2011-3

Touch2Query Enabled Mobile Devices: a Case Study using OpenStreetMap and iPhone

Junjun Yin
Technological University Dublin

James Carswell
Technological University Dublin, jcarswell@tudublin.ie

Recommended Citation

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

2011-03-01

Touch2Query Enabled Mobile Devices: A Case Study Using OpenStreetMap and iPhone

Junjun Yin
Dublin Institute of Technology

James D. Carswell
Dublin Institute of Technology, jcarswell@dit.ie

Touch2Query Enabled Mobile Devices: A Case Study Using OpenStreetMap and iPhone

Junjun Yin and James D. Carswell

Digital Media Centre, Dublin Institute of Technology, Ireland
{yinjunjun@gmail.com, jcarswell@dit.ie}

Abstract. This paper describes our mobile spatial interaction (MSI) prototype *Touch2Query* which presents the idea of using the touch screen on mobile devices to assist in performing ad-hoc spatial queries. This approach differs from conventional mobile LBS applications where the query shape (search space) is limited to either a bounding box or radius. Instead, we provide functionality that allows users to interactively draw any desired query shape overlaid on an area of interest directly on a mobile device with their finger by combining vector primitives such as circles, polygons, polylines, and points. With the help of location and orientation aware mobile devices, mobile maps, and real-time distance and area measurements, *Touch2Query* gives the users freedom to perform customised spatial queries on objects/areas of interest while realising a better contextual understanding of their spatial environment at the same time.

Keywords: Touch screen, MSI, OpenStreetMap, Geospatial web-services

1 Introduction

As the development of mobile technology keeps marching forward, new generation *smartphones* like the Apple iPhone, Google Android, and Nokia Symbian based devices have become increasingly popular (Shek, 2010; Takeuchi and Kennelly, 2010). Position aware micro-sensors (i.e. GPS together with digital compass and accelerometers) are integrated into most of today's state-of-the-art COTS (commercial off-the-shelf) devices together with supplemental locationing technology from Wi-Fi and cell tower signals. As such, these mobile devices are now fully spatially enabled in terms of determining their geo-location with sufficient accuracy and consistency for most outdoor LBS applications (Skyhook, 2010). Since 3G networks became prevalent, fast internet connections have now turned mobile devices into ideal terminals for World Wide Web discovery. Together, these advances bring a variety of new opportunities for development of Location-Based Service applications, which are essentially based on a user's geo-location and delivered as a web-service, for example, "*show me all the coffee shops within 1 km*" or "*what are these buildings around me*".

Most new smartphones are also mobile map ready for navigation purposes, such as Ovi Maps for Nokia phones and Google Maps for Android and iPhones, etc. Using *mashup-maps* is a popular LBS approach, where the results retrieved from the web-service are displayed as annotations on top of the base mobile maps. In this case, the mobile map annotations help users to get to know their *spatial context*; i.e. in relation to the nearby geographic environment surrounding a user's current location, while behind the scenes each request is interpreted as a spatial query (search) to retrieve corresponding geometries and other associated information. The results from the spatial queries are determined by the spatial relationship between the source query shape/window and the spatial data (geometries) repository. Taking the "*show me all the coffee shops within 1 km*" request for instance, it is a type of range query that generates a bounding circle or bounding box with a specified dimension (e.g. radius) as the source query shape. The spatial operator for the query determines whether those geometries (building *footprints* of the coffee shops in this case) have the "within" topographical relationship when compared to the source query shape and subsequently deliver any retrieved results. Currently, there are a couple of well known web-services that deliver such functionality, such as Google Maps and Bing Maps, and applications running on the mobile device built on top of such services, for example, AroundMe (2010) and GeoVector (2010), etc. However, some limitations are still present when performing such types of spatial queries:

1. The results retrieved from such services are entirely dependant on the completeness of the geospatial datasets of the data providers. In some areas, especially smaller cities or rural areas, map coverage is considerably poorer and hence query result accuracies are likewise affected.
2. Even though the bounding query shape has already confined the search space and filtered out those objects that are out of bounds, it assumes that users have at least some local spatial knowledge of their surrounding environment. For instance, if the search radius is too large, in a densely built environment the returned results will be overwhelming, producing display clutter or *information overload*. However, according to Willis et al. (2009), users that use mobile maps for navigation usually have poor local spatial knowledge familiarity. Therefore, if a user is a stranger in a new environment, a 1 km radius query range makes little sense in this context.
3. The query shape is limited to either a bounding box or a circle, and often without the shape's extents actually being displayed to users on the device. In some cases, where users are particularly interested in a certain area, such a query will either include some unnecessary geometries or miss some potential geometries altogether since the users have no freedom to manually adjust the position of the query shape boundaries and move beyond the default rectangle/circle range query parameters.

In relation to the availability and completeness of a geospatial dataset, (Goodchild, 2007) asserts that there is a rising interest in contributing volunteered geographical information (VGI) content to the Web that enriches the amount of accessible spatial data available. One of the more successful VGI examples is the *OpenStreetMap* (OSM, 2010), where the map contents are contributed from volunteers through GPS trajectories and digitized map content, etc. Moreover, OSM is not limited to only street data but also footprints of buildings, point-of-interests (POI), etc. As a result of globally volunteered contributions, the completeness of overall map coverage is growing daily where OSM already shows a significant advantage in this respect over Google Maps and others in many cases (Jacob et al., 2009) and we concur with (Becker and Bizer, 2009) and others that these detailed and free to access spatial datasets offer great potential for developing added-value applications for LBS.

With respect to limitations arising from the restriction of query shapes, a new feature found in some of today's smartphones, i.e. the touch screen, is now available for developing enhanced user interaction functionality on these COTS devices where high precision multi-touch screens can leverage a user's existing experience with human computer interaction (Albinsson and Zhai, 2003; Benko et al., 2006). Even though the screens on mobile devices are smaller than those on a desktop, with high precision selection users can easily perform customized gestures to view map contents, such as pan/zoom/draw etc. Our case in point is the Apple iPhone, which integrates built-in GPS, digital compass, accelerometer, and a multi-touch function enabled touch screen. One of the obvious benefits of using touch screen interaction is that users can now specify a search space by interactively drawing query shapes directly on top of the mobile map display using their finger. This idea mimics using a mouse to draw ad-hoc query shapes for geo-processing in traditional desktop Geographic Information Systems (GIS). On one hand, users can visually see the query shape while interacting with the map contents in real-time. While on the other hand, since it is drawn on top of a mobile map, users get a better awareness of the context of their surrounding environment instead of simply a conceptualized radius.

This paper describes a prototype application, namely *Touch2Query*, which uses the touch screen of an Apple iPhone to perform spatial queries over OSM datasets. It presents the idea using the touch screen as an interface for directly capturing user gestures. Such interaction is translated (interpreted) into source query shapes for further spatial queries in the database. Although at the current stage the geospatial data coverage in OSM is not entirely complete, as VGI moves towards more collaborative input as part of Web 2.0, we see the *collective intelligence* as a promising mobile map and metadata resource for LBS apps. Therefore, this paper also intends to demonstrate the usefulness of OSM data already available for spatial information retrieval in general.

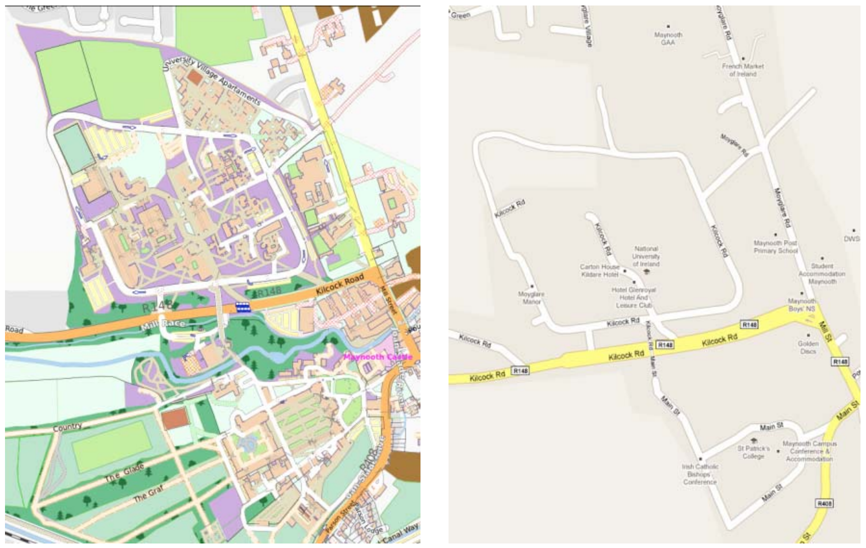
The remainder of the paper is organized as follows: Section 2 focuses on usability and objectives of the *Touch2Query* application from the perspective of geospatial information retrieval on smartphones, where some related applications in this field are introduced. Section 3 gives a comprehensive description of the data integration module, methods and functions that are employed in this application. Detailed demonstration of the implementation on the iPhone is shown in Section 4, and the conclusions and future work is presented in Section 5.

2 Understanding Spatial Information Retrieval for Mobile Devices

2.1 Volunteered Geographical Information (VGI)

The availability of detailed geographic data is critical for delivering comprehensive LBS applications. Most geospatial data, in Europe at least, is collected and controlled by national mapping agencies or private companies, such as Google Maps, Yahoo! Maps, and Bing Maps, etc. However, data coverage over certain areas can still be of considerably poor quality and extent - especially in less populated areas. Some research attempts to overcome this shortcoming looked at imaging and georeferencing public displays of "You Are Here" maps to fill the coverage gap for local navigation purposes (Schöning et al., 2009). But a rapid growth in VGI has also started to fill this gap with OSM being a successful example of this. In relation to VGI, the *citizen-as-sensor* paradigm contributes to OSM by creating, assembling and disseminating geospatial features including streets, highways, buildings, etc. and gradually this collective geospatial information shows a surprising coverage all over the world (Goodchild, 2007; Haklay and Weber, 2008; Haklay and Ellul, 2010). Another vitally important feature of using OSM data is that "*you are free to copy, distribute, transmit and adapt our maps and data, as long as you credit OpenStreetMap and its contributors*" (OSM, 2010), which affords users a lot of commercial freedom when building added value applications on top of it.

Considering this increasing trend of VGI sourced data, this paper aims to demonstrate spatial information retrieval for mobile devices by building upon the base geospatial data supplied from OSM. As a case study, the map data that covers the National University of Ireland, Maynooth (NUIM) and its surrounding areas is targeted. A screenshot of the OSM map coverage is shown in Figure 1(a), while the corresponding map coverage over the same area from Google Maps is shown in Figure 1(b). The detailed OSM data is created by students from NUIM and is available for everyone to use, a clear example of how OSM grows through volunteers contributing datasets (Jacob et al., 2009).



(a) (b)

Fig. 1. (a) Map coverage around NUIM, Ireland from OSM (b) Map coverage for the same area from Google Maps

2.2 Spatial Information Retrieval for Mobile Devices

2.2.1 Query Shapes in Spatial Information Retrieval

By definition, spatial information retrieval involves a query process to determine the spatial relationships between a source query window (shape) and a target data repository, where those objects that satisfy the spatial relationships get returned. For instance, searching for objects (e.g. buildings, streets, etc.) that are within a certain radius from the current location of a user is referred to as a *range query* or *spatial proximity query* in GIS (Mountain and MacFarlane, 2007). In this case, a bounding box or circle with specified radius eliminates those objects that are out of bounds and returns only those that have the "*contains*" spatial relationship when compared to the query shape.

There already a wide range of web-based services that provide such operations, e.g. Google Maps, Bing Maps, etc., and mobile applications built on top of these services. Taking advantage of powerful search engines, these services and applications can already perform queries on large spatial datasets. However, the query shape in each case is still limited to either a bounding box or a bounding circle with a specified radius. An illustration of these two query types is shown in Figure 2.

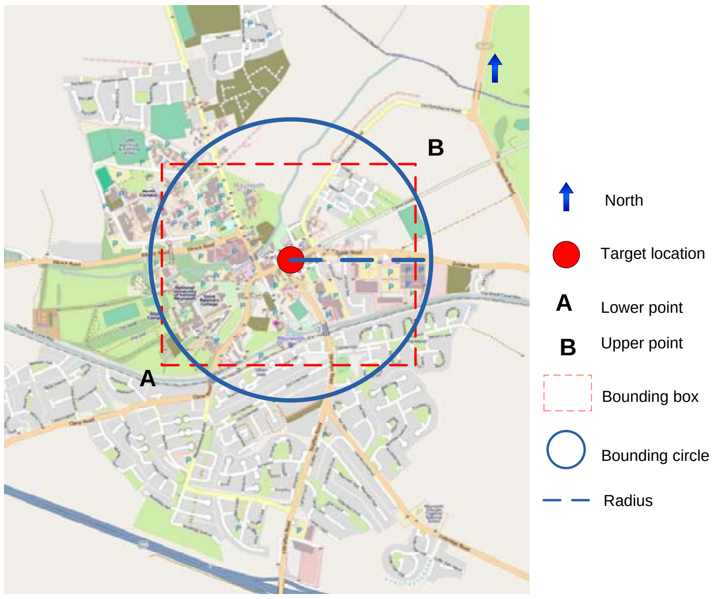


Fig. 2. Using bounding box and circle for range queries

In many cases, both types of query will work to aid users in reducing the information overload problem by filtering information that is out of bounds and thus presumably out of interest. However, if a user is navigating at a large map scale (i.e. zoomed-in to street/building level) in an unfamiliar environment, the specification of radius makes less sense than it would at a smaller map scale (i.e. when zoomed-out). At street level, the users might be particularly interested in nearby areas only where it is hard to know the effect of scale, i.e., the difference between a 100 meter query and 75 meters for instance.

Other innovative applications make use of a combination of integrated GPS and digital compass to perform directional spatial queries where the query shape is a calculated portion of the bounding circle in the pointing direction of the device to simulate a user's actual field-of-view (Gardiner et al., 2009; GeoVector, 2010). A similar approach used in GeoWand (Simon et al., 2007) and 3DQ (Carswell et al., 2010), where returned query results are only the objects visible within the mobile device's pointing direction. Such spatial queries are accomplished by determining whether the pointing vector (a straight line projecting along the pointing direction) intersects with any objects in the database. These approaches aim to involve human interaction when performing the spatial queries and focus on the user's egocentric perspective to improve the query experience. Nevertheless, these approaches rely on a GPS to provide constant and accurate positioning even though any positioning error (of even a few metres) can significantly influence the query results and constant running of the GPS and digital compass has been shown to quickly drain battery life in a mobile device.

We suggest that a true egocentric query experience can never be fully realised if the users themselves cannot fully control the spatial query shape. With fully position-aware mobile devices now available, current location and orientation is a given. Therefore, we conclude that it's now time to provide tools that aid the user to interactively refine their exploration of the environment, instead of "fixing" every query parameter for them. From this point of view, this paper presents the idea of exploiting the touch screen of the mobile device and allow users to draw their own query shapes in their own defined locations (i.e. not limited to their current location) on top of a mobile map. As a result, the customized search space is an arbitrary form. For example, swiping a finger over the mobile map is now considered equivalent to a directional query when pointing the device. The difference is, in this case, the user picks the direction they are interested in querying based on the current map displayed on the device, which may still be positioned and orientated according to the device's sensor inputs, but not necessarily so. This approach has the added advantage that it removes any requirement for these sensors to be absolutely accurate – a common but often false assumption among many existing mobile directional query type applications.

2.2.2 Spatial Database and Web Services

Retrieving comprehensive and meaningful results from spatial queries involves considerably large amounts of data and data processing. Considering the physical limitations of today's mobile device battery power, CPU speed, and data storage space, etc., it is impossible to accomplish all these tasks efficiently on the device itself. Therefore, a client-server architecture is the preferred option where calculation intensive tasks are carried out on the server side, leaving the mobile device as a client to take care of sending requests, receiving responses, and displaying query results to the users (Simon and Fröhlich, 2007; Shek, 2010).

To deal with the large amounts of geospatial data on the server, a spatial Database Management System (DBMS) is required to store, index and manage these data. Most databases today can fulfil this role, such as Oracle Spatial 11g, PostGIS, SQLite Spatial, MySQL, etc. (Zhou et al., 2009). Each of these databases has mechanisms to build spatial indexes on the stored geospatial data which significantly speeds up the query process of discovering spatial relationships (e.g. cover, contain, intersection, etc. between two geometries). In *Touch2Query*, to embrace the possibility that 3D data might be soon available in VGI, we employ Oracle Spatial 11g for this application as it already supports 3D geometries (Ravada and Kothuri, 2009). Although the integrated SQLite database on the iPhone is not spatially enabled (i.e. "geometry" is not a native data type), it can still be used to store shapes of geometries in the form of lists of coordinates and can attach other customized information, such as web links, descriptions, etc. In this case, if datastores are pre-loaded, such an approach can save on Internet connection and other data traffic costs.

Otherwise, a client-server communication architecture is a key element of successful LBS implementations. In this respect, a common method is to use a *request-response* style, which are completed by remote procedure calls (RPC). The *Touch2Query* server delivers Representational State Transfer (RESTful) web-services using TurboGears (tgws 2010), which refers to an architecture where the request is an *http* call and the response can be in different data formats such as XML or JavaScript Object Notation (JSON). A successful example is the Google Ajax search API where the remote call is an http request with specified parameters embedded in the web link and the response data is in JSON format (Google Ajax, 2010). In particular, we utilise a JSON response on the iPhone side simply because we believeve the data format is easier to work with.

We exploit the strategies mentioned above to accommodate customized spatial query shapes. However, in relation to working with OSM data, there are already some useful web-services available. For example, (GeoNames 2010) provides the service *findNearbyStreetsOSM* which finds the nearest street to a given coordinate in the OSM datasets and provides both a JSON and XML response. There are also other interesting services available that the *Touch2Query* prototype utilises to demonstrate the integration of rich and free online resources built on top of OSM data.

3 *Touch2Quey* Application Architecture

The aim of this section is to give a detailed description of the architecture of the *Touch2Query* prototype. An overview of the system is shown in Figure 3. The architecture as a whole consists of three parts: Data integration model, query shape control model, and data exchange model.

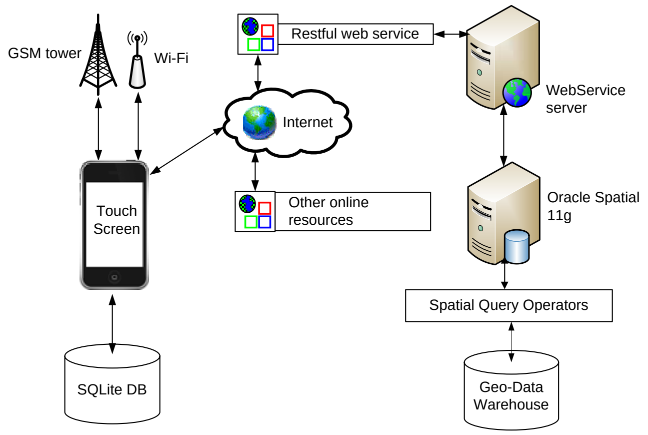


Fig. 3. Architecture of the *Touch2Query* system

3.1.1 Data Integration Model

The spatial data queried in this paper is downloaded directly from OSM, which covers our test area of NUIM and surrounding area, as shown in Figure 1(a). The geometries are in two data types: polylines for highways and streets; and polygons for buildings.

The data integration model consists of three parts: On the server side, both polylines and polygons are stored in the spatial database in two separate tables, where each feature has its corresponding non-spatial attributes (e.g. name, type, etc.) attached. The second part of the data model utilizes the existing online web-services for OSM data, GeoNames in particular. Even though the actual data is stored in the GeoNames server, it is downloaded from the same up-to-date OSM repository. These two cases need Internet connections for data exchange. The third part of the data integration model uses the integrated iPhone SQLite database to locally store detailed information of each building on campus, such as building shape, name, description, etc. There are total 144 building footprints in the NUIM campus; each individual shape is recorded as a collection of coordinates for vertices in the footprint polygon. Therefore, once the building names in the query results get identified, the displayed contents can be derived locally from the SQLite database to save Internet data costs. Another benefit in using SQLite is that a simple range query can be performed using the bounding box; of course in our case the bounding box is user specified using the touch screen. Therefore, a minimum bounding rectangle (MBR) of each building is pre-calculated and the lower and upper points of each MBR are attached as attributes respectively.

3.1.2 Customized Shapes Control Model

For actual spatial query processing, query results are determined according to each spatial operator the user specifies (e.g. cover, intersect, contain, etc.). The spatial operator filters out those objects that do not have the specified spatial relationship with the source query shape and hence aims to reduce the effect of information overload and offer the user a clearer understanding of the surrounding environment.

As mentioned, in *Touch2Query*, query shape specification is not limited to a bounding box or a range radius. Instead, it is an arbitrary shape specified by the user and is drawn interactively via the touch screen. The implemented drawing functions support point, polylines, polygons and circles. To be more specific: a point is when the user touches a specific object (e.g. building) in the mobile map; a polyline is when the user swipes the screen over one or several objects in the map; a circle uses the first touch as the centre and moves to adjust the radius while keeping the finger in contact with the screen; a polygon is drawn when the user touches the screen multiple times in various locations on the mobile map.

The implementation of these query interfaces is relatively straight forward. We attach two layers to the touch screen: one at the bottom is the *map layer* and one at the top is an *empty layer*, which is used to capture any touch gestures from the user. The map layer consists of the iPhone integrated Google Maps. Considering the fact that Google Maps has poor coverage over the test area, we use OSM as our base map. For faster display, a group of geo-referenced OSM tiles are used as overlays on top of the background map. These tiles are split into smaller tiles and stored according to different map zoom levels and behave transparently as part of the native map. The most important role for the top layer is that it interprets the touch point into a latitude/longitude coordinate in the underlying base map. Once the user double-taps the touch screen, the shape is drawn and ready for query processing to begin.

3.1.3 Data Exchange Model

The data exchange model of the system is split into two parts (online and offline) and shown in Figure 4:

1. Web-Services based: The spatial queries over the OSM data from the GeoNames and *Touch2Query* server is accomplished online via RESTful web services, where the function call from the mobile device is a simple http request with specified parameters, such as the coordinates of the query shape; the response data from the server is in JSON format.
2. Local SQLite based: In offline mode, the spatial query using a simple query shape is carried out on the SQLite database. A detailed example query process is described in the following section.

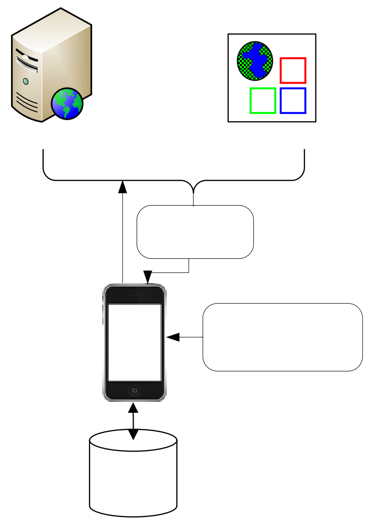


Fig. 4. *Touch2Query* data exchange model

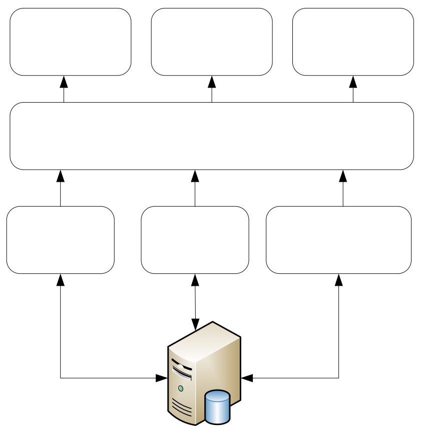


Fig. 5. The *Touch2Query* web-service

In relation to the functions implemented in the *Touch2Query* server, three types of spatial operators are provided, i.e. "*touches*", "*intersects*" and "*covers*". To be more specific: In case the query shape is a polygon, the "*touches*" operator treats the polygon as an "outline" instead of a plane and returns those objects (buildings or streets) that intersect with the boundary only. While, "*covers*" treats the shape as a plane and returns those objects that are within the query shape. When the query shape is a circle, it is equivalent to a range query with the radius visible and manually adjustable in our case. When the query shape is specified as polylines, those objects that "*intersect*" with the line will be returned. For instance, a swipe gesture in this case will generate a straight line, which is equivalent to a direction query. The diagram of the *Touch2Query* server shown in Figure 5 is built on top of the *tgws* framework and provides three types of responses (JSON, SOAP and XML) automatically.

As for integrating web-services from GeoNames, the *find nearest intersection OSM* service is adopted where once a point is located in the mobile map, users can use this query to find the nearest street and the next crossing street from the specified location.

As the integrated SQLite database on the iPhone is not spatially enabled, there are no native spatial operators supported. However, with help from pre-calculated MBRs of geometries, simple spatial operators can be managed via SQL selections (Rubin, 2006). In particular, in *Touch2Query*, two types of query shape are supported: point and bounding box (rectangle). The implemented spatial operators determine the topological relationship between the query shape and each MBR of buildings in the database and retrieve the corresponding data.

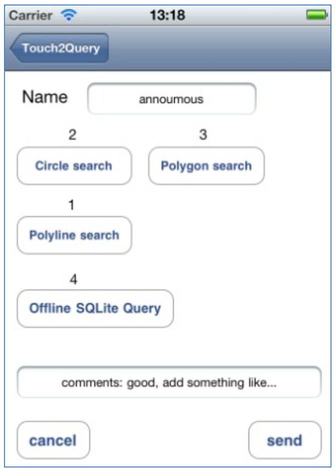
3.2 Helper Utilities

3.2.1 Basic Geographic Measurement Utility

When performing spatial queries, it is common that users are promoted to specify a search radius, which helps to eliminate the risk of inefficient querying over an entire huge dataset. However, such procedures assume the users have local spatial knowledge familiarity, for instance, understand roughly how far 1 km is on the map. *Touch2Query* avoids this assumption not only by letting users specify the radius by drawing the query shape directly on top of the area of interest, but also providing two geographic measurement tools: measurement for area (polygon) and length (polyline). For example, when specifying a circle or bounding box, the radius or width and height in ground units (e.g. metres) gets updated in real-time respectively.

3.2.2 Utility for User Studies

Even though this application aims to give users flexibility while interacting with their spatial environment, query choices are still limited to those that are implemented. Therefore, feedback from users regarding how they like or dislike the functionality, which query functions are preferred, and any additional comments, are important and valuable for future improvements. A *Touch2Query* user feedback utility is therefore embedded in this prototype. Users can grade each function on a scale from 1-5 and submit their comments and suggestions. Such feedback information gets received and stored in the database tied to a unique user id. This allows for further user studies analyzing usage patterns based on the preferences they have graded for future functional improvements. A simple interface for the feedback utility is shown in Figure 6.



Name announmous

2 Circle search
3 Polygon search
1 Polyline search
4 Offline SQLite Query

comments: good, add something like...

Fig. 6. Interface for user feedback utility

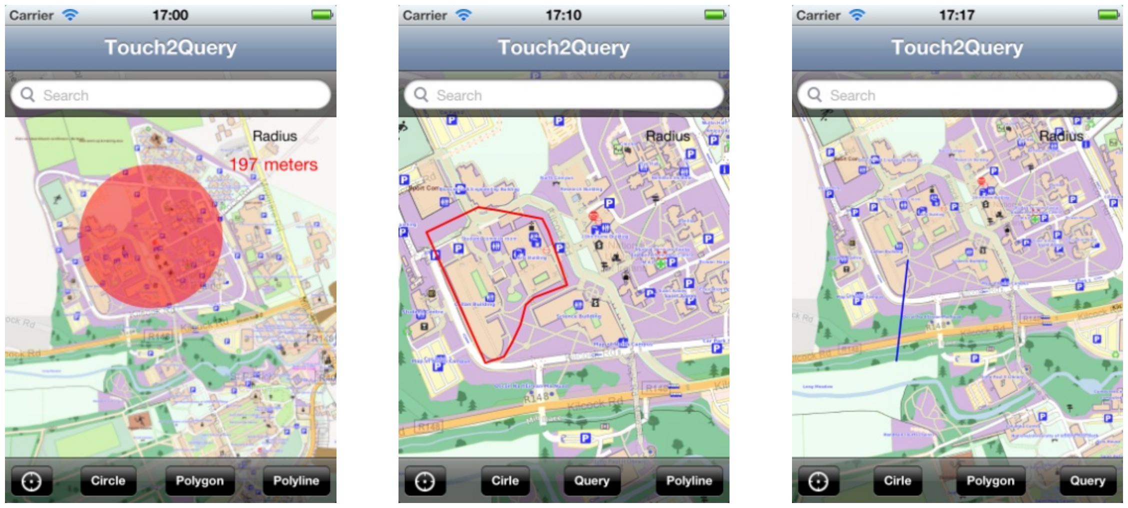
4 *Touch2Query* on the iPhone

This section describes the *Touch2Query* prototype as implemented on the iPhone. The integrated Google Map together with the OSM overlay serve as the base maps, the touch screen is used to specify query shapes, the spatial query functions utilize the services from *Touch2Query* server, GeoNames server, and the integrated SQLite database, and user location and orientation awareness is supported by the integrated GPS and digital compass.

4.1 Touch Screen for Customized Spatial Query Shape

The query interface on the iPhone is shown in Figure 7, where the black transparent bar at the bottom contains buttons for different query options. In particular, the left most button will trigger the device to provide the current location with a single tap on the button; and the orientation the device is currently pointing in with a double tap. Once the digital compass is working for orientation, the base map will auto-rotate to make the map point in correspondence with the pointing direction of the device.

The customized query shape is then drawn on top of the base map, after the user gets informed of their location and orientation. Any finger touches are captured and interpreted into geo-referenced shapes, i.e. each point is translated into a coordinate on the map. The shape is updated in real-time with the touching gestures. In particular, in Figure 7(a), the shaded radius of the circle is continuously updated in meters to let the users know the exact area they are querying; Figure 7(b) shows a query window as an arbitrarily drawn polygon; In Figure 7(c), the user draws a straight line on the screen to search or buildings and streets that are intersected along the path of the line.



(a) (b) (c)

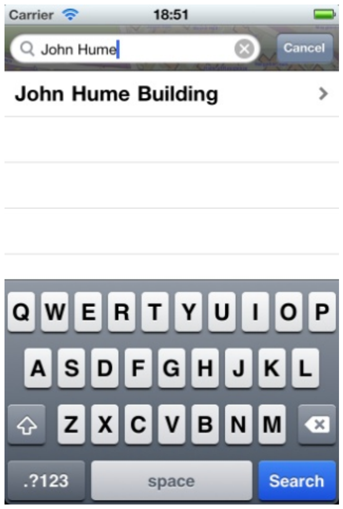
Fig. 7. (a) A query shape using a circle with drawn radius. (b) A query shape in the form of an arbitrary polygon. (c) A straight line drawn using a swipe gesture for the query shape. Note that none of these query shapes need to be necessarily centred on the user's current location and so can be interactively drawn on/over any area of interest.

4.2 Data Integration and Exchange Models

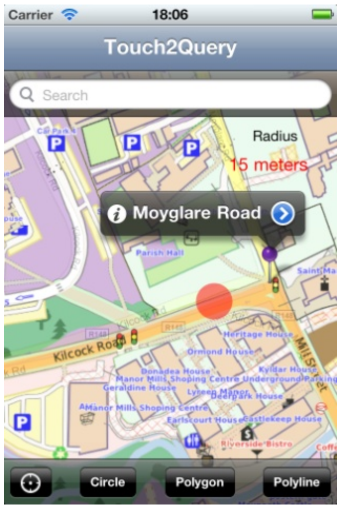
The OSM data is downloaded and stored in the spatial database (Oracle Spatial 11g) on the server side and locally on the iPhone using SQLite database respectively. Since the SQLite does not handle geometry as a native data type, the coordinates of the vertices in a geometry is stored as a collection. In addition, the MBR of each building is pre-calculated and the lower and upper point of the rectangle is stored, as well as other information like web links and description, etc. The spatial query process can be accomplished by either sending requests to the *Touch2Query* and GeoNames web-services (Figure 8(b)), or querying from SQLite locally (Figure 8(a)). As a demonstration, an http request sent to GeoNames about the nearest street intersection with the current street location in OSM is shown as:

```
http://ws.geonames.org/findNearestIntersectionOSMJSON?lat=53.382573&lng= -6.596088
```

It is a simple URL calling the function name "*findNearestIntersectionOSM*", with specified coordinates for the current street location (red circle) and the response data type in "*JSON*". The returned result (purple pin) is then annotated on top of the mobile map, as shown in Figure 8(b).



(a)

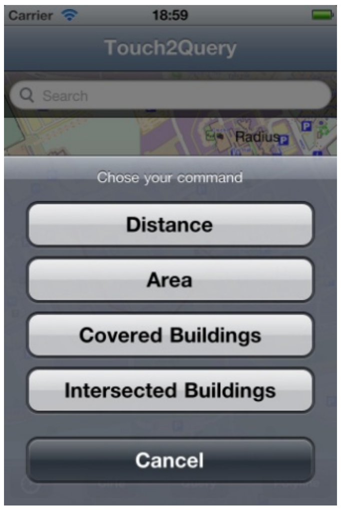


(b)

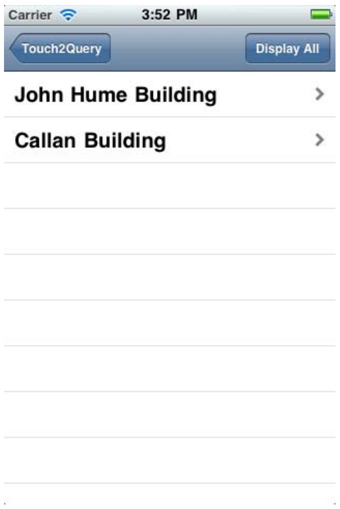
Fig. 8. (a) Searchable building information stored in SQLite. (b) The result returned from GeoNames (purple pin) for nearest street intersection from a given point (red dot)

4.3 An Example of Execution Flow

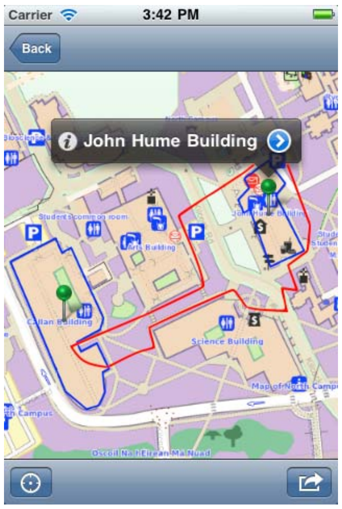
This section demonstrates a spatial query using polygons in the *Touch2Query* web-service with a user drawn polygon as a query shape. After specifying the query shape, users are provided with certain query options as shown in Figure 9(a). Taking the "*Intersected Buildings*" option for instance: an http call using the coordinates of the polygon shape is sent to the *Touch2Query* web-service; the server side uses the "*any intersection*" spatial operator to discover any intersected buildings and returns/displays the building names on the iPhone as shown Figure 9(b). Such results can also be visualized as shown in Figure 9(c), where the building footprints are pulled from SQLite (to save on network download time/costs) and drawn in blue.



(a)



(b)



(c)

Fig. 9. (a) Available options for an arbitrary polygon query shape. (b) Returned results shown in a table. (c) Visualized results displayed on top of the mobile map.

5 Conclusions and Future Work

This paper describes our MSI prototype *Touch2Query* which presents the idea of using the touch screen on today's smartphones to perform ad-hoc spatial queries. The query shape is drawn using finger touches for customized shapes such as; circle, polygons, polylines, and points. With the help of location and orientation aware COTS devices, mobile maps, and real-time distance and area measurements, *Touch2Query* gives the users freedom to perform spatial queries on their areas/objects of interest with a clear understanding of spatial context (wrt scale) in their environment at the same time. The potential advantages are:

1. While mobile devices do need GPS and digital compass to show current location and orientation, in *Touch2Query* these two sensors do not need to work constantly or accurately as users can explore and search their surrounding area manually, which can also significantly extend battery life on the device.
2. During the process of query shape (search space) specification, users can visually see what the query shape looks like, exactly what sort of area it is covers, etc. and therefore get a more intuitive impression of their space and gain a better egocentric experience.
3. In general, compared to conventional range queries, *Touch2Query* allows more flexibility and freedom for users to discover their spatial environment while reducing information overload at the same time.

In relation to the data integration model used by this application, geospatial data from OSM is employed not only because its coverage of our NUIM test bed is better than others, but also because of its VGI nature in that such data can be freely accessed and distributed in mobile LBS applications. Similarly, the *Touch2Query* server delivers open accessed web-services via a RESTful architecture and offers responses in XML, JSON and SOAP formats. This means with a simple http request, most touch enabled mobile devices including iPhone, can *plug and play*. In addition, this application also integrates OSM web-services from GeoNames to demonstrate the possibilities of incorporating other freely available online web-services on mobile devices.

Future work will consider the following areas:

1. In relation to the employed datasets for this application, the accuracy of the data shall be investigated and validated. It was found that some building and street names are missing while others are associated with wrong names.
2. Although OSM has been evolving fast as part of the VGI phenomenon, at this stage it is still too early to conclude that OSM will replace its commercial counterparts. It is inevitable that we should embrace the trend and make good use of these valuable resources by integrating them into future applications.
3. Although a user feedback utility is embedded in this application, at this moment *Touch2Query* has not been tested by many users. Once more user feedback information is gathered, a detailed user study about usage patterns will be investigated and summarized.
4. Even though this application shows how touch screens can make mobile maps more interactive in 2D, we are living in a 3 dimensional world. Future work will focus on making 3D objects touchable in the mobile device, including being able to draw 3 dimensional query shapes for spatial information retrieval.

Reference:

Albinsson, P. and Zhai, S. 2003. High precision touch screen interaction. SIGCHI Conference on Human Factors in Computing. p.105-112.

AroundMe (2010). Retrieved from: http://www.tweakersoft.com/mobile/aroundme.html, accessed 2010-09-17

Benko, H., Wilson, A., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. Proceedings of CHI '06. p. 1263-1272.

Becker, C. and Bizer, C. (2009). Exploring the Geospatial Semantic Web with DBpedia Mobile. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4), pp. 278-286, DOI: 10.1016/j.websem.2009.09.004

Carswell, J.D. (2010). 3DQ: Threat Dome Visibility Querying on Mobile Devices. GIM International, Vol.24, (8), 24, August 2010

Gardiner, K., Yin, J. and Carswell, J.D. (2009). EgoViz --- A mobile based spatial interaction system. In Proceedings of the 9th International Symposium on Web and Wireless Geographical Information Systems, pp. 135-152 (Maynooth, Ireland: Springer-Verlag)

GeoNames (2010). GeoNames web services for OSM. Retrieved from: http://www.geonames.org accessed 2010-09-17

GeoVector (2010). GeoVector world surfer. Retrieved from http://www.geovector.com/applications/world-surfer/, accessed 2010-09-17

Goodchild, M.F. (2007). Citizens as sensors: the world of volunteered geography. GeoJournal, 69, pp. 211-221

Google AJAX (2010). Google AJAX search API. Retrieved from: http://code.google.com/apis/ajaxsearch/ accessed 2010-09-17

Haklay, M. and Weber, P. (2008). OpenStreetMap: User-generated street maps. IEEE Pervasive Computing, pp. 12-18

Haklay, M.M.E. and Ellul, C. (2010). Completeness in volunteered geographical information – the evolution of OpenStreetMap coverage in England (2008-2009). Journal of Spatial Information Science

Jacob, R., Zheng, J., Ciepluch, B., Mooney, P. and Winstanley, A.C. (2009). Campus guidance system for international conferences based on OpenStreetMap. In Proceedings of the 9th International Symposium on Web and Wireless Geographical Information Systems, pp. 187-198 (Maynooth, Ireland: Springer-Verlag)

Mountain, D. and MacFarlane, A. (2007). Geographic information retrieval in a mobile environment: evaluating the needs of mobile individuals. Journal of Information Science, 5, pp. 515-530

OSM (2010). OpenStreetMap, www.openstreemap.org, accessed 2010-09-17

Ravada, S., Kazar, B.M. and Kothuri, R. (2009). Query processing in 3D spatial databases: Experience with Oracle Spatial 11g. 3D Geo-Information Sciences, pp.153-173, DOI 10.1007/978-3-540-87395-2

Rubin, A. (2006). Geo/Spatial Search with MySQL. Retrieved from: http://www.scribd.com/doc/2569355/Geo-Distan-ce-Search-with-MySQL, accessed 2010-09-17

Schöning, J., Krüger, A., Cheverst, K., Rohs, M., Löchtefeld, M. and Taher, F. (2009). PhotoMap: using spontaneously taken images of public maps for pedestrian navigation tasks on mobile devices. ACM, Bonn, Germany

Shek, S. (2010). Next-generation Location-Based Services for mobile devices. CSC Grants. Retrieved from http://asset-s1.csc.com/lef/downloads/, accessed 2010-09-17

Simon, R. and Fröhlich, P. (2007). A mobile application framework for the geospatial web. In Proceedings of the 16th international conference on World Wide Web, pp. 381-390 (Banff, Alberta, Canada: ACM).

Simon, R., Fröhlich, P., Obernberger, G. and Wittowetz, E. (2007). The Point to Discover GeoWand. In: 9th International Conference on Ubiquitous Computing (UbiComp 2007), Innsbruck, Austria, 2007

Skyhook (2010). Skyhook location positioning, context and intelligence. Retrieved from: http://www.skyhookwireless-.com/ accessed 2010-09-17

Takeuchi, K. and Kennelly, P. (2010). iSeismometer: A geoscientific iPhone application. *Computers & Geosciences*, 36, pp. 573-575

tgws (2010). TurboGears Web Services, Retrieved from: http://code.google.com/p/tgws/, accessed 2010-09-17

Willis, K., Holscher, C., Wilbertz, G. and Li, C. (2009). A comparison of spatial knowledge acquisition with maps and mobile maps. Computers, Environment and Urban Systems, 33, pp. 100-110

Zhou, Z., Zhou, B., Huang, Q. 2009. Evaluating query performance on object-relational spatial databases, ICCSIT, pp.489-492, 2009 2nd IEEE International Conference on Comptuer Science and Information Technology, 2009, DOI 978-1-4244-4250