spacebook-project.eu

D3.3.2 Final Populated City Model Component

Phil Bartie, William Mackaness,
Michael Minock, Johan Mollevik

Distribution: Public

SpaceBook

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

Deliverable 3.3.2
June 2013

COOPERATION

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

CogSys
Cognitive Systems

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	M27 = June 2013
Actual date of delivery	1 June 2013
Deliverable number	3.3.2
Deliverable title	D3.3.2: Final populated city model component
Type	Prototype
Status & version	Final 1.0
Number of pages	16 (excluding front matter)
Contributing WP	3
WP/Task responsible	UE / UMU
Other contributors	
Author(s)	Phil Bartie, William Mackaness, Michael Minock, Johan Mollevik,
EC Project Officer	Franco Mastroddi
Keywords	City model, GIS, PostgreSQL, PostGIS, OpenStreetMap, spatial data, networks, topology, route finding, wayfinding

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

Executive Summary

This report accompanies the city model prototype and extends the work developed in *D3.3.1 Initial City Model* and *D3.1.2 The SPACEBOOK City Model.*

The *Populated City Model* component is a storage facility for known physical features, and the uses of those features, within the city. This includes polygons, points, and polylines as well as attribute information. The City Model is stored in a PostgreSQL database using the PostGIS extension, and includes the capability to carry out wayfinding tasks on a network using the pgRouting extension. The network is suitable for pedestrian route finding, and includes steps and pathways not accessible to vehicles. Topographic information for terrain elevation is stored with each network node and segment midpoint so that additional hill details may be included in the wayfinding instructions.

The city model component accesses a number of feature type sources, such as *OpenStreetMap*, to provide information on how space in the city is used, for example if a building includes a restaurant, public telephone, or cash machine.

This report describes the data included in the model, the database schema, and examples of how it may be accessed using SQL.

1 Overview

The *city model component* in SpaceBook is the central repository for information about the city, and is accessed by the Interaction Manager (IM) component (Figure 1). It contains both spatial representations of features as well as attribute information, such as names and cuisine types. It also includes a topological network dataset for determining optimum routes between locations.

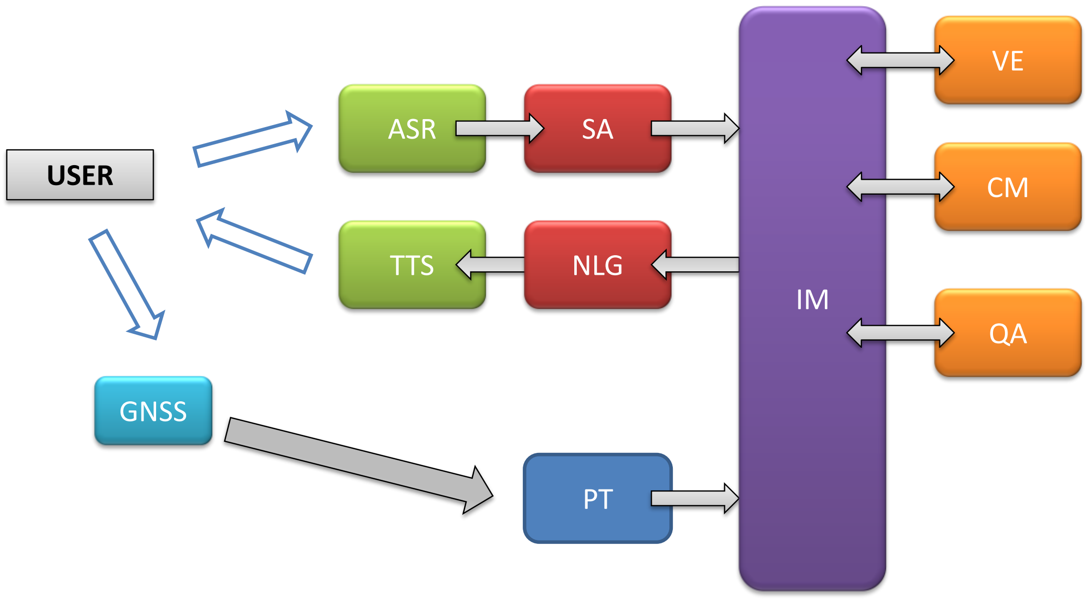


Figure 1: SpaceBook Components

ASR: automatic speech recognition SA: semantic analysis IM: interaction manager VE: visibility engine
CM: city model QA: question answering NLG: natural language generation TTS: text to speech
GNSS: global navigation satellite system PT: pedestrian tracker

1.1 Requirements

The city model is required to support a number of queries which determine answers to questions asked by the user including:

- Where is X?
- What is X?
- How do I get to X?

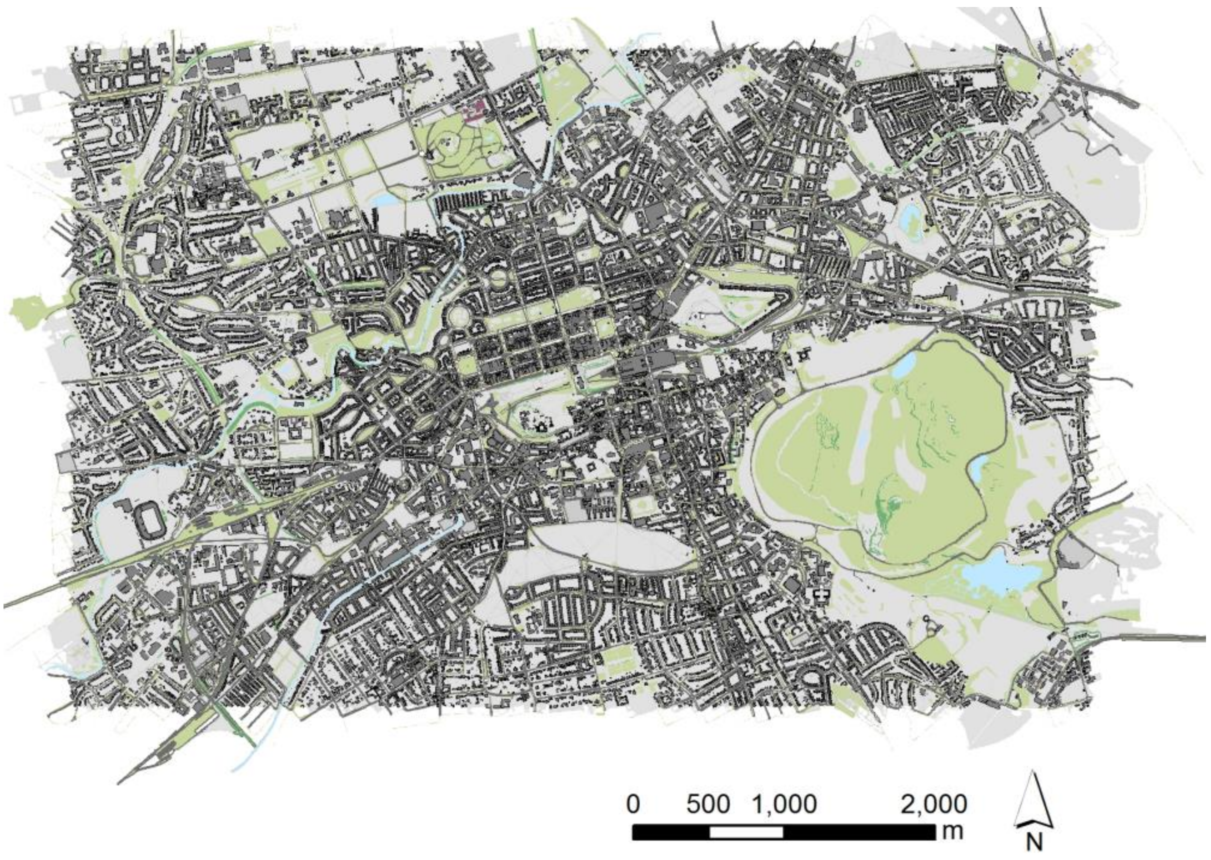
In addition to the user *pulling* information from the city model it is also used to *push* information considered relevant and of interest to the user. This requires querying of the city model at frequent intervals to determine what may be of interest to the user based on previous preference selections, user histories, and context (eg vista space, time of day, weather, current task).

To answer questions on ‘how do I get to?’ or ‘where is?’ the model calculates the most suitable pedestrian route using a topological network, and supplies details with reference to key navigational landmarks visible along the journey.

2 City Model Data Architecture

2.1 Edinburgh City Model Extent

The Edinburgh City Model is centred on Waverley Station and Princes Street and covers just over 6km by 4.5km as shown in Figure 2. It consists of 124429 polygons, 23335 points, and 10578 polylines.



(MasterMap data, Ordnance Survey © Crown copyright. All rights reserved OS)

Figure 2: Edinburgh City Model Extent

2.2 Data Quality and Data Integration

There are many sources of data of varying quality and completeness available which can be used to populate the city model. There is however a requirement to define a base layer which forms the foundation to which other datasets are attached. For reasons of positional accuracy, completeness, quality, and resolution the Ordnance Survey Master Map (OSMM) digital data series was selected for the Edinburgh pilot study area. In other regions where such sources are not available OpenStreetMap (OSM) data may be used. However the digitisation processes used in populating the building polygon layer in OSM varies greatly across cities, and most frequently does not consider radial distortion issues. As a result buildings are mapped to their roof line, rather than their foundation, and this introduces building location errors up to tens of metres from their actual location. The OSM data itself does not contain building height information and therefore is considered unsuitable for use with visibility analysis.

While the Ordnance Survey (OS) provide a number of road transport network layers, such as the Integrated Transport Network (ITN), they are less suited to the access pathways used by pedestrians than that available from OSM. Therefore OSM routes were used in forming the topological network which to support way-finding tasks.

The Point of Interest (POI) were sourced from OS PointX, OSM, and the Gazetteer for Scotland. Please note that due to data licencing requirements the OS (Master Map and PointX) and Gazetteer for Scotland data, and derived products, have had to be removed from the public populated city model prototype.

2.3 Coordinate Systems

The most popular *Global Navigation Satellite Systems* (GNSS) is the US based Global Positioning System (GPS) which uses the WGS84 geographic coordinate system (i.e. EPSG:4326). While WGS84 is a suitable general model for world based navigation there are a number of advantages in using a local projected coordinate system for the city model. These include the ease and speed of calculating distances and areas, and greater functionality available in PostGIS, the spatial database extension used to handle the city model data inside PostgreSQL.

For the Edinburgh pilot study region the British National Grid (OSGB36) is used (i.e. EPSG:27700). There are a number of transformations available between WGS84 and OSGB36, varying in complexity and spatial accuracy. The most accurate uses an NTv2 grid transformation (also known as OSTN02 in UK) as it introduces minimal positional errors, however for Edinburgh the difference in value between a 7 parameter transform and an NTv2 transform are less than 1 metre. NTv2 grids are not so commonly supported in applications, and if both the city model and user's GPS location are transformed using the same algorithm then any offsets will be aligned. For this reason a 7 parameter transformation will be used throughout the project for the Edinburgh test area.

The function to convert from WGS84 to OSGB36 is provided using the open source CS2CS tool as follows:

```
cs2cs +proj=latlong +ellps=WGS84 +towgs84=0,0,0 +nodefs +to +proj=tmerc
+lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy
+towgs84=446.448,-125.157,542.060,0.1502,0.2470,0.8421,-20.4894 +units=m
+no_defs
```

Test for a point in Edinburgh (Scotland): WGS84 location (-3.185, 55.95)
OS Online NTv2 Conversion: (326097.07,673630.932) [taken to be correct conversion]
ESRI (3 par option_1): (326097.86,673625.357)
ESRI (3 par option_5): (326095.042,673619.452)
ESRI(petroleum option): (326096.974,673630.405)
CS2CS (as above): (326096.98,673630.4)

Note the 5-10m errors introduced when using 3 parameter transformations.

A large number of datasets from a range of providers are fused together in the city model, in many cases relying on the spatial location to link data.

2.4 Sites (partonomic structure)

The database has been built around points, lines, and polygons. The polygons are treated as the smallest regional unit and the type is in the *isa* table (eg building, road region). However collections of polygons may be treated as sites, such as Edinburgh Castle which consists of 326 polygons (see Figure 3). The membership of these sites is handled in the city model using two tables which are the sitelink table, and sitelink_site table. A polygon may be a member of many sites (eg part of Edinburgh Castle, and also part of the Old Town). The hierarchy of memberships can be calculated by checking the site_area value, which is calculated from the combined area of polygon members.

```
Tables:      sitelink: id (integer)  site_id (bigint)
             sitelink_site: site_id (bigint) site_name (text)  site_type (text)
                            site_area (double precision)
```

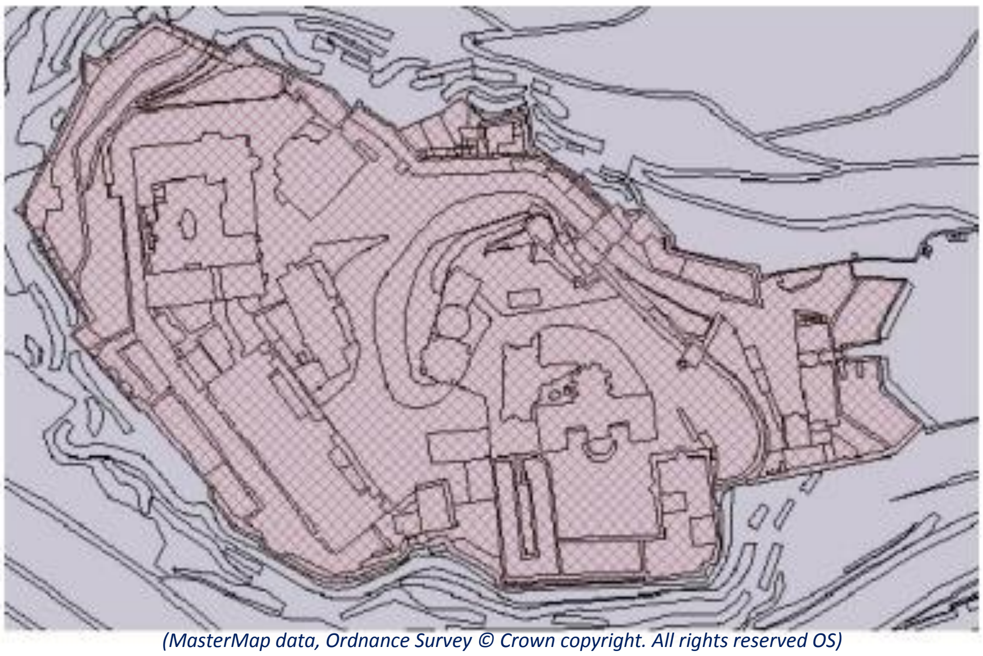

(MasterMap data, Ordnance Survey © Crown copyright. All rights reserved OS)

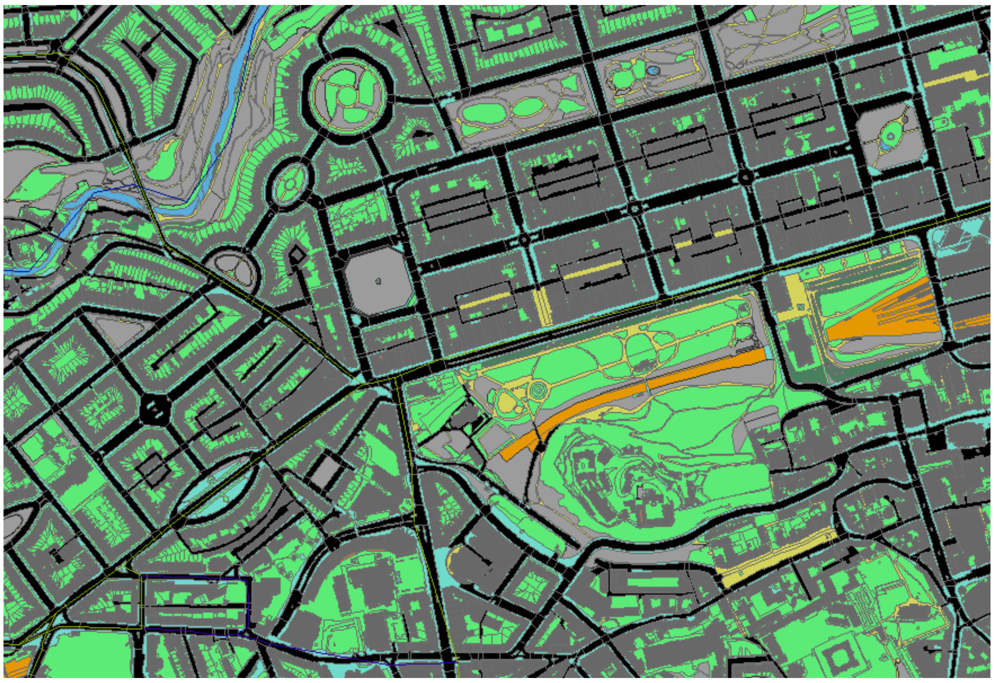
Figure 3: Edinburgh Castle as a Site

3 Data Sources

The following section outlines the datasets used in the Edinburgh City Model, some of which are used under licence and not publicly available.

3.1 Landcover including Building Outlines (polygons)

The OS MasterMap layer gives 100 percent coverage of the test region space, and defines the land surface type (natural land, building, river, etc). This layer is used to define building objects, which when linked to the DSM/DTM is used to calculate their visibility by the visibility engine component.

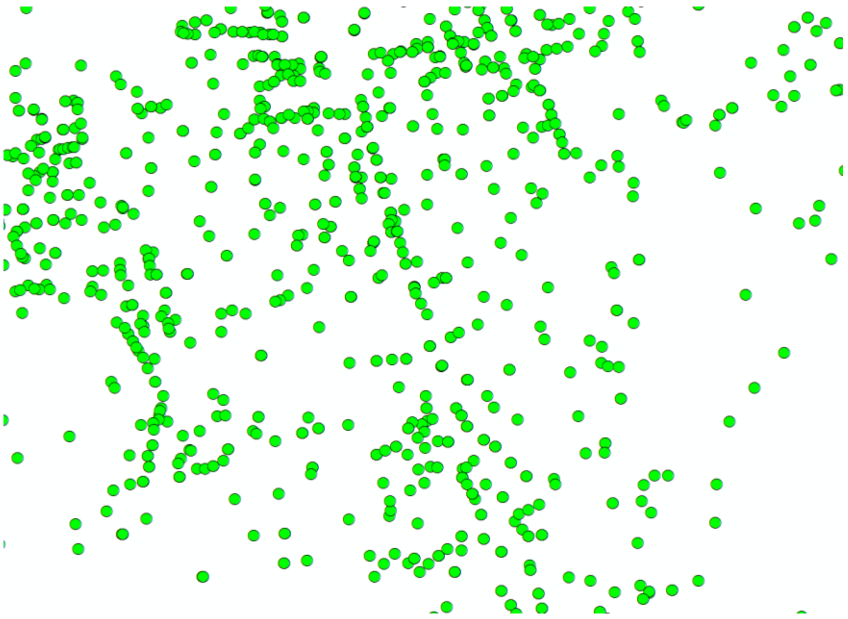


(MasterMap data, Ordnance Survey © Crown copyright. All rights reserved OS)

Figure 4: Landcover

3.2 Points of Interest (POI)

The POI layer adds attributes for building use (eg café, restaurant) and other facilities such as public telephones and ATMs.

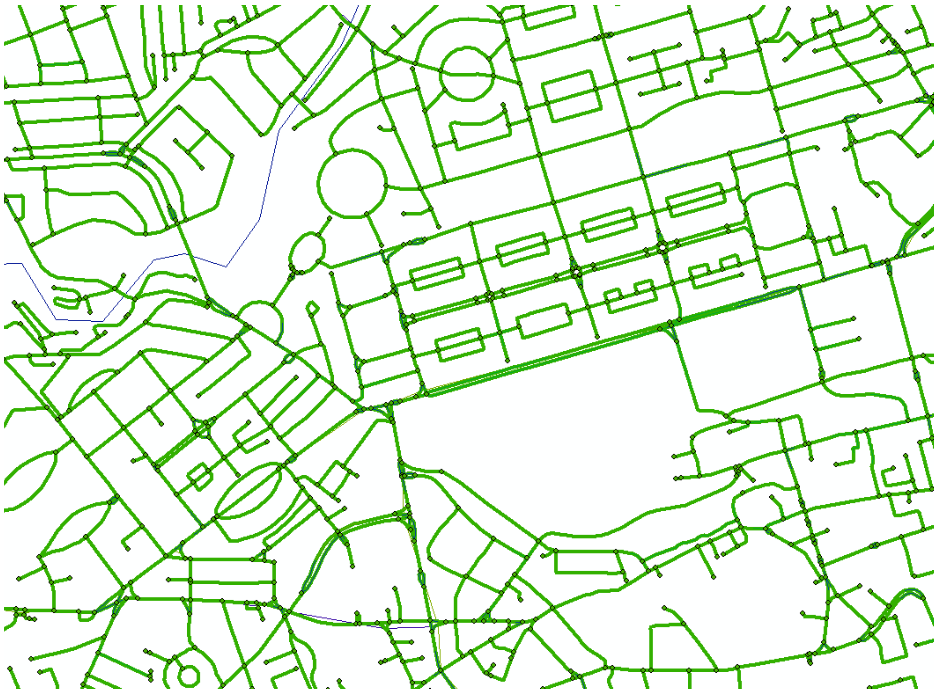


(PointX dataset - Ordnance Survey © Crown copyright. All rights reserved OS)

Figure 5: Points of Interest

3.3 Route Networks (topological model)

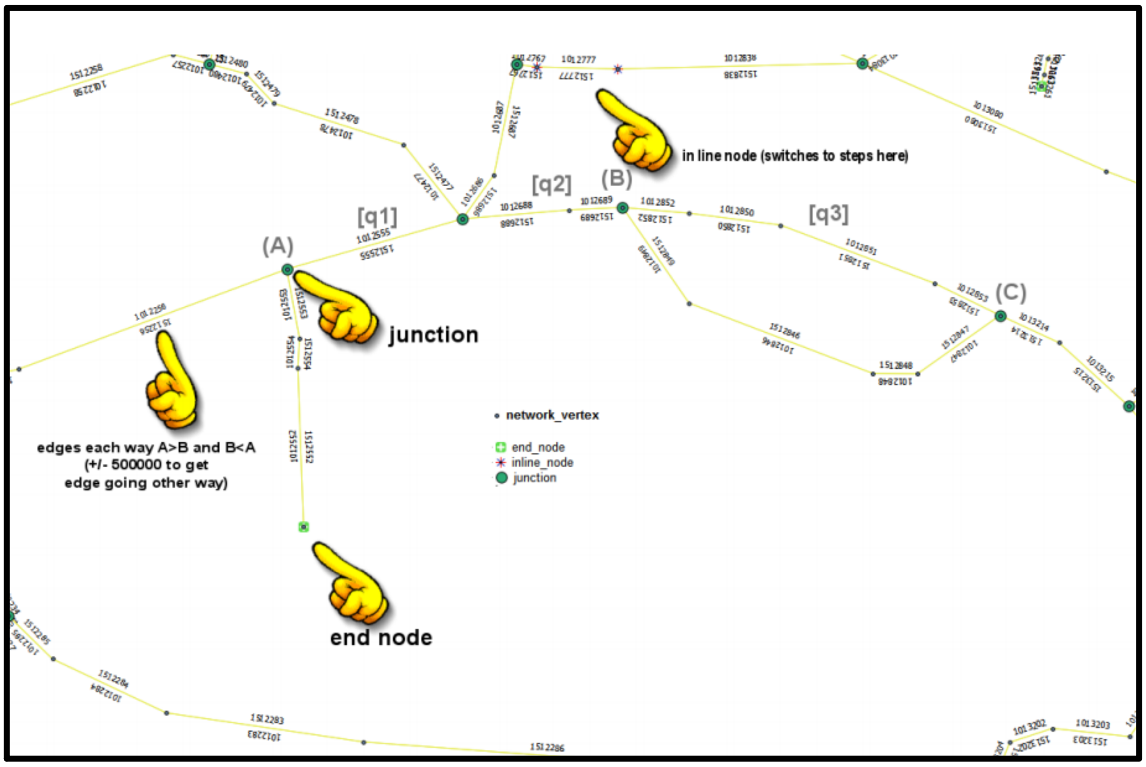
The road network is used to determine the optimum path between destinations, and is suitable for pedestrian wayfinding. This is sourced from OSM and includes main roads, tracks, paths, and steps.



(© OpenStreetMap contributors)

Figure 6: Road Network

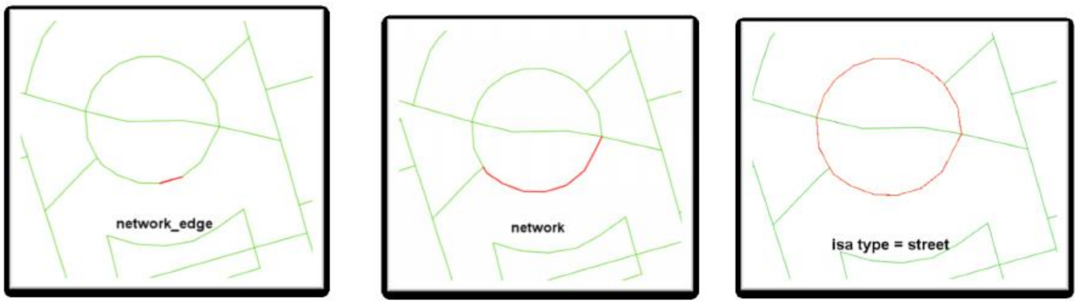
The network is stored in two complimentary ways. The *network* and *node* table can be used by pgRouting to calculate the shortest path. However the network data are also stored as separate parts as tables *network_edge, network_vertex, network_lieson.* A street is defined as a collection of network segments which connect to each other and have the same name, the geometry is stored in the *haspolyline* table, and attributes are held in the *isnamed* and *isa* tables. All geometries are defined using vertices, and a node is a special vertex found at decision points (junctions), ends of network (cul de sac), or where there is a change in network type (eg road, bridge, steps). An edge is a straight line between two vertices, which are joined together to make network table. Further definitions are given in Figure 7.


in line node (switches to steps here)
[q1]
[q2]
(B)
[q3]
(A)
(C)
junction
edges each way A>B and B<A (+/- 500000 to get edge going other way)
end node
network_vertex
end_node
inline_node
junction

(© OpenStreetMap contributors)

Definitions:

- network_edge => each unique number between vertices
- node => junction point (between many edges, or end of an edge, or inline where network edge type changes (eg street, steps))
- network => junction to junction road pathways (eg q1,q2,q3) – can be single edge or many edges depending on road pathway
- isa type = 'street' => collection of network pathways which connect and are of same name (eg Q may consist of q1, q2, q3)
- haspolyline => stores the geometry for the 'street' level of road network (with isnamed and isa tables)
- network_lieson => id of each network feature, and the street id that it occurs on (eg edges, nodes all occur on a street id)
- network_vertex => every point which defines a road network pathway exists as a uniquely identifiable vertex in this table, those which are nodes also appear in the node table and in the isa table under type 'junction'
- isA = 'street' => this id at street level will be found in the haspolyline, isnamed tables. Also in the network_lieson table which will be the link to network_edge, network, and the nodes.

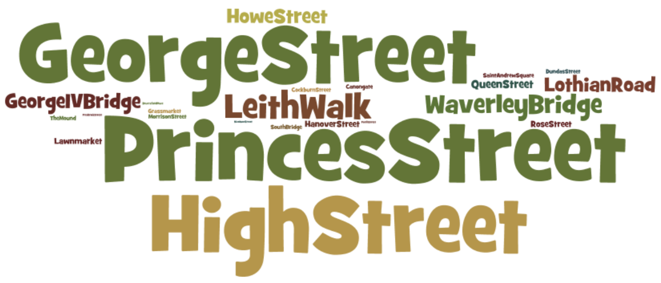


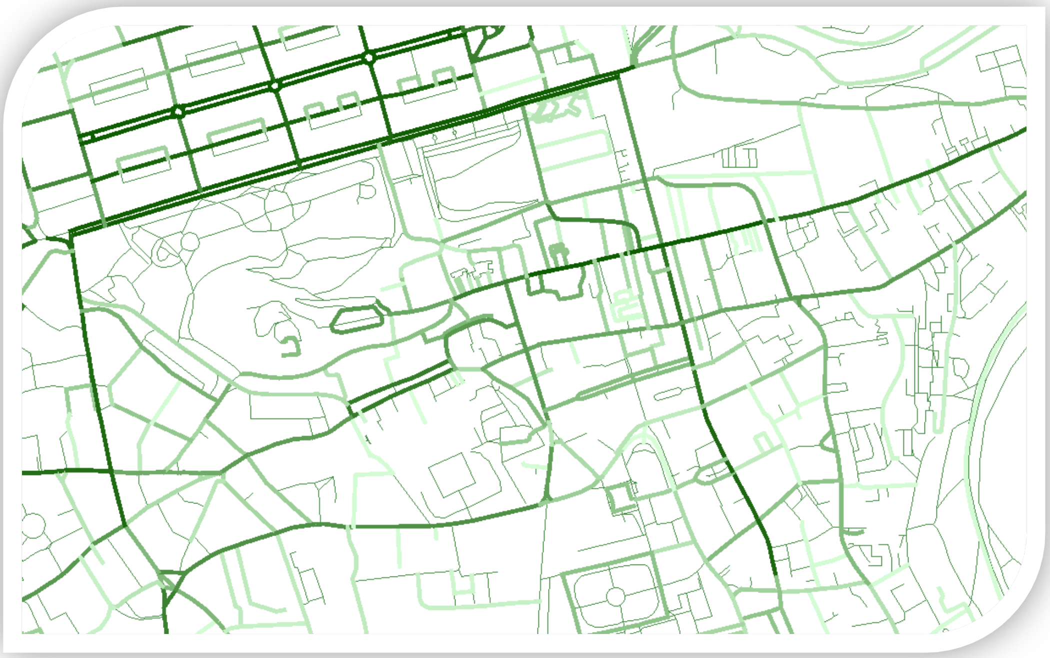
(© OpenStreetMap contributors)

Figure 7: Network Table Definitions

3.4 Well Known Streets

To be able to better model the user as they explore Edinburgh data were collected on how well each network segment is known. This included using FlickR photograph locations, and Foursquare checkins to determine which parts of the city and streets were the most frequently visited. These values were normalised as integer values from 1-1000 and saved in the *network_known* table. Higher values indicate the street is generally better known, and this can be used to customise route instructions, or to suggest places to visit when pushing information to the user.



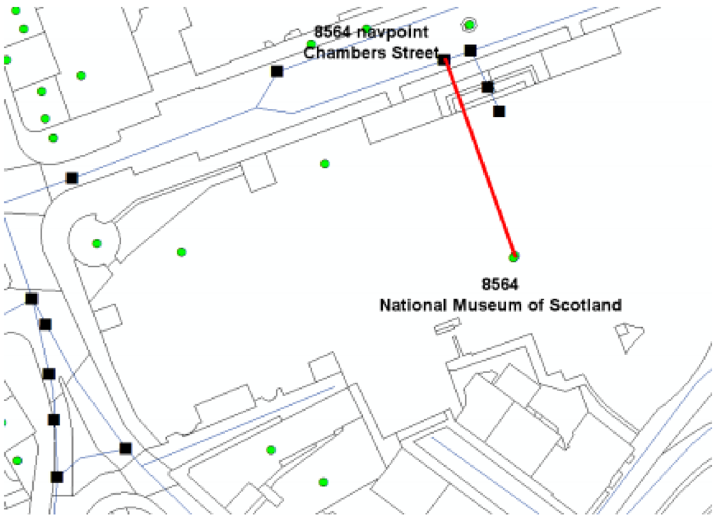


(© OpenStreetMap contributors)

Figure 8: Well known streets of Edinburgh (shaded darker if better known)

3.5 Navpoints

Navpoints were introduced as an alternative location to the hasPoint for OS PointX data that included a street address. Using this information it was possible to geocode the OS PointX data to the main address location and snap this to the OSM street network. This is useful during navigation tasks to ensure the user is directed to the main entrance rather than nearest street based on the location of the hasPoint feature which could be a rear or side street without building access. Figure 9 shows an example of the *navpoint* for the National Museum of Scotland, which is situated on Chamber Street and the corresponding *hasPoint* which is arbitrarily located within the building polygon, and actually closer to the rear street which has no public access to the building.



(Ordnance Survey © Crown copyright. All rights reserved OS)

Figure 9: Navpoint (black squares) **and corresponding hasPoint** (green circle) **for National Museum of Scotland**

4 Data Extraction, Transformation and Loading

The following steps were carried out to extract, transform and load various data into the city model component, which can be queried in real time from other SpaceBook components (see Figure 1).

1) Prepare GIS datasets as Shapefiles by using the field calculator to set IDs to increase from 1

2) Load the shapefile dataset into PostGIS using OGR2OGR (open source tool)

```
ogr2ogr -f "PostGreSQL" -a_srs "EPSG:27700" PG:"host=localhost
user=yourusername dbname=sbcitymodel password=yourpassword"
c:\dataset.shp
```

3) Use OGR2OGR to load OS Master Map polygon building layer
To load MasterMap into PostGIS requires the removal of any Int64 columns. Also when selecting polygons from OSMasterMap for a desired Area of Interest a clipping tool should not be used, as it results in duplicated TOIDs and a multipolygon data type. Instead a polygon selection based on spatial intersection is required, giving a ragged edge but ensuring a polygon data format with unique TOIDs.

```
ogr2ogr -f "PostGreSQL" -a_srs "EPSG:27700"
PG:"host=localhostuser=yourusername dbname=sbcitymodel edin
password=yourpassword" c:\dataset.shp
```

4) Topological Network
The previously developed OSM2SB tool (D3.1.2) allows for OpenStreetMap XML data to be uploaded to the central PostgreSQL database. One of the layers available from this is a route network which provides details of the pedestrian pathways, tracks, steps and roads.

The network table is used by the pgRouting database extension, which must be installed on the host database server. This is done by downloading the pgRouting extension and running `routing_core.sql`, `routing_core_wrappers.sql`, `routing_topology.sql` files from the City Model database.

5 Database Schema

5.1 Extensibility

The city model adopts a vertical partitioning approach to data storage, as outlined in public deliverable D3.1.2. This enables new data and knowledge about existing entities to be easily added to the *city model* without database redesign. The mostly static nature of the city model enabled wide variety of GIST-based indexes [1] without having to consider the costs of run-time insertion into the indexes. It is important to write spatial SQL queries that use these indexes during execution.

5.2 Schema Overview

An overview of the tables within the City Model is provided here, further details may be found in D3.1.2.

Table	Description
_featuretypes_level1 **_featuretypes_level2** **_featuretypes_level3** (catid, featuretype)	A hierarchical definition of feature types which are used in the *isa* table. These tables are for reference and are based on the Ordnance Survey Points of Interest user guide contents. e.g. (level 3) accommodation_eating_and_drinking > (level 2) eating_and_drinking > (level 1) cafes_snack_bars_and_tea_rooms
_saliency_brandname (name, score)	A table for defining the better known brand names in Edinburgh based on crowd sourced datasets
_source(id,source,confidence)	A variety of data sources are used in the city model, and this table is used to track their origin. A confidence value is assigned based on the source and its trustworthiness. This value may be used to determine the most best feature to use when two or more are returned (eg for points of interest within a building polygon)
entity(id)	An entity is an object or a region. The argument id comes from a common namespace of integer values.
isA(id,type)	Each entity has a set of associated types. A given entity may be a member of multiple predicates. For example a given object could be both a bar and a restaurant.
isNamed(id,name,namesearch)	This associates text with an entity. The namesearch attribute is a tokenization of the given name represented in a tsvector.

	The tokenization speeds up partial string matches.
hasProperty(id,property,value)	This may be used to store extra information relating to an entity. The property field defines what type of information is stored. For example it is used to store cuisine type for restaurants and cafes. It may also be used to store telephone numbers, web addresses, street addresses and so on.
hasVisualDescription(id,type, value)	The City Model is able to store visual description details for entities. These may be used when assisting the user to identify an object in a view, or to find the destination. This table consists of types: is, has, opposite, nextto. This enables more intuitive ways of describing features in a form that it meaningful at the level of the pedestrian
hasPoint(id,geom)	Stores points of information across the city (eg hotel)
hasPolyline(id,geom)	Stores streets, rivers, rail
hasPolygon(id,geom)	Stores the regions which define buildings, open spaces, rivers etc
hasPolygon_dtm (id,dtm)	The z value for the base of each polygon is stored so that relative heights of buildings may be used in the project. For example in describing a building as being further up a hill, or higher than the user. The z values were sourced from the value of the DTM.
sitelink(id,site_id)	The sitelink table enables the definition of sites which consist of sets of haspolygon features. Edinburgh Castle is an example of site which consists of over 300 polygons. This enables partonomic modelling within the City Model.
sitelink_site(site_id,site_name, site_type, site_area)	This stores the site information, name, type and the ares is used to determine the site hierarchy should a polygon be a member of many sites (eg Edinburgh Castle and Old Town).
sblink (id, within_id, score)	The sblink table stores the results from spatially joining the haspoint and haspolygon table based on point in polygon relationship.
node(id, geom, nodetype, z,connections)	Corresponds to a branching point [2]. The z value comes from the DTM.
network (id, geom, pathtype, length2d, length3d, rdlength, startpoint,endpoint, start_id, end_id, sview, midptz, sinuosity, bendpt, bends, bendang)	Corresponds to a path segment [2]. The value for length3D is calculated across the terrain surface so includes real walked distance across hills, rather than straight line, point to point, distances. The attribute sview is a boolean for if the road is available in StreetView for web based trials, used to change behavior of routing engine (eg not tracks, steps, Princes street and other non-accessible streets to private vehicles). The attribute midpt is the DTM elevation at the midpoint along that edge used to calc if youre going up or downhill from a node (which have their own z elevation values) The attribute sinuosity is the calculated ratio of edge length against Euclidean distance from start to end node to give a value for how straight or bendy that edge is. The attribute bendpt is the greatest bend in the path segment and the bendang is the angle of such a bend.
Network_Vertex(id,geom)	Defines the network path shape
Network_Edge(id,fromId,toId,geom,type,networkid, networksequence)	Corresponds to an elementary segment [2] between 2 vertices. The network sequence value indicates the order an edge occurs along a network segment.
navPoint(geom, id, nearest_node, on_roadid)	where id relates to the hasPoint id nearest_node is the closest node id on_roadid is the network id that the navpoint is on these are stored in this table for speed purposes so we don't need to repeatedly carry out spatial joins.
vernnames (name,namebreak)	The vernacular names in the isnamed table which relate to the haspolygon and isa entires were sourced from processing FlickR data. The delivery of tags from FlickR can have spaces removed, and this table stores the results of breaking those names into their most likely original form.

6 Example Queries

6.1 Shortest path

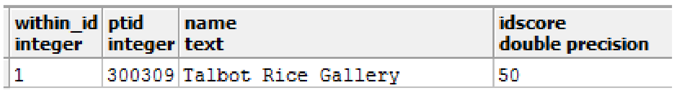
Calculate the shortest path between two nodes on the network using pgRouting.

```
SELECT vertex_id,edge_id,cost FROM shortest_path('SELECT id AS id,start_id::int4 AS
source,end_id::int4 AS target,length3d::float8 AS cost FROM network', 2007554, 2006726,
false,false;
```

6.2 Main Polygon Name

As each polygon may have many points of interest within it, there was need to develop a method to rank the output and offer the most suitable name for that polygon. This is considered as the mainname, although the polygon may be known by any of occupant names. For example a building may be occupied by Blackwells bookshop, Café Nero coffee shop, and an accountant office space above. When pushing information to the user it is necessary to determine the most suitable name to use for that building. This is performed using the sb_polygonname(entity id) function.

```
select * from sb_polygonname(1);
```

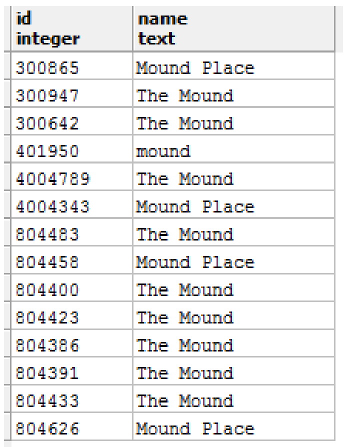


within_id integer	ptid integer	name text	idscore double precision
1	300309	Talbot Rice Gallery	50

6.3 Fuzzy Name Searching

To improve name based searching a fulltext search function is used. This will find results with words in different orders and ignores stop words. For example searching for ‘Edinburgh University’ would also find ‘The University of Edinburgh’. The method supports & (and), | (or), and ! (not) parameters.

```
select * from sb_name_fulltext('the & mound');
```



id integer	name text
300865	Mound Place
300947	The Mound
300642	The Mound
401950	mound
4004789	The Mound
4004343	Mound Place
804483	The Mound
804458	Mound Place
804400	The Mound
804423	The Mound
804386	The Mound
804391	The Mound
804433	The Mound
804626	Mound Place

```
select * from sb_name_fulltext('the & mound & !place');
```

id integer	name text
300947	The Mound
300642	The Mound
401950	mound
400478	The Mound
804433	The Mound
804483	The Mound
804400	The Mound
804423	The Mound
804386	The Mound
804391	The Mound

6.4 Joining tables

The following example shows how to use the entity id in a join between the *isNamed* table and the *hasProperty* table to find places that sell pizza.

```
select a.id,a.name,b.value from isnamed a
join hasproperty b on a.id=b.id
where property = 'cuisine_type' and value='pizza';
```

id integer	name text	value text
300415	Marchmont Take Away	pizza
301195	Pizza Express	pizza
301365	Pizza Paradise	pizza
301445	L'Aquila Bianca	pizza
301446	Franco's	pizza
301490	Pizza Hut	pizza
301737	Pizza Express	pizza

6.5 Spatial Example

This example demonstrates how to supply a coordinate and return all entities within 80 metres. The st_dwithin(geom,geom,double) automatically makes use of any existing spatial indexes.

```
select a.id,b.name, st_distance(a.geom,st_geometryfromtext('Point (325762 673323)', 27700)) as
distance from haspoint a join isnamed b on a.id=b.id where
st_dwithin(a.geom,st_geometryfromtext('Point (325762 673323)', 27700),80);
```

id integer	name text	distance double precision
301000	Museum Of Scotland	11.2681504791456
300648	George IV Bridge	51.1608281146996
301365	Pizza Paradise	74.7086151805446
300647	Chambers Street	69.8018713899309

6.6 Nearest Road

A frequent requirement is to determine the closest node or network segment from a supplied user position coordinate. To make this process simpler a function was added which returns the closest road, from which the corresponding nodes can be retrieved. For example in Figure 10 the user located at A would be considered to be on the Princes Street network segment 803865. The straight line distance from the user to the network road centreline is also returned.

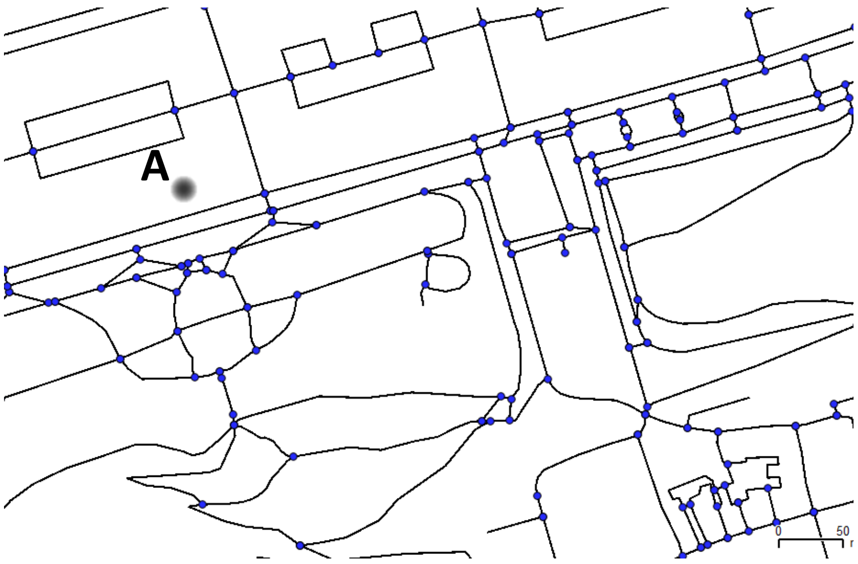
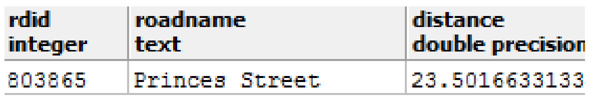


Figure 10: Nearest Road for a Supplied Coordinate

select * from sb_nearest_road(325135.1,673831.9);



rdid integer	roadname text	distance double precision
803865	Princes Street	23.5016633133

7 Conclusion

The city model plays a critical role in the SPACEBOOK project. It is the spatial repository which stores the geographic and related attribute information for features in the city upon which the Interaction Manager and other modules rely. The spatial database used was PostgreSQL [3], with PostGIS and pgRouting extensions, and data is retrieved using spatial SQL [4,5].

Modelling the pedestrian for navigation is more difficult than vehicle navigation, as pedestrians are able to explore more freely and are not strictly tied to road networks but can also use open spaces, footpaths, and steps. In examining the ways that pedestrians navigate that space, it was clear that rich descriptions of space were required in order to unambiguously instruct the pedestrian and to refer to different objects relative to other features. This required us to model the city at different partonomic scales, to include the facility to store visual description of buildings, and to consider the familiar and the vernacular information gathered from social media sites such as Foursquare and Fickr.

In the past 12 months, the Edinburgh City Model has handled 7.8 million queries with an average processing time of 4ms and PostgreSQL and PostGIS have proven to be a stable platform with very good performance.

8 References

[1] J. Hellerstein, J. Naughton, and A. Pfeffer. Generalized search trees for database systems. In VLDB, pages 562–573, 1995.

[2] Kai-Florian Richter and Alexander Klippel. A model for context-specific route directions. In Spatial Cognition, pages 58–78, 2004

[3] B. Momjian. PostgreSQL: Introduction and Concepts. Addison Wesley, 2001.

[4] H. Samet. Applications of Spatial Data Structures. Addison-Wesley, Reading, Massachusetts, 1990.

[5] S. Shekhar and S. Chawla. Spatial Databases: A Tour. Prentice Hall, 2003.