2010-12-01

Mobile Visibility Querying for LBS

James Carswell  
Technological University Dublin, jcarswell@dit.ie

Keith Gardiner  
Dublin Institute of Technology

Junjun Jin  
Dublin Institute of Technology

Follow this and additional works at: https://arrow.dit.ie/dmcart

Part of the Databases and Information Systems Commons, and the Graphics and Human Computer Interfaces Commons

Recommended Citation

doi:10.1111/j.1467-9671.2010.01230.x

This Article is brought to you for free and open access by the Digital Media Centre at ARROW@TU Dublin. It has been accepted for inclusion in Articles by an authorized administrator of ARROW@TU Dublin. For more information, please contact yvonne.desmond@dit.ie, arrow.admin@dit.ie, brian.widdis@dit.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License
Research Article

Mobile Visibility Querying for LBS

James D. Carswell
Digital Media Centre
Dublin Institute of Technology

Keith Gardiner
Digital Media Centre
Dublin Institute of Technology

Junjun Yin
Digital Media Centre
Dublin Institute of Technology

Abstract
This article describes research carried out in the area of mobile spatial interaction (MSI) and the development of a 3D mobile version of a 2D web-based directional query processor. The TellMe application integrates location (from GPS, GSM, WiFi) and orientation (from magnetometer/accelerometer) sensor technologies into an enhanced spatial query processing module capable of exploiting a mobile device’s position and orientation for querying real-world spatial datasets. This article outlines our technique for combining these technologies and the architecture needed to deploy them on a sensor enabled smartphone (i.e. Nokia Navigator 6210). With all these sensor technologies now available on off-the-shelf devices, it is possible to employ a mobile query system that can work effectively in any environment using location and orientation as primary parameters for directional queries. Novel approaches for determining a user’s visible query space in three dimensions based on their line-of-sight (ego-visibility) are investigated to provide for “hidden query removal” functionality. This article presents demonstrable results of a mobile application that is location, direction, and orientation aware, and that retrieves database objects and attributes (e.g. buildings, points-of-interest, etc.) by simply pointing, or “looking”, at them with a mobile phone.

1 Introduction

This article focuses on directional querying and the development of algorithms for mobile 2D and 3D spatial data interaction. In conjunction with location, an orientation module provides angular parameters to a spatial query processor, making it possible to...
perform directional queries on a mobile device in a real world environment. The fundamental requirements for this type of service interaction are location, direction and orientation.

Some typical applications from current literature enable a user to point a custom-built mobile device at a building and, using position and direction, determine the building’s address/identity (Simon and Fröhlich 2007a). This requires both accurate location and orientation as query parameters and the sw components that gather and provide this input data in our case are the LocateMe and DirectMe modules. The locator component, or LocateMe module, is by design an open source, network independent mobile location determination application that can utilize GPS, Wi-Fi and GSM beacon information, or any combination of them, to trilaterate for cellphone position estimates (Kilfeather et al. 2007, Rooney et al. 2007a).

The orientation component, or DirectMe module, is the primary focus of this article. It uses data from a number of sensors including a GPS sensor, a magnetometer sensor (digital compass) and an accelerometer sensor (tilt sensor). Using these data, the TellMe application formulates directional queries that are performed in a spatial database to determine if any spatial interaction exists between the query “window” and any of the buildings or Point-of-Interest (PoI) objects in our 3D model testbed. The shape of this query window takes on a variety of 2D and 3D forms from a simple ray, to a polygon, to a volume (see Figure 14). The query results of this interaction are subsequently returned to the user in the form of building address/details plus web-links to more information (e.g. classroom timetables, office hours, etc.).

Essentially, the TellMe application provides a framework for processing two dimensional queries in an open, non-directional query-space (e.g. standard range query), but with 2D hidden query removal. This allows for limiting query results to only what is actually visible to the user and subsequently addresses some of the major information overload issues in the field of MSI research within cityscape environments.

In relation to sensor data quality (e.g. signal-to-noise ratio), tests to compare the data gathered from a higher quality external sensor pack to that of integrated phone sensors are ongoing. These findings will ultimately help determine the suitability/accuracy of current mobile device technology for providing meaningful mobile spatial interaction in LBS applications. Following this, our attention shifts to exploring three dimensional visibility with the development of an “Ego-Visibility” query processor that further confines the query space to simulate a user’s view field-of-view frustum. Results from this will enable us to incorporate full 3D hidden query removal functionality.

The remainder of the article is organized as follows. Section 2 describes some related work in the area of directional querying. Section 3 outlines some methods used in current GIS to perform these queries using visibility analysis. Section 4 describes the design of our TellMe application which includes the LocateMe and DirectMe modules, the TellMe Server, and other hardware considerations. Section 5 demonstrates our directional query approach on a Nokia Navigator cell phone with five different types of possible queries, and Section 6 concludes with a summary and future work.

2 Related Work

There have been a number of applications recently that utilize compasses and tilt sensors attached to custom built mobile devices. Some early work by Wilson and Pham (2003)
and Wilson and Shafer (2003) introduced the idea of an XWand pointing device that queries an intelligent environment using a variety of sensors in a combination that supports pointing and gesture recognition inputs. A museum guide is described in Chan et al. (2005) to help visitors quickly locate exhibits using orientation aware handheld devices. In Baillie et al. (2005), an MSI method is presented where a user points a mobile device at a building to view a virtual representation of it at different times in the past. In Essl and Rohs (2007), an approach to use sensor data for turning the device into a musical instrument using shaking and sweeping gestures is reported. The Shoogle project (Williamson et al. 2007) aims to use inertial sensing to provide an eyes-free vibrotactile display that mimics objects rattling around inside the device for a rich multimodal interaction without any visual attention.

The majority of current research focuses on providing enhanced navigation capabilities in the area of MSI. The Point-to-Discover GeoWand (Simon and Fröhlich 2007a, 2008) is a system and application framework for orientation-aware location-based mobile services. This application demonstrates how their custom-built device can be used as a pointing tool to display web-based information about bars and restaurants on the device. A very similar approach is taken by Intelligent Spatial Technologies with the iPointer application (see http://www.i-spatialtech.com/ipointer.html for additional details). It is based on an augmented reality engine and a thin client API that provides a local mobile search based on GPS and an eCompass to deliver content, such as pictures, menus, and audio overviews streamed back to the user’s phone. A rather different approach is taken by Robinson et al. (2008) and their Point-to-GeoBlog application, where users are able to select landmarks by using the point and tilt functionality of yet another custom built device. No content is provided up front but is later when the user logs onto a computer with more visually adequate display capabilities.

A detailed usability study in Robinson et al. (2008) reports that the most user friendly approach was to provide a simple point and tilt interface over a visual map and also a preference for remote tagging that allows users to select landmarks beyond their line-of-sight. These results confirmed a comparative outdoor study by Fröhlich et al. (2006) that tested conceptual designs for four interaction areas considered important for spatial information applications. The information Pull technique, where the user decides what information to view and has control over it, was the most intuitive and preferred data interaction approach tested by users (Persson et al. 2002, Strachan et al. 2007, Strachan and Murray-Smith 2009). The Push technique, where all information is automatically presented to the user, is not easily managed on the variously display (and memory and processor and network connection) constrained mobile devices.

An alternative approach that combats problems related to information overload is to restrict the search space based on certain criteria. In Gardiner and Carswell (2003), the approach is to restrict the search space to a user’s field-of-view (FoV) using the concept of an observer’s two dimensional query frustum to determine what the user can actually see from their location in the environment. In contrast, the use of the Point-to-Discover block model and visibility computation algorithm in (Simon and Fröhlich 2007a) uses a rather different approach to determine what buildings the observer can actually see from a single vantage point. In (Simon and Fröhlich 2007b), this idea is extended with the development of a local visibility model that introduces the concept of “billboards” as a mechanism to identify what buildings a user can actually see. Determining this type of egocentric visibility is a primary focus of this
article. We consider only possibilities in relation to what a user can physically see from their current position by performing visibility analysis calculations. Utilizing this approach to visibility querying on today’s “off-the-shelf” mobile phones is a new concept and a key aspect of our research.

3 Visibility Analysis

There are a number of methods used in current research for performing line-of-sight queries. This section gives an overview of the technologies and methods used by a number of them, and describes some emerging possibilities.

3.1 Ray Tracing

The ray tracing process is fundamental in this area and works by simulating the light that travels within a given space. In many applications, this process is performed backwards to determine the visibility from one or multiple points (e.g. user positions), which instead of perceiving the light emitted from the objects themselves (e.g. buildings or other infrastructure), the ray is transmitted from a given user position outwards in all directions. By retrieving all the intersections between any generated ray with objects, a polygon in a 2D environment and a volume in a 3D environment can be constructed to represent the visibility query space from a view point (Figure 1). This approach is termed Line-of-Sight (LoS) in Geographic Information Systems (GIS).

3.2 Line-of-Sight (LoS)

The LoS function determines the visibility from the observer’s position to a target point considering DTM fluctuations and other obstructions. The path from the view-
point to the target point is one of the rays emitted from the viewpoint. In effect, some of the projected rays are truncated where they intersect with obstacles (e.g. building blocks, hills, etc.) while on route to the target. This collection of interaction points forms a convex polygon or volume representing the visibility area of a specific viewpoint. In a 2D environment, where the elevation of the objects is not taken into account, the line-of-sight calculation can be simplified by recording all the intersection points between the rays and obstacles surrounding it in the horizontal plane (Figure 2).

In a three dimensional space, however, the ray is not just projected in the horizontal plane but also in the vertical plane using the tilt angle in 360 degrees. In space syntax terminology, this particular shape is referred to as an Isovist (Benedikt 1979). A 2D Isovist is a visibility polygon comprised of the set of all points visible from a given vantage point in space within an environment. In Jiang and Liu (2009) this concept has been extended recently with the automatic generation of an axial map which in turn can be used to generate the Isovist.

3.3 Viewshed Analysis

A similar function to Isovist is available in GIS and is known as Viewshed Analysis. It adopts the same approach as line-of-sight analysis; however instead of examining the single path from the viewpoint to the target point, a beam of multiple rays is generated from the viewpoint in all directions in the horizontal plane. Concurrently, a beam of rays is generated vertically along each horizontal ray path and the azimuth of each ray is taken into consideration within a certain range. For instance, in Figure 3 the viewshed is based on the viewpoint and the range combined with the horizontal, vertical, and tilt angles, where the actual visible space is a volume excluding any section of this 3D query shape that intersects with a building or other object.

Adopting such an approach to measure the visible space within a certain radius can significantly reduce the number of calculations required to determine the possible intersection shape. This strategy is similar to radar scanning the environment with a limited signal distance (strength) and is essentially an extreme case of the radial line-of-sight approach.
3.4 Threat Dome Analysis

To estimate the visibility shape in all directions from the viewpoint using viewshed analysis, the azimuth in the horizontal plane and the view angle in the vertical plane can be extended to a full range, which generates rays between 0 and 360 degrees to form a dome shaped viewshed. An example of this idea is the Threat Dome from military applications, which is used for determining all possible locations in space that are visible to/from a given position (ESRI 2009, Skyline 2009).

In contrast to viewshed analysis, capturing the visible volume from a point taking into account all existing obstacles (buildings, etc.) in the environment can be very computation intensive. The threat dome approach specifies a radius defining the view distance from a point and hence the rays emitted from the viewpoint form a sphere. By determining the intersections of the rays with obstacles at different levels of elevation within the sphere, the actual visible space within this specified radius can be determined (Figure 4). Decreasing the distance between the levels of calculation will improve the granularity/accuracy of the query shape.
4 TellMe

Our mobile TellMe application requests information from two modules, namely the LocateMe and DirectMe modules, to perform directional queries using the TellMe Server against various spatial data sets. Two main approaches were investigated in relation to implementing the TellMe Server.

One approach was to host the spatial dataset on the mobile device itself. In this case the data retrieved from LocateMe and DirectMe is used to perform spatial queries locally on the mobile device. The results of the queries are displayed to the user by the TellMe Mobile application. Because of limited memory and the processing speeds required to perform spatial queries on large geographic data sets, this type of architecture was deemed unsuitable for unconstrained real-time spatial querying.

The alternative approach (i.e. thin client/server approach) is to request the required data from LocateMe and DirectMe and perform spatial queries on a dataset hosted on an external server. The parameters are passed to a web application, which in turn carries out the queries on a spatial database to determine any spatial interaction between the user’s location and orientation with the dataset. The results are subsequently displayed to the user in a web browser on the mobile device. This architecture is illustrated in Figure 5.

**Figure 5** Overall TellMe architecture
This approach of using a web server to query and display the returned data in a web browser follows current trends because of the increasing complexity associated with developing applications to run on many different devices (W3C 2008). To avoid developing a different application for each mobile operating system, a more attractive solution is to develop applications using a web scripting language capable of producing highly interactive applications that can perform the same functions as desktop applications without the concern about underlying operating system functions or restrictions.

Taking this approach a step further, it is now becoming more acceptable to allow the browser access to more different types of data as well. For example, using a set of JavaScript APIs, it is possible to access some of the hardware on a mobile device directly from the browser. This is one of the newest features of Google Gears where the Geolocation API allows an application access to the GPS hardware on the device (Google 2008). Using this approach, if all sensor data were made available to a web application, it would be possible to eliminate the need for an on-device application altogether, essentially making the TellMe application entirely web-browser based.

4.1 LocateMe

The LocateMe module is used in conjunction with the DirectMe module to gather information about a user’s current position and orientation. These data are then used to execute directional queries against various data sets both internal and external to the phone. The LocateMe module is based on a hybrid positioning calculation that utilizes GSM, Wi-Fi and Bluetooth radio signals in addition to GPS to determine location. A more comprehensive description of this technology can be found in Rooney et al. (2007b).

4.2 DirectMe

The DirectMe module is one of two modules that provide data to the TellMe application. Its function is to determine the direction that the mobile device is currently pointing by using a digital compass and tilt sensors. These data are then collected on request from the TellMe application and synchronized with the location data coming from the LocateMe module. The architecture of the DirectMe module is similar to the LocateMe module where each sensor has a native hardware spotter that relays data to a higher-level component that synchronizes data coming from each spotter. This architecture is illustrated in Figure 6.

This type of architecture is required to overcome the frustrating restrictions imposed on would-be developer access to various mobile device hardware components (e.g. compass and tilt sensors) found within some proprietary operating system APIs.

4.3 Mobile Device Hardware/Sensors

There are currently three main mobile platforms that contain the required sensor hardware for our TellMe system. The Nokia Navigator 6210 has a digital compass and tilt sensors and the API providing access to these sensors has just (Q4, 2009) been back ported from Symbian S60 5th Edition to Symbian S60 3rd Edition FP2 (Nokia 2008). Using this version of the API, the DirectMe module can ascertain the current heading and orientation of the device. However, there have been recent reports that the
Another hardware option is the HTC Dream (Google phone), also with integrated digital compass and accelerometers. This device runs the Android operating system from Google and at time of writing is not yet available for sale in Ireland (HTC 2008). The most recent possibility in terms of hardware suitability is the Apple iPhone 3GS (2009). In addition to providing WiFi and accelerometer access there is now a compass integrated as well.

Considering today’s limited number of off-the-shelf mobile devices available for acquiring orientation type data, another option is to use external sensors packs such as the SHAKE. The SHAKE SK7 is an external sensor pack with digital compass, accelerometer, and tactile feedback, among others. This sensor pack communicates with a recording device (e.g. laptop, cellphone) via Bluetooth. Unlike current cellphone sensors, the SHAKE device has better quality sensors and a number of filters on board to reduce any noise introduced by, for example, the mobile phone’s antenna. However, following a review of these options, it was decided that the DirectMe module would collect data from a number of integrated MEMS (Micro Electro-Mechanical Systems) sensors on a
Nokia Navigator 6210 (Figure 7) instead of the SHAKE to demonstrate the state-of-the-art application development potential of today’s mobile devices.

4.4 TellMe Server

The TellMe Server is essentially a spatial application server that performs all the complex spatial queries in the TellMe system. It is responsible for communicating with the TellMe Mobile application, which collects data (including location, direction and orientation) from the LocateMe and DirectMe modules on the mobile device. These data are communicated wirelessly to the spatial application server and used to perform spatial queries against an Oracle Spatial database.

The TellMe Server is based on ESRIs ArcGIS Server platform and is used to perform the complex queries required for determining the mobile spatial interaction between a user’s line-of-sight and the 3D database. This platform provides server extensions for many of the traditional spatial query functions found in most GIS. Using these extensions, it is now possible to perform these types of spatial queries in a mobile context that were previously only possible in a desktop setting. Another important reason for this choice of server is based around the issue of scalability; using this software ensures that the TellMe system is scalable to manage a large number of users efficiently.

4.5 3D Database

To perform directional queries, a 2D spatial database is the minimum requirement for supporting extensive feature types, indexing techniques, and for performing 2D spatial queries effectively to identify objects (e.g. buildings, PoIs) that intersect with a direction vector. Some types of queries possible in 2D are illustrated in Figures 10–12.

However, to perform 3D spatial queries this process becomes somewhat more complex. In comparison, a 3D spatial query should be able, for example, not only to...
identify which building a direction vector is intersecting with but also the floor (or window, door, etc.) of the building it is directed at. This means a true 3D database should support three-dimensional data types such as point, line, surface and volume in its geometric data model, be capable of indexing the data and must also offer functions and operations embedded in its spatial query language that can operate on these data types (Schön et al. 2009). In fact, these requirements significantly reduce the number of database options of enterprise level capabilities that are available to us – there are two main options here.

4.5.1 Oracle 11g

Beginning with the 11g version of Oracle, it is now possible to store, index and query 3D spatial objects using the sdo_geometry data type. Using this datatype, it is now possible to store point, line, polygon, polygon with hole, and collection data types in 2D and 3D (Schön et al. 2009). An example of the types of data that can be stored is illustrated in Figure 8b.

4.5.2 ESRI ArcGIS

To support this rising trend in 3D spatial data storage, ESRI developed a native volumetric geometry feature type called the Multipatch feature supported by its geodatabase models. The Multipatch is treated like any other geometry type in the database and is constructed of triangle strips and fans to define object boundaries using triangular faces (Figure 8a).

In our case, we chose Oracle 11g for its 3D forms: simple solid, composite surface, composite solid and collection to represent our cityscape data. Our 3D data is based on Ordinance Survey Ireland’s (OSi) 2D vector footprint data overlaid on a 10 m DTM and extruded using height values taken from airborne LiDAR scans (see http://www.osi.ie/ for additional details). The result is a building block model of the Dublin city centre. This

![Figure 8](image-url)
data are spatially indexed and queried using a 3D query window generated by the sensor data collected from the mobile device. All attribute data (building name, class timetables, etc.) presented to the user are also stored in the Oracle database. To perform 3D queries, the set of operators is restricted to SDO_Filter, SDO_Anyinteract, SDO_Within_Distance, and SDO_NN (nearest neighbour) using the Geographic-3D coordinate system. An example of a typical 3D frustum query shape is shown in Figure 9.

5 Demonstration on Nokia Navigator 6210

In this section, the different types of queries that are performed by the TellMe application are discussed, all of which are based on the user’s egocentric point-of-view. Egocentric visibility refers to the portion of a search space that is visible to a user at a particular time based on their location, direction and orientation. For example, in the TellMe application, a user’s visible query space in 2D acts as a secondary filter on data returned by the query processor by further restricting it to contain only objects that are in the user’s field-of-view.

Ego-visibility therefore excludes the portions of the dataset obscured by buildings and is primarily used to identify points-of-interest (PoI) other than buildings that are in a user’s FoV within a predefined distance (Figure 11). This query method could be used to identify objects that may be too small to point at precisely from a distance but are still in the user’s FoV nonetheless, like a monument or other tourism artefact. The algorithm...
to determine the query space in this instance builds on our previous work in Gardiner and Carswell (2003) where a 2D directional query processor was developed for an on-line virtual environment.

Although Nokia mapping is based on Navtec maps, for ease of implementation (i.e. avoiding Symbian C++ programming issues) and interaction we use Google Maps as the background map for the query results. The returned result is overlaid on the map in the form of building outlines or PoIs to offer the user an opportunity to adjust the pointing gestures and re-query if necessary. The mapping interface on the mobile device was shown in Figure 7, where the arrow indicates the position and direction a user is pointing and the colour represents the status of the calibration level of the sensor signals (i.e. green = good, red = poor). The following sections present the types of queries that can be performed by the TellMe application.

5.1 Point-to-Select Query

The Point-to-Select query function allows the user to point the device in a particular direction and using this direction vector, retrieve information about a particular object. All intersecting objects are presented initially to the user for refining the selection. When the user selects one from the list, an outline of it (outline coordinates taken from the Oracle database) gets displayed on the mobile device with the direction vector highlighted. An example of this is shown in Figure 10.

5.2 Field-of-View Query

The Field-of-View query simulates the user’s actual visual field in a particular direction and generates a 2D view shape as a query window in the spatial database. Only database objects that intersect the FoV shape get returned to the user. By default a 120° angle is...
chosen for the view field to mimic the natural view angle of human vision. The radius of
the query shape (visible distance) is dynamic and varies based on the tilting angle of the
device. For example, if the device is pointing vertically (i.e. to the sky) the radius is 0 m,
and if the device is pointing horizontally the radius is set to 200 m, with variable
distances in between. An example is shown in Figure 11.

5.3 Isovist Query

Recent work in this area by Simon and Fröhlich (2007b) describes a local visibility model
(Lvis) that uses the concept of billboards to determine what buildings are in the user’s
FoV in a 2.5D environment. To achieve this in our case, a different approach is taken
based on work carried out by Jiang and Liu (2009) into the concept of Isovists and
medial axes. Using Isovists, we attempt to automatically generate a panoramic (360°)
line-of-sight search space. This method has shown to be very effective as the fundamental
aim is firstly, to determine precisely the geographic area visible to the user in all directions
and secondly, to determine what objects inside this area are of interest to the user.
Considering the dense distribution of buildings in a city, by default we only consider
buildings within a 300 m radius in our calculations. Figure 12a shows the result of the
query displayed on the server, and Figure 12b shows the results overlaid on the map of
a mobile device indicating the user’s visibility convex polygon (Isovist). Figure 12c
illustrates how the Isovist can be used to identify all visible buildings or PoIs that interact
with a user’s complete visibility polygon and Figure 12d overlays them on the phone’s
map display.

5.4 Frustum Query

The Frustum View query simulates a user’s three dimensional field-of-view. It essen-
tially allows the user to utilize the mobile device like a flashlight beam scanning
the building to retrieve any information it illuminates. An example is shown in Figure 13.

In this case, a 3D Isovist is represented by a 3D object that represents a user’s actual LoS omitting the geometry of any other objects that interact with it. Building this type of 3D Isovist query requires a detailed 3D database of buildings and building “furniture” (e.g. windows, doors, etc.) at floor/room level in order to calculate the query frustum geometry accurately. With today’s usage of Lidar scans for 3D cityscape construction, the availability of such geometrically detailed datasets is becoming more and more common. Examples of possible query shapes used for the frustum are illustrated in Figure 14.

**Figure 12** (a) Isovist calculation as displayed on the server; (b) The shape of Isovist overlaid on the client-side map; (c) Further request to return all intersecting PolS to the user; and (d) A PoI query result using the Isovist shape as the query window

© 2010 Blackwell Publishing Ltd
*Transactions in GIS, 2010, 14(6)*
5.5 Ego-Dome

The Ego-Dome is essentially an extension of the 2D Isovist described previously except the visible open space around a user at a particular location is reproduced in 3D. The concept of querying all open space in this manner has been explored previously in military applications but usually in standalone desktop situations only. In the case of our TellMe application, an Ego-Dome is created using the current location of the user (Figure 15) and any objects that interact with the dome volume are detected and presented using visual, auditory, or tactile interfaces. This type of spatial interaction is very useful for a host of applications in terms of real-time feedback, allowing the user an ability to interact with and have knowledge about their surrounding environment in real-time.

6 Conclusions

The TellMe MSI application is designed to allow users to interact in a context sensitive way with environmental information using off-the-shelf mobile phone technology. Today’s mobile devices are beginning to offer integrated hardware such as digital compasses and tilt sensors combined with the now almost ubiquitous GPS sensor. Through these technologies, a wide spectrum of mobile spatial applications is emerging. Further development and testing of the DirectMe module and its synchronization with the LocateMe module is ongoing. Concurrently, spatial database development for refining/testing our directional query techniques in a comprehensive TellMe mobile application is taking place.

In relation to performing directional queries there are a number of fundamental issues, highlighted during our research, regarding the quality of the data produced from
the sensors. In Simon and Fröhlich (2008) the quality of the sensor data from a custom built sensor pack is analysed and compared to actual data showing good results for performing directional queries. With the quality of the sensor data being an important aspect of directional queries, in terms of the relevance of the results returned and the move towards making sensor interaction available to developers on mainstream mobile devices, we intend to carry out more sets of quality assurance testing on real-world datasets. We have already investigated the quality of selected external sensors (i.e. SHAKE and Nokia LD-4W GPS) and will compare these results with data quality returned from the integrated sensors in our off-the-shelf mobile phone (i.e. Nokia Navigator 6210).

Figure 14  Examples of possible 3D visibility frustum query shapes
Further testing will be carried out on a detailed NUI Maynooth 3D Campus model constructed from LiDAR data. In relation to 3D construction of the query shape, a new approach will be introduced that uses the FoV parameters from the built-in cell phone camera for the query frustum. This will combine reality with the query results in the same display. In addition, and in keeping with current trends of publishing mobile apps on the web, we intend to investigate making the TellMe technology available to a larger audience through various online app stores (e.g. iPhone App Store, Android Market, Nokia OVI Store).

Acknowledgements

The research presented in this article was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support.

References

Benedikt M L 1979 To take hold of space: Isovists and Isoivist fields. Environment and Planning B 6: 47–65

Figure 15  Ego-dome for determining the visible query space within a certain radius


Simon R and Fröhlich P 2007a The point to discover GeoWand. In Proceedings of the Ninth International Conference on Ubiquitous Computing (UbiComp 07), Innsbruck, Austria: 3–10


