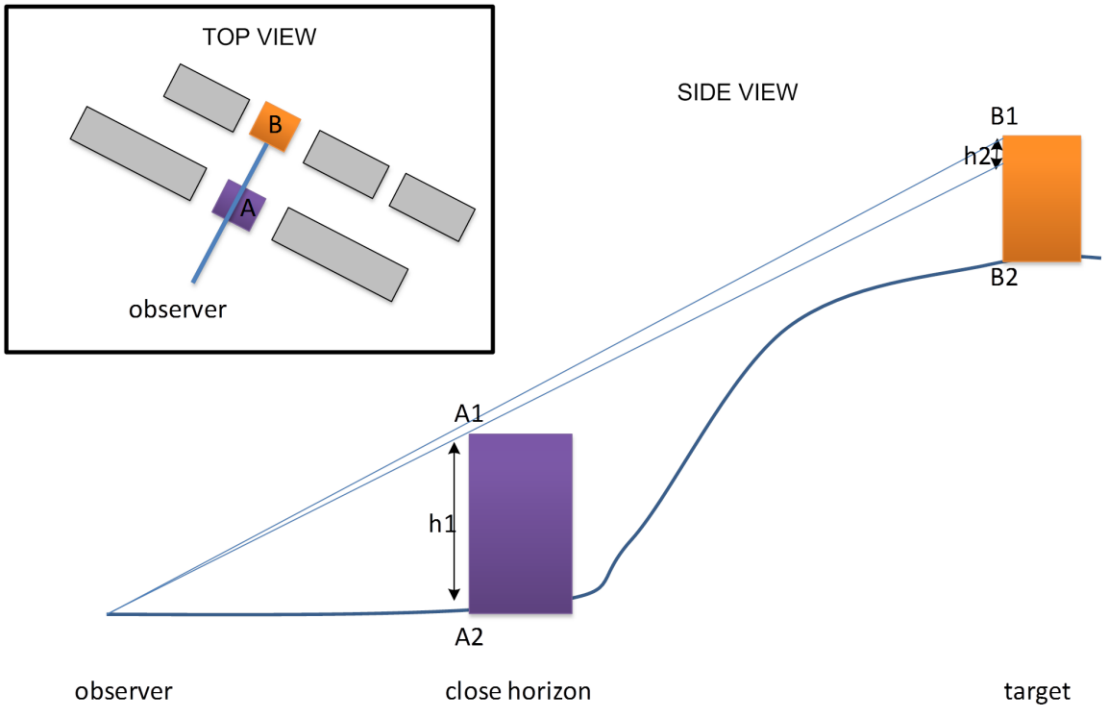

Figure 10: Visible Extent and DSM Values

3.3 Raw Visible Point Details Recorded with each Visible Cell

A range of details are recorded with each of the visible cells as they are scanned. These are:

- the distance from the observer to the cell (horizontal metres – Euclidean space)
- the absolute bearing from the observer to the cell (in degrees based on map grid)
- the location of the foreground cell – (see Close Horizon inFigure 10) as a point (x,y) in OSGB36
- the vertical extent of the target cell showing above the close horizon (in vertical metres)
- the SPACEBOOK ID of the foreground cell (close horizon cell)
- the elevation of the foreground cell (vertical metres – from DSM)
- the SPACEBOOK ID of the target cell
- the location of the cell as a point (x,y) in OSGB36
- elevation of the cell (in vertical metres – from DSM).

The foreground information is useful when identifying the projective relations between features, and may be used in later stages of SPACEBOOK to build referring expressions which describe the layout of objects in vista space. For example allowing descriptions such as “the building behind the park”, or the “tower behind the monument”.

The SPACEBOOK IDs relate to the 100 percent coverage of the land use layer, whereby for any given location within the City Model (see D3.3.2) an entity will exist. This is a polygon which defines a particular land use type, such as park, river, building, or road.

3.4 Zonal Statistics – Visibility Metrics

To describe the visibility of a FOI in a scene the individual ray outputs need to be summarised on a per feature basis. These are known as Zonal Statistics, whereby each of the 116,398 polygon zones in the Edinburgh study region are summarised for each observation location, to reveal a number of visual metrics. These include the field of view occupied, the façade area, and range of elevations as now explained.

3.4.1 Point Count

The *number of points* within the zone which have been identified as visible. This does not correlate to visible area as it does not consider the visible vertical extent, but it may be used as a way to filter out zones which have few visible ray interactions in busy scenes.

3.4.2 Distances

The *average* distance, the *minimum* distance and the *maximum* distance to visible parts of the feature are recorded. The minimum distance reflects the distance from the user to the closest visible part of the object and is probably the most useful in many instances.

3.4.3 Bounding Points

The *minimum*, *maximum,* and *average* values of *(x) and (y) coordinates*, using the OSGB36 map projection. Coordinate values may be transformed into WGS84 if required. The (x) and (y) values are calculated independently, not as point pairs.

The average value it not restricted to be within the interior of the zone (polygon boundary). The minimum and maximum values are useful for determining the bounding rectangle for the visible zone.

3.4.4 Foreground

The foreground information is summarised as the bounding box for all foreground points as *foreground min (x), min (y), max(x), max(y)* values. Also *the First and Mode of the Foreground ID* are given with each zonal summary. This allows for a rapid determination of which close horizon feature is in front of the target object. For example being able to easily describe Princes Street as being in front of Carlton Hill, knowing both are visible to the observer. The First value is given in addition to the Mode, as in some cases there may not be a Mode value. It is also possible that the most common foreground ID will be the same as the target ID, implying nothing more significant than the target occupies the foreground space.

More detailed analysis of the foreground and background space, including estimates of how much of a feature are on the skyline are possible from the raw visible point information.

3.4.5 Elevations

The *minimum* and *maximum elevations* for visible cells based on the DSM are recorded per zone. Also the *maximum* and *average vertical extents* are summarised.

These elevations differ in that the DSM values reflect the elevation of the feature including topography, while the visible extents reflect how much of the façade is visible to the observer. The visible extent is used to calculate the total visible area.

Minimum and maximum elevations may be used to form referring expressions which include phrases such as the "highest" (max DSM) or "lowest" (lowest max DSM) feature, while visible vertical extents may be used to define the "tallest" (highest max extent) or "shortest" (lowest max extent) buildings. This enable us to identify, for example, 'the short building, which on top of the hill, makes it the highest'.

This idea is clarified in the scene depicted in Figure 10 whereby an observer's view of Feature A allows for viewing of the entire front façade (h1). While the view of Feature B is more limited due to the close horizon created by Feature A, allowing only h2 to be viewed. Therefore the visible vertical extent is lower in B than it is for A.

Yet feature B occupies a higher vantage point, and therefore exist in the highest position in the user's vision along this line of sight. Feature B may therefore be described as the highest object in the scene, while feature A is the lowest.

The distinction between these two elevations is important, yet useful for building and parsing referring expressions.

3.4.6 Areas

We also record the *total façade area* visible, and the *perceived area* visible. The total façade area reflects the area visible irrespective of viewing distance, and is an indication of the size of the structure. This is included because it is known that the human mind is able to counteract distance effects when viewing recognisable objects. This is known as size constancy (Boring 1964). For example although a distant view of Scott Monument may occupy only a small portion of the view, it would be recognisable as a tall structure.

To rank items based on the amount of space occupied the perceived area is introduced, which considers the area decrease with viewing distance. This uses the formula:

$$Perceived\ Area = \frac{\text{Façade Area}}{\text{Distance}^2}, \qquad (1)$$

based on the premise that the size of an object in the retinal image is equal to the size of object divided by the distance to the object (Schlosberg 1950).

For example a clear view of a piece of A4 paper would be considered as having the same façade area yet varying perceived area, depending on the viewer's distance.

3.4.7 Angles

The bearing from the observer to the leftmost and rightmost point is calculated, giving the *leftmost bearing* and *rightmost bearing*. This also gives a *field of view* and a *central bearing* value. These are given irrespective of the observer's facing direction as absolute angles in degrees, but can be easily converted to relative angles using the observer's facing direction if required.

The field of view is calculated between widest visible points on a feature, irrespective of how much of the central part of the feature may be seen. Therefore there may be minimal correlation between the field of view occupied and the visible area, as it will depend on the foreground scene and total area obscured from view. An example of the output for a number of features in view around an observer is given in Figure 11.

Feature	Left	Right	Central Bearing	Field of View
A	8.66	50.97	29.8	42.31
B	77.46	137.43	107.44	59.97
C	172.91	235.55	204.22	62.63
D	258.70	338.66	298.68	79.96

Figure 11: Absolute Angle Summaries (degrees) for Features A,B,C,D

3.4.8 Cache System

The processing time varies depending on the openness of the location, whereby urban corridors with more limited views are calculated more rapidly as it is possible to determine rapidly that a ray is ascending into open air and not going to reach another DSM cell value. However to cater for the more demanding areas (when an observer has an expansive view of the city), and when multiple users are interrogating the system, a dynamic cache was introduced.

The cache makes use of a quadtree index, to improve search speed. When a request is made for the visibility summary results for a specified location, orientation and field of view the cache is checked, if an item is found the results are re-parsed and filtered for the supplied orientation and field of view. If a cached entry does not exist for this location then a live viewshed is carried out, and the results for 360 degrees are saved to disk. The filtered results with relative angles are returned to the client, and the item added to the quadtree index. The quadtree index performance is very good, typically with results found, loaded from disk, re-processed and filtered according to supplied orientation and field of view within 15ms.

It is possible to define when a cached result will be offered rather than calculating a new result. For example a cached result may be considered suitable if located within 3 metres of the user's location. The cache will search for the closest point from those available, but can be set to stop searching when a given distance tolerance is reached (such as a point within 1m of the user's location).

3.5 Example of Output

Figure 12: Example Output of the Visible Polygons from a defined Observation Location
(observer shown as orange point, visible polygons highlighted in blue)

The first time calculation including the zonal summary on an 8 core PC for the scenario outlined in Figure 12 took 370ms, while a second retrieval from the zonal summary cache took only 17ms.

The selected FOIs are generally around the observer, but also include items in the across roof tops and in the distance (eg Dome on Old College, National Museum of Scotland).

As outlined in D3.3.2 the visibility of the polygon areas can be linked to actual building uses, by carrying out a spatial join on the feature tables. For example the polygon 139777 is returned as visible, and a spatial join would show this to have the name "Old College", contain a war memorial, gallery, many public telephones, and be part of the University of Edinburgh.

3.6 Using the Visibility Engine with the City Model

The viewshed engine is available on a virtual machine server located in Edinburgh. Connection to the visibility engine is made through a raw Telnet connection. The query protocol is based around sending a single line XML-styled request to the server, which will reply with an XML-styled document. Each request requires an authentication token to be issued as part of the outer tag, otherwise the request will not be processed. This is introduced to limit access to the City Model, which contains copyright data licenced and made available for research use by Spacebook team members only.

Five different request question types are supported, which are:

1) The visibility from the Observer to a specified Target Point (given in WGS84 or OSGB36 coordinates). The request outer tag is <SecureToken_vispt/> This makes use of a hopping optimisation (Bartie and Mackaness 2012) whereby staggered samples are taken at 26 metre intervals from the observer to the target, terminating if the target is found to be out of view. During stress testing this increased overall performance by a factor of eight. The return is a boolean value (true, false).

Example:

```
<SecureToken_vispt><wgs84>55.949864,-3.1901</wgs84><tarpt>55.95,-
3.1882</tarpt></SecureToken_vispt>
```

Notes:

- The observer's location is defined using either geographic coordinates <wgs84>lat,lng</wgs84> or projected coordinates <osgb36>x,y</osgb36>
- The server will handle the conversion of coordinate system
- Do check that the coordinates are supplied in the correct order: geographic(lat,lng) or projected(x,y). Note that lat <=> y, lng <=> x
- x,y,lat,lng are of type float (eg 3.141)
- the target point is always processed using the same coordinate system, so you only pass across the location of the target using the form <tarpt>lat,lng</tarpt> or <tarpt>x,y</tarpt>

2) The visibility from the Observer to all City Model entities in view around them (given in WGS84 or OSGB36)

Example:

```
<SecureToken_vis><osgb36>325229.9,673847.0</osgb36><orientation_fov>94.67,140</orientation_fov
><rtype>all</rtype></SecureToken_vis>
```

Notes:

- The observer's location is defined using either geographic coordinates <wgs84>lat,lng</wgs84> or projected coordinates <osgb36>x,y</osgb36>
- The server will handle the conversion of coordinate system for you
- Do make sure that you pass the coordinates in the correct order though: geographic(lat,lng) or projected(x,y). Note that lat <=> y, lng <=> x
- x,y,lat,lng are of type float
- an *optional* <objidlist>[n1][n2][etc]</objidlist> may be included to reduce the resultset to only those visibile items which are included in the list. Each objid relate to the polygon object id. To determine the feature's uses a further SQL query is required (e.g. to correlate the object to the IDs used in the table 'isNamed'). The

object ID list values are encased in square brackets []. If you do not wish to filter by any object IDs then remove this tag from the query.

- an *optional* <orientation_fov>a,b</orientation_fov> tag may be provided. This consists of two parts the orientation (a) of type float, and a field of view (b) of type integer. Orientation is the user's facing direction, and the field of view is the width of their view, typically 180-200 degrees. However if desired this may be limited to any value from 1-360 inclusive. This may be used to filter the results. If no <orientation_fov> tag is supplied then all visible entities for the full 360 degree view are returned
- If an orientation tag is supplied then the resulting bearings (leftmost visible point, rightmost visible point, average bearing) are reported both as absolute and relative angles degrees (an extra set of tags are included in the XML returned).
- All angles are given using clockwise notation
- An <rtype> tag defines what is returned in the resultset. It currently supports the values of ALL, MIN or ID. ALL will send back all the visibility metrics, MIN returns a reduced set of the most important visibility metrics, while ID limits the returned information payload to only the City Model IDs in view.

3) It's possible to determine if a building is visible on the skyline or not by checking the lines of sight beyond the feature to see if they intersect a more distant object or not. Those which have no lines of sight intersecting a more distant object are considered to be on the skyline. The user position and the target polygon are specified. The Boolean result is returned.

Example:
<SecureToken_skyline><osgb36>325369.7,673815.5</osgb36><polygonid>130276</polygonid></SecureToken_skyline>

4) The visibility engine may be used to sample points along the shortest path and returns a summary of the landmarks along a route. The results returned are divided into two sections which are the statistics on number of times each landmark has been seen along the entire route, with details on where the landmark was visible. The second table returned details for each route segment (node to node) the most frequent and rarest landmark sightings.

Figure 13 shows an example of the output for a route (marked in blue) from Princes Street to the High Street (travelling east and south). The buildings visible along this route are highlighted in green, and those considered important are highlighted in yellow, orange and red. The calculations considers the viewing direction as the user progresses along the route, the junction where a turn is required, and the visibility of the features.

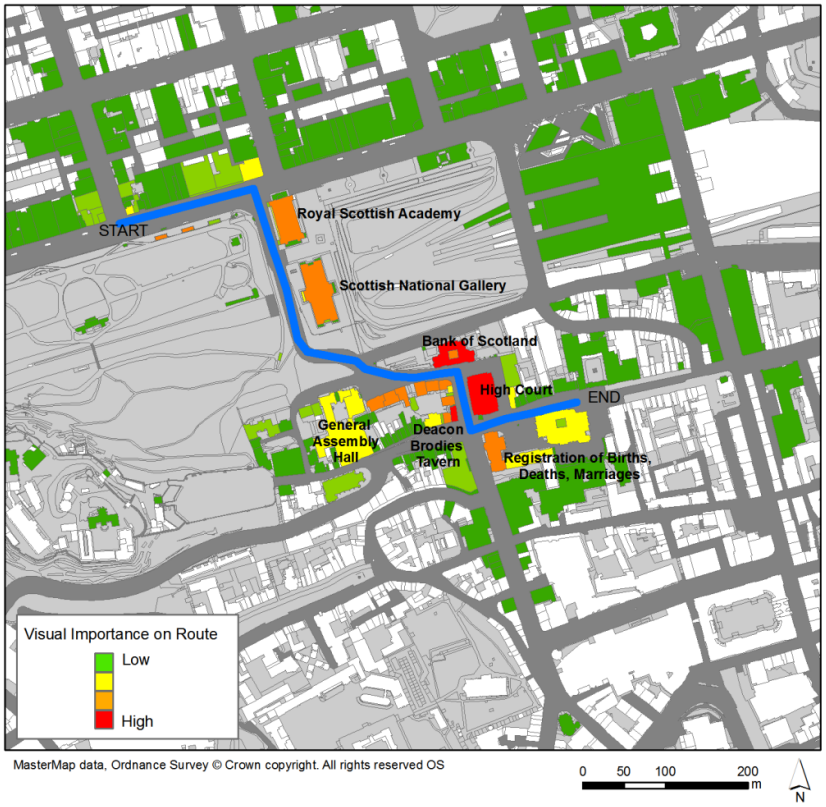

Figure 13: Important Landmarks along a Route

Example:

```
<SecureToken_landmark_route><startnode>2008199</startnode><endnode>2009768</endnode></SecureToken_landmark_route>
```

5) A SQL select statement to the City Model can be made through this service. This is effectively a passthru SQL but limited to Select statements only.

Example:

```
<SecureToken_select>id, name from isnamed where id in (1,2,3,4,5);</SecureToken_select>
```

- An <_select> tag will be passed on directly to the City Model, and the results passed back. These SQL requests are limited to SELECT statements only, and begin with the list of field names (ie the 'select' word is not repeated).

Example queries:

```
<SecureToken_vis><osgb36>326012.8,673397.8</osgb36><orientation_fov>23.1,200</orientation_fov><rtype>all</rtype></SecureToken_vis>

<SecureToken_vis><osgb36>326012.8,673320.1</osgb36><rtype>id</rtype><orientation_fov>270,200</orientation_fov><objidlist>[139777][12345][954547]</objidlist></SecureToken_vis>

<SecureToken_vispt><wgs84>55.949864,-3.1901</wgs84><tarpt>55.95,3.1882</tarpt></SecureToken_vispt>
```

Quitting a client session:

To terminate a client session enter 'quit' (without quotes).

If a client session does not send requests for a 45 minute period the connection will be terminated by the server.

3.7 Examples of How the Output May be Used

This visibility engine provides the output for other components of SpaceBook to assist in information filtering, choice selection, or to generate sentences. Figure 12 outlines the supporting roles of the visibility engine (highlighted by black rectangles).

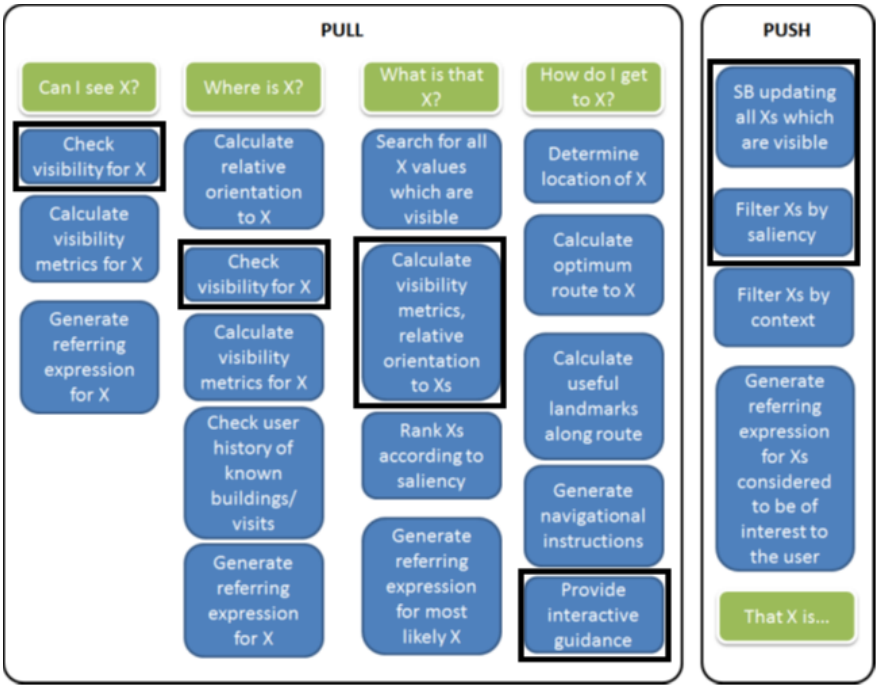

Figure 1: Integrated Role of the Visibility Engine throughout SpaceBook Tasks

The sorts of sentences which may be generate based on this visibility engine output include:

- "The Royal Bank of Scotland is clearly visible in front of you"
- "The National Museum of Scotland can be seen behind you"
- "The supermarket is opposite the Old College" *(use of 'opposite' requires items in relationship to be visible)*
- "Scott Monument is immediately in front of you, with Edinburgh Castle in the background just to the right"
- "From here you can see the four dominant towers of George Heriots School with the meadows in the far distance, to the south"
- "Nelson Monument can be seen on Carlton Hill straight in front of you"
- "The tower on the right is Scott Monument, the one in the middle is part of the Balmoral hotel, the one on the left is Nelson Monument"
- "Head towards The Hub which is in view on your right now"
- "You can see the cross roads of the High Street and the Bridges"
- "The Institute of Geosciences is just on your right, but in the far distance you can see the top of St Giles Cathedral"
- "You should be able to make out the dome of the Talbot Rice Gallery across the rooftops"
- "The church to your right is St Catherine's with the larger Standard Life Building immediately behind it"
- "If you head to the junction, you'll be able to see the Pear Tree pub on your left".

4 Conclusion and Future Work

Vista modelling extends the capabilities of LBS to more closely model the user's real world experience. It can be used in forming natural language descriptions of entities, landmark identification, and in building more natural succinct way-finding instructions. Where detailed surface model datasets are available it is possible to return a range of object level visibility metrics using zonal statistics against the land cover polygon dataset. These enable the system to not only determine what is in view, but also to model how much of the view is filled by an object, and the enormity of the visible portion of that object. Where spatial data are restricted to OpenStreetMap it is possible to use the line-of-sight approach to determine the visibility of nominated targets.

There are a number of improvements which can be made to the viewshed engine. Notably the handling of vegetation and underpasses is restricted due to the 2.5D nature of the DSM dataset. To better model under canopy and bridge views the base heights and vegetation opacity values need to be collected from oblique angles to measure clearance heights. It would also be prudent to look at methods to capture façade interest, such as colour information and architectural detailing. This would assist in determining the more visually different objects in a scene beyond what is currently available. However in its current form the visibility engines have proven to be useful additions to LBS and enriched the user experience of mobile interaction with spatial datasets in the contexts of wayfinding and exploration.

5 References

Bartie, P. and W. Mackaness (2012). Optimal Sampling Strategies for Line-Of-Sight Calculations in Urban Regions. GISRUK2012, Lancaster, UK.

Bartie, P. J. and W. A. Mackaness (2006). "Development of a speech-based augmented reality system to support exploration of cityscape." Transactions in GIS **10**(1): 63-86.

Bartie, P. J., F. Reitsma, et al. (2010). "Advancing visibility modelling algorithms for urban environments." Computers Environment and Urban Systems **34**(6): 518-531.

Benedikt, M. L. (1979). "To take hold of space: isovists and isovist fields." Environment and Planning B **6**(1): 47-65.

Boring, E. (1964). "Size constancy in a picture." American Journal of Psychology **77**: 494-498.

Chisholm, G. (2011). Landmark and Vantage Point Identification: Using Visibility Analysis of Urban Models for Location Based Services. Institute of Geosciences. Edinburgh, The University of Edinburgh. **MSc in GIS**.

De Floriani, L. and P. Magillo (1994). "Visibility algorithms on triangulated digital terrain models." International Journal of Geographical Information Systems **8**(1): 13-41.

De Floriani, L., P. Magillo, et al. (2000). "VARIANT: A system for Terrain modeling at variable resolution." GeoInformatica **4**(3): 287-315.

Fisher, P. F. (1993). "Algorithm and implementation uncertainty in viewshed analysis." International Journal of Geographical Information Science **7**(4): 331-347.

Fisher, P. F. (1994). Probable and fuzzy models of the viewshed operation. Innovations in GIS 1. M. F. Worboys. London, UK, Taylor and Francis**:** 161–175.

Fisher, P. F. (1995). "An exploration of probable viewsheds in landscape planning." Environment and Planning B: Planning and Design **22**: 527-546.

Haklay, M. and P. Weber (2008). "Openstreetmap: User-generated street maps." Pervasive Computing, IEEE **7**(4): 12-18.

Kidner, D., A. Sparkes, et al. (1999). "GIS and wind farm planning." Geographical Information and Planning: 203-223.

Lee, J. and D. Stucky (1998). "On applying viewshed analysis for determining least-cost paths on digital elevation models." International Journal of Geographical Information Science **12**(8): 891-905.

Lynch, K. (1976). Managing the sense of a region, MIT Press Cambridge.

Montello, D. (1993). "Scale and multiple psychologies of space." Spatial Information Theory A Theoretical Basis for GIS: 312-321.

Morello, E. and C. Ratti (2009). "A digital image of the city: 3D isovists in Lynch's urban analysis." Environment and Planning B **36**: 837-853.

Rana, S. and J. Morley (2002) "Optimising visibility analyses using topographic features on the terrain." Centre for Advanced Spatial Analysis, University College London, London, UK.

Schlosberg, H. (1950). "A note on depth perception, size constancy, and related topics." Psychological Review **57**(5): 314-317.

Tandy, C. R. V. (1967). The isovist method of landscape survey. Symposium: Methods of Landscape Analysis, London, England, Landscape Research Group.

Ying, S., L. Li, et al. (2006). Incremental terrain visibility analysis. Proceedings of SPIE - The International Society for Optical Engineering, Wuhan, SPIE.