Performance evaluation of storing and querying spatial data on mobile devices for offline location based services

Ricky Jacob[*], Sean Smithers[**], Adam C. Winstanley[*]

**Geotechnologies Research Group*
Department of Computer Science
National University of Ireland, Maynooth
email: rjacob@cs.nuim.ie

*** Department of Computer Science*
National University of Ireland, Maynooth
email: sean.smithers@nuim.ie

***Abstract*— Use of location based services has been gaining popularity over the years especially with the increase in smartphone sales. The benefits include helping you find nearby places of interest like a restaurant, café, atm among others and additional information like location and distance and providing navigation assistance to these places. In most cases such services are dependent on the availability of an internet connection on the phone. Thus the use of such services is lost in scenarios where an internet connection is not available or not used (due to expensive internet plans) like when visiting another country. We evaluate the performance of querying spatial data when stored locally on the mobile device. The circular query and the pointing gesture based geowand query are performed on a copy of the OpenStreetMap data stored in SQLite on the mobile device. The SpatiaLite extension provides the vector geodatabase functionalities to query such databases. In this paper we report on the performance of such queries on mobile devices. We also compare some features of this offline prototype with alternative proprietary solutions like Google Places and highlight the importance of Open data and crowdsourcing.**

***Keywords* – location based services, geowand, spatialite, mobile interaction**.

I INTRODUCTION

In the past couple of years mobile phones have developed into devices not only capable of making and receiving phone calls but can now access the internet via Wi-Fi or mobile networks. The majority of devices known as 'smart' phone also come embedded with a GPS receiver [1] as well as other in-built sensors like gyroscope, proximity sensor and accelerometer.

With the availability of such in-built sensors in handheld devices like a 'smart' phone, Location Based Services (LBS) have become more and more popular [2][3][4]. Any application that makes use of the user's current geographic location of the mobile device (hence the user) can be classified as an LBS application. With the help of such applications, users are able to find Points of Interest (POI) like café or restaurants in their local area through recommendations or queries.

In the UK alone, 84% of smartphone owners use their device to search for local information with 78% of these users taking action afterwards [5]. This shows the demand for these types of applications and their benefit to local businesses and communities.

There is a limitation associated with the majority of these applications though; they all require a network connection to function fully [1]. This means that for users without an available network connection (Wi-Fi or mobile network) or without the funds to pay for expensive mobile network data plans while visiting another country would not be able to use such applications to good effect [1].

In this paper we test and report upon performance of an offline spatial database stored in SQLite (with SpatiaLite extension) on a smartphone. We used the geographic data obtained from OpenStreetMap which is Open data produced by crowd sourcing [6]. We performed two types of queries – the *circular* query and the *Geowand* query. To check how size and performance are related, we tested the queries with three datasets – Maynooth, Ireland and UK.

In the following section 2, we review literature related to offline location based services and usefulness of crowd sourced open data.

II LITERATURE REVIEW

In Coelho et al. [1] a system for retrieving and accessing location packages is proposed and a prototype system developed to show how this works. In their paper they propose that user's access location based content by scanning a 2D barcode or RFID tag. This location trigger then accesses the location content from a local database stored on the device.

The model proposed by Coelho et al. fails if a user wants to query what is around them instead of just accessing information for a single location. They don't consider the case where an application queries spatial data using geometric functions based on the user's current position and what is around them.

During the last few years Volunteered Geographic Information (VGI) such as that provided by OpenStreetMap has become more and more popular [8]. The quality and quantity of this data continues to improve and can provide an alternative data source to those provided by closed data sets such as Google Maps or Bing Maps, although this can depend greatly on the area in question [9].

There are currently LBS applications available for Android that function offline such as the offline readability of Google Maps [2] or OSMAnd [8] and MapDroyd [9] which both provide offline OSM maps for Android users. These apps for android are just simple map viewers, some of which only include basic search options. Google applications such as Google Places allow users to search for nearby POIs by category. This however can only be used with data provided by Google and requires a network connection to function. Here the ability to use custom data would be of great advantage to users. Events such as conferences could provide custom POI and map data to attendees before an event takes place [10]. Colleges and universities could also provide detailed information, such as lecture theatre locations, to students on the campus.

In the following section we discuss the system components. This is followed by the experiments carried out to test system performance while performing spatial queries.

III SYSTEM DESIGN

The development of an offline location based service requires us to consider few key elements which constitute the system. We try to understand them in detail.

a) Data storage

One of the most important challenges in any offline system is the storage of data on mobile devices for querying and manipulation. SpatiaLite is an extension for SQLite that adds spatial functionality and is Open Geospatial Consortium (OGC) compliant. The SpatiaLite provides the vector geodatabase functionalities to query and manipulate such SQLite databases. Thus the entire database obtained for the selected area is stored locally on the phone.

The data to be stored by the application can be broken into two areas:

1. Spatial data
2. Map data

The spatial data is the exported OSM data and, combined with some meta-data, provides the data for all of the applications functionality. This data is stored in a SQLite file (.sqlite extension) on the devices external storage. The OSM data used is the *planet_osm_point* table (see Figure 1) taken from a PostGIS database. This table structure is the one used by OpenStreetBrowser and is created when OSM data is imported into a PostGIS database using *osm2pgsql* [16].

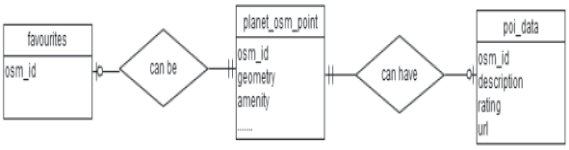

Figure 1: Database entity relationship diagram

This data is exported from a PostGIS database to a SpatiaLite file using the *ogr2ogr* [17] tool. This process creates several other tables, such as *spatial_ref_sys*, used to store spatial meta-data.

After exporting, other tables are added to the file through the use of a simple SQL script. This script sets the database up to be used on an Android device, creates a table to store user favourites (favourites) and also one to store extra information on POIs (*poi_data*).

The map data is a single file (*.map* extension) that is used by the application to render map tiles. The map file is generated using the *Mapsforge* plugin for Osmosis by providing an OSM XML file or *shapefile*. The structure of this file is defined by the Mapsforge project.

b) Map

Representation of geographic data on a map to help users navigate is an important challenge. Most of commonly used techniques involve storing map tiles as PNG files in an organised directory structure. These tiles are then stitched together by the map view before being displayed. MapsForge is an open source project that allows you to generate a single map file from OSM data [13]. This map file can be stored locally on the mobile device and is used by the map view provided for Android to render the map at runtime. This removes the need to store individual PNG tiles and is very simple to implement. Thus Mapforge was used to prepare maps for the selected databases.

c) POI Queries

Querying for POI information is one of the most common services provided by such systems. There are various query techniques in mobile spatial interaction like *touch2query* [14], *circular query* (as shown in figure 2) and *geowand query* [15] as shown in figure 3.

The *circular query* takes the current location *A* and uses a radius, *r* to select the query region for querying the database. Users can search for POIs by specifying what categories they are looking for and the max distance they are willing to travel. All relevant POIs within this radius around the user's current position are returned as shown in figure 3.

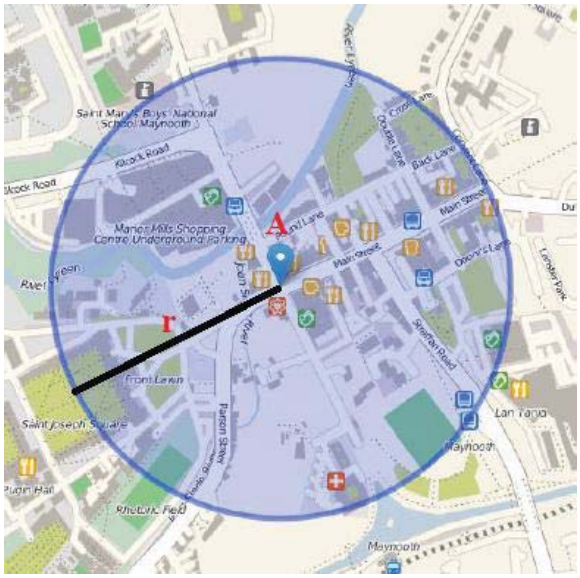

Figure 2: The circular query for POI search from a point *A* with radius *r*.

The *geowand query* takes the current location *A* and uses a radius, *r* to select the query region for querying the database based on phone pointing gestures. POI search can be narrowed by using the 'buffered bounding box' based search method. This method requires the user's current position, direction, *d* in which the device is pointed, the desired POI categories and max distance (*r* in this case) the user wishes to travel. Based on these three inputs the destination point is calculated which is *r* metres away from point *A* at an angle of *d* degree.

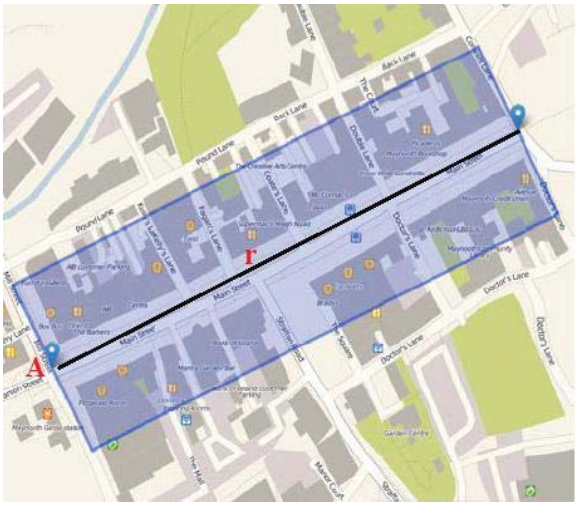

Figure 3: The geowand query for POI search from a point *A* with radius *r* along direction *d*.

From this information, a box is constructed that extends out on either side of the user and stretches out as far as the user has specified. Any relevant POI contained within this box is returned (see Figure 3). This is achieved using the ST_Within function as defined by the Open Geospatial Consortium (OGC).

d) User interface design

The interface was defined to let the user select the POIs that are of interest to the user. After selecting the query type and distance information about the POI is presented to the user with distance information and link to view it on a map. The user can save this location information for further use by marking it as '*Favourite*'.

The first tab the user sees is the search tab (see Figure 4a). A list of all POI categories is displayed here in alphabetical order. To search for POIs the user can check boxes beside each category they would like to include. A search button, at the bottom of this screen, will present different search options to the user via a dialog box (see Figure 4b) when pressed. This box asks the user to input the maximum distance they would like to travel, which can be changed using the onscreen slider. The current value of the slider is displayed to help the user select their distance. Below this, the user can chose between the different search methods by selecting one of the radio buttons shown. From here, the user can chose to continue or to cancel. If the user decides to continue the screen will update with a list of results based on the user's search criteria.

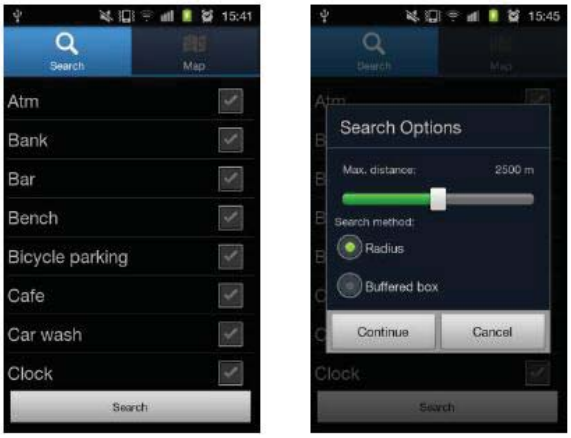

Figure 4: a) The *Search* tab with POI category list b) Advance features like distance and query type (*circular* and *geowand*)

Results are displayed as POI name along with the POI category displayed below it. The distance from the user's location to that POI is also shown. If a user clicks on POI from the results list they will be shown a screen containing more detailed information on that particular POI (see Figure 5a). They will also have the ability to add this POI to their favourites or view the POI on the map from here. To do this the user just has to click the relevant button on screen.

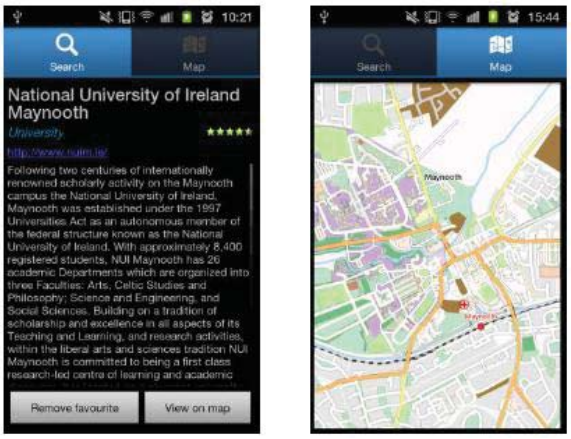

Figure 5: a) Additional POI information and marked as *favourite*. b) Map interface to view location information (*Mapforge*)

Various options are available in this tab by pressing the menu button on the device. From here the user will have the option to view their saved favourites, to select a database file to load or to view information on the application itself.

The second tab contains the map interface (see Figure 5b). The map view is a familiar and intuitive interface used in popular map applications such as Google Maps [2]. The user has the ability to scroll around the map using swiping gestures and also zoom-in and out using pinch gestures. There is also on-screen buttons for zooming.

Again, further options are available to the user by pressing the menu button on the device whilst viewing this tab. In this case, the user will have the option to view their current location (places a marker on the map at users location), the option to select a map file to load or to view additional information.

The class diagram of the entire system is shown below in figure 6.

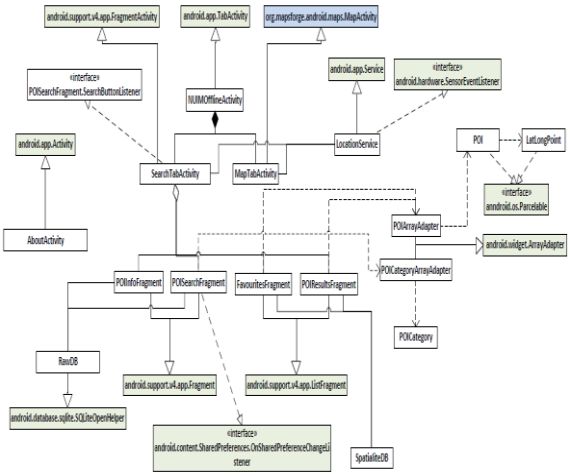

Figure 6: Class diagram of the system

IV EXPERIMENTS

Inputs for this experiment were the database and map files produced for three datasets –

1) Maynooth,
2) Ireland & Northern Ireland and
3) UK.

This let us check for system performance with various size of dataset.

a) Performance

To test performance a location, max distance, POI categories and orientation (see Table 1) were chosen and searches were performed using these with both the radius and buffered box methods on the 3 databases generated. For convenience the same parameters were used for each database. As the time for these queries varies, each one was performed 3 times and the average time taken as the result.

Location:	53.38301305, -6.60289324 (~ 4th year lab)
POI Category:	Cafe
Distance:	2.5km
Orientation:	~90°

Table 1: Performance testing input

b) Database result evaluation

To compare results produced by this application with those of Google Places an arbitrary location, POI categories and maximum distance was chosen and the same search performed in each application. Tests were performed using two different categories as seen in Table 2.

	Test 1	Test 2
Location:	53.38301305, -6.60289324 (~ 4th year lab)	53.38301305, -6.60289324 (~ 4th year lab)
POI Category:	Cafe	Pubs / Bars
Distance:	2miles / 3.2km	2miles / 3.2km

Table 2: Inputs for comparison of different applications

IV RESULTS

a) *Database size*

From Table 3 we see that the size of the files generated for the different areas are reasonable and could certainly be stored locally on a mobile device. Most current mobile devices come with multiple Gigabytes of storage built-in to the device or with the option to expand the storage space through the use of an SD card.

Area	Number of POIs	Database File Size (Bytes)	Map File Size (Bytes)
Maynooth	1,144	844,800	213,540
Ireland & NI	74,978	17,606,656	59,542,566
UK	1,010,551	231,768,064	432,612,951

Table 3: File sizes for map and POI data

b) *Query performance*

From Table 4 it can be seen that the number of POIs in the database does have a direct effect on the performance of the queries being ran, especially for the buffered box method.

Area	Number of POIs	Query Type	Time Taken (ms)			Average Time
			Test 1	Test 2	Test 3	
Maynooth	1,144	Radius	107	66	80	84.33
		Buffered box	2,831	2,945	2,937	2,904.33
Ireland & NI	74,978	Radius	1,479	1,444	1,617	1,513.33
		Buffered box	177,514	177,985	177,122	177,540.33
UK	1,010,551	Radius	14,801	15,190	14,806	14,932.33
		Buffered box	2,442,799	N/A	N/A	2,442,799

Table 4: Search query performance

The buffered box helps to reduce the query region to give information based on the road the user is willing to take. But depending on the size of the dataset, the buffered box search method can take too long to compute making this method unusable under certain circumstances.

As an indication for how this performance might map to other city level datasets, based on the number of POIs for some popular world cities was also recorded (as in Table 5).

City	Number of POIs
Berlin	11,778
London	93,139
New York	11,523

Table 5: Search query performance

c) *Comparison with proprietary systems*

The results returned by this project exceeded those returned by the Google Places application for both Test 1 and Test 2 (see Table 6). The results returned by the Google Places application were also irrelevant in both cases as an engineering company was also returned as a result in both tests.

	Number of POIs returned	
	Test 1	Test 2
Google Places	1	4
NUIM Offline	11	5

Table 6: Input for different application results comparison

V CONCLUSION

It has been shown that there are ways to store and query spatial data on a mobile device. The solution implemented has allowed two different search methods (*circular* search and *geowand* serach) but could certainly be extended to include others as well. The use of *SpatiaLite* meant that SQL written to run in PostGIS on a server can easily be ported to run on a mobile device as SpatiaLite is OGC compliant.

Maps can also easily be provided offline for users using open source services like *Mapsforge* and their implementation of *MapView* for Android.

In experimenting with the two search methods implemented, it was shown that, while these methods are feasible, the size of the dataset has a huge influence on the performance. As seen, it is possible to store the data required on a mobile device but it may not be possible to perform certain queries on large datasets. This suggests that this implementation is more suited to querying smaller datasets for cities and towns than larger, national, datasets.

The tools and data used by this project have also shown that there are open source components available to build this type of application using open data like OpenStreetMap. The results can even out-perform proprietary alternatives (such as Google Places) depending on the area and the level of geographic data coverage.

ACKNOWLEDGEMENT

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan.

REFERENCES

[1] P. Coelho, A. Aguiar and J. Lopes, "OLBS: Offline Location Based Services," in *Next Generation Mobile Applications, Services and Technologies (NGMAST)*, Cardiff, 2011.

[2] Google, “Google Maps for Mobile,” *http://www.google.ie/mobile/maps* [Accessed April 2012].

[3] Yelp! Inc., “Yelp for Mobile”, *http://www.yelp.com/yelpmobile*. [Accessed April 2012]

[4] Foursquare Labs Inc., “Foursquare”, *https://foursquare.com/download/*. [Accessed April 2012]

[5] J. Spero, “Google Inc.”, *http://googlemobileads.blogspot.com/2012/02/consumers-love-their-smartphones-now.html*.[Accessed April 2012]

[6] OpenStreetMap, “OpenStreetMap”, http://www.openstreetmap.org/. [Accessed April 2012]

[7] Michael Kenteris, Damianos Gavalas, and Daphne Economou Mytilene, “E-guide: a multiplatform mobile application tourist guide exemplar”, *Multimedia Tools Appl.* 54, 2 (August 2011), 241-262.

[8] P. Mooney, P. Corcoran and A. Winstanley, “Towards quality metrics for OpenStreetMap,” *in SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Jose, 2010.

[9] B. Ciepłuch, R. Jacob, P. Mooney and A. Winstanley, “Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps,” *in 9th International Symposium on Spatial Accuracy Assessment in Natural Resuorces and Enviromental Sciences*, 2010.

[10] R. Jacob, J. Zheng, A. Winstanley, B. Ciepluch and P. Mooney, “Campus Guidance System for International Conferences Based on OpenStreetMap,” in *Proceedings of 9th International Symposium, W2GIS*, Maynooth, 2009.

[11] OSMAnd., “OSM Android Offline”, *http://osmand.net/*. [Accessed April 2012]

[12] MapDroyd, “MapDroyd App”, *https://mapdroyd.com/*. [Accessed April 2012]

[13] Mapforge, “Mapforge – free mapping and navigation tools”, *http://code.google.com/p/mapsforge/*. [Accessed April 2012]

[14] J. Yin, and J. Carswell, “Touch2Query Enabled Mobile Devices: a Case Study using OpenStreetMap and iPhone”. *10th International Symposium on Web & Wireless GIS* (W2GIS2011), Springer LNCS; Kyoto, Japan, March, 2011.

[15] R. Jacob, and P. Mooney, and AC. Winstanley, “Whats up that street? Exploring streets using a Haptic GeoWand", *ADVANCES IN LOCATION-BASED SERVICES Lecture Notes in Geoinformation and Cartography, 2012*, Springer-Verlag Berlin Heidelberg pages 91 - 103. Eds Georg Gartner and Felix Ortag. November 2012

[16] OSM2pgsql, “OSM to PgSQL”, *http://wiki.openstreetmap.org/wiki/Osm2pgsql.* [Accessed April 2012]

[17] ogr2org, “Convert to Spatialite”, *http://www.gdal.org/ogr2ogr.html.*[Accessed April 2012]