A PostGIS-Based Pedestrian Way finding Module Using Open Street Map Data

Jianghua Zheng[1,2]*, Zhangang Zhang[2], Błażej Ciepłuch[2], Adam C. Winstanley[2], Peter Mooney[2] and Ricky Jacob[2]
[1] School of Resources & Environment Science, Key Lab of City Inteligenlizing and Environment Modeling
Xinjiang University, Urumqi 830046, China,
[2] Department of Computer Science, National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland
*Corresponding author, e-mail: itslbs@126.com

Abstract—**Open source GIS (OSG) is a fast developing field. When OSG is combined with Web2.0 and Service Orientated Architectures (SOA) technologies and more applications of Public Participation GIS, it has many advantages over commercial GIS software. Despite this, OSG still needs more improvement in terms of stability and functional integrity. In order to build more robust, more practical, and more functional LBS applications, this research investigates pedestrian-orientated wayfinding, with special requirements as its study topic. We describe some Web 2.0 routing APIs which can be easily used to provide general shortest path planning. However, these APIs cannot provide guidance services for specific user groups with special requirements, such as tourists in small towns. We take Maynooth as case-study. Maynooth is the only University town in Ireland with a population of approximately 20,000. This research uses OpenStreetMap (OSM) as spatial data source. OSM contains very spatially rich dataset. It is stored and managed in PostGIS/PostgreSQL. Through previous work on LBS applications using the CloudMade Routing API and OSM data, we present a Java-based wayfinding module implementing a restricted area version of Dijkstra algorithm. A set of native PostGIS spatial functions are used to improve performance of the routing algorithm. Results from our wayfinding algorithm are presented and compared with those obtained by using the CloudMade Routing API. Our results are promising and show that this special version of Dijkstra algorithm can take advantage of the spatial data stored in OSM. This work provides a base to build more effective pedestrian wayfinding algorithms which can be implemented in open source software and open APIs. This approach provides a feasible and economical LBS solution for small towns, villages and tourism regions outside larger cities.**

Keywords-OpenStreetMap(OSM);CloudMade;PostGIS; Pedestrian; Wayfinding

I. INTRODUCTION

Wayfinding is a key function of Location Based Services (LBS) for various types of users, including vehicle drivers, pedestrians, bicycle riders, and those travelling by public transportation. Most commercial LBS applications contain wayfinding module with standard shortest path planning algorithm and mostly together with various navigational aids, including verbal navigational directions, route description in text, static/interactive maps, animations, additional landmarks, and virtual/enhanced environments. Actually, wayfinding algorithms are the core, especially for pedestrian navigation applications because of complexity of pedestrians' behaviors. For example, Millonig and Schechtner [1] divide application areas of pedestrian navigation into six categories: tourism, business trips, recreational trips, rescue services, military and security operations, and individual navigational aid. Each of the six types of applications might have special characteristics of users' behaviors to be taken into account when they carry out the applications. Namely, navigation services need personalized wayfinding modules to satisfy various users requests based on different types of behaviors. It is obvious that current commer-cial exciting wayfinding applications, such as "Get Directions" function of Google Maps, "Get directions" function of Microsoft Bing Maps, and that of NAVITIME (Japan), which mainly focus on wayfinding services at most general cases, even some them go further and provide services for not only vehicles but also pedestrians or users seeking services of bus transfer. We have to develop our own wayfinding module if we want to realize a wayfinding service to meet the requirements of a special user group. Fox examples, a type of users may want to find shortest path in open-boundary areas/walking areas [2], such as squares, parks and grasslands. Some users may want to choose an optimal route which is mostly covered with shelters as it is raining heavily. Some others might want to choose a most reliable path to avoid being lost on the way to a destination. Though it is convenient and efficient to realize a navigation system by using third party wayfinding APIs, such as CloudMade Routing APIs, and directions in the Google Maps API, to provide navigation services, we have to develop our own wayfinding modules as potential third-party components for various special requirements of users. Open source GIS tools are most suitable for this purpose.

II. BACKGROUND AND RELATED WORK

With wide spreading of Web2.0 and SOA (Service Oriented Architecture) technologies and penetration of Public Participation GIS (PPGIS), the advantages of OSG(Open Source GIS) applications are stronger than ever.

- First, more and more people take part in development of OSG will make the software and applications more powerful in functions. This is virtuous circle.
- Second, since having open architectures, if more and more users are involved in and more and more applications based on OSG are released, the OSG tools will be more robust and practical.

Supported by a Strategic Research Cluster grant (07/SRC/I1168) by Science Foundation Ireland under the National Development Plan and the National Natural Science Foundation of China (No. 40801058).

• Third, together with mashups techniques, it is efficient and low cost to realize applications using OSG software.

• Fourth, it is more feasible and economical to implement creative applications to meet some special requirements and realize personalized services by using OSG software than those of commercial ones.

The gap between OSG software and commercial ones will be greatly narrowed. Since LBS applications generally have rich commercial values, OSG software will benefit a lot to minor enterprises in LBS field.

Richard M. Stallman gives first concept definition of Free Software in form of four freedoms: 1) The freedom to run the program for any purpose; 2) The freedom to study how the program works, and adapt it to your needs; 3) The freedom to redistribute copies; 4) The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. [3] If strictly stick to this definition, CloudMade [9] should not be listed in Table I because it is a product, including a set of map-based APIs, belongs to a company with the same name and one of whose founders is also a founder of OpenStreetMap (OSM). This might be a major reason why CloudMade APIs could utilize OSM data seamlessly. CloudMade provides Navi Studio as a suite of tools to build fully featured turn-by-turn navigation applications on any mobile platform. It also provides CloudMade Routing API (http://developers.cloudmade.com/projects/show/routing-http-api, 2010) for web and mobile applications with similar functions. CloudMade is free for LBS applications in the event that visitor volume of routes or local searches functions is limited to no more than 5,000 per month [4]. This newly added pricing policy limits its commercial applications. In this paper, we list it in components library in Table I mainly because it is free to use for small applications and let users to edit map data which can also be shared or edited by others.

OpenStreetMap is a free map of the entire world and founded by Steve Coast in July 2004. It allows you to view, edit, and use geographical data in a collaborative way from anywhere on Earth [5]. It is a public participation map source and is fast developing since its foundation was established in April 2006. It let users have more choices on styles of spatial data, fresher data, more detailed data of tiny villages, etc. And best of all, all the services are free. OSM data can be well stored and managed by open source SDBMS (Spatial Database Management Systems) such as PostGIS/PostgreSQL[10,11] and MySQL Spatial/MySQL. OSM data can also be easily converted into other widely used spatial data formats, such as KLM (Google Earth) and SHP (ArcGIS). As extension of PostgreSQL, PostGIS is used to support a range of important GIS functionality, including full OpenGIS support, advanced topological constructs (coverages, surfaces, and networks), desktop user interface tools for viewing and editing GIS data, and web-based access tools. Its functions are much like ESRI's SDE or Oracle's Spatial extension. Latest release of ArcSDE 9.3 supports PostGIS layers.

pgRouting [6] is a free, open-source project maintained by PostLBS, which provides core tools for Location Based Services (LBS) as Open Source Software (OSS). Main goal of pgRouting is to provide routing functionality, currently including traditional Dijkstra algorithm, A* algorithm, Traveling Sales Person (TSP) algorithm, Shooting Star algorithm, and Driving Distance calculation function, to PostgreSQL/PostGIS. Pulis and Attard [7] present a LBS prototype, which provides shortest path service considering multiple variables by using PostGIS and pgRouting as extensions of PostgreSQL. Neis and Zipf [8] show that pgRouting is a powerful OSG component for wayfinding. It is seldom to see academic references about the topic, even less about wayfinding, and existing references are mostly recently published.

III. Problems Description

Though CloudMade Routing API is powerful and free of charge for small applications, its original codes are not open so that we cannot edit them for special needs from specific group of users. We cannot use the Routing API to realize wayfinding services for such special requirements, for example, the three cases mentioned in Section 1. Though pgRouting provides more wayfinding strategies than CloudMade does, it is not convenient to use in other OS platforms than Linux. As to develop applications of the three cases mentioned in Section 1, we have to build our own wayfinding modules based PostGIS with OSM data. In this paper, we implement an improved Dijkstra algorithm, which does Dijkstra path planning in a restricted area, as the first step. Its main contributions lie in two parts. One is to build a reusable java module to create road network topology. The other is realization of the restricted area shortest path planning algorithm. How to realize applications of the three cases mentioned in Section 1 will be discussed in the future.

IV. Using PostGIS to Build a Wayfinding Module with OSM Data

OSM data can be flexible stored in PostGIS/PostgreSQL database, which is adopted by our prototype. A reasonable link-node network for path planning is essential to build own wayfinding modules. However, the original OSM data cannot meet the needs. Intersections of roads in OSM are like general basic geometric points stored in database. We need methods to obtain the intersections of roads as the nodes of the link-node network for path computing. This is the first task. The second one is much easier – calculating length of each possible link between nodes. The third is using a certain path planning algorithm to calculate for optimal routes. As to improve quality of service of the being built wayfinding module by reducing the computing time, we want to develop a restrict area shortest path Dijkstra algorithm by using standard PostGIS functions.

We built a reusable Java module (top.jar) to obtain network topology for next wayfinding module. The module can automatically generate link-node relationships, including orientation, length, line geometry and id of link and nodes. After obtaining network topology, we can run optimal path planning algorithms based on it. Since PostGIS does not provide the function to generate network topology up to the latest version 8.4, the module, top.jar, is a useful tool. PostGIS provides many useful functions to fulfill task of building a

wayfinding module with restricted area Dijkstra algorithm. The following lists major functions at each phase and its instance to demonstrate the realization of the wayfinding module.

- To find cross roads

  ST_Crosses(geometry, geometry)

  (http://postgis.refractions.net/documentation/manual-1.3/ch06.html#id2574517)

  SELECT b.osm_id, b.way FROM planet_osm_line a, planet_osm_line b where a.osm_id = 39505657 and ST_Crosses(a.way,b.way);

- To filter a restricted area for path planning

  If we want to obtain the shortest path from point A to point B, calculate the strait distance between them and then use the distance as the radius and the points A and B as centres to form two individual circles A and B. The part of link-node network inside the $A \bigcup B$ is the restricted area for path planning computing.

  ST_distance(geometry, geometry)

  Returns the smaller distance between two geometries.

  ST_max_distance(linestring,linestring)

  Returns the largest distance between two line strings.

  ST_DWithin(geometry, geometry, float)

  Returns true if geometries are within the specified distance of one another. Uses indexes if available.

  ST_Intersection(geometry, geometry)

  Returns a geometry that represents the point set intersection of the Geometries.

  We use above functions to realize the target to obtain the restricted area. Figure 2 shows an example of this phase result (From John Hume Building to Arts Building)

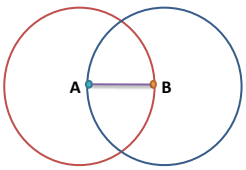


Figure 1. Mechanism of obtaining restrict area

Figure 2. Example of restricted area

- To obtain intersections of roads

ST_Line_Locate_Point (geometry a_linestring, geometry a_point);

-It returns a float between 0 and 1 representing the location of the closest point on LineString to the given Point, as a fraction of total 2d line length. We can use the returned location to extract a Point.

ST_line_interpolate_point(linestring, float location)

-It returns a point interpolated along a line. First argument must be a LINESTRING. Second argument is a float8 between 0 and 1 representing fraction of total LineString length the point has to be located.

S_AsEWKT(geometry)

-It returns a Geometry in EWKT format (as text).

An example: A LineString with the interpolated point at 20% position (0.20) --Return point 20% along a 2d line.

SELECT ST_AsEWKT (ST_Line_Interpolate_Point (the_line, 0.20)) FROM (SELECT ST_GeomFromEWKT ('LINESTRING(25 50, 100 125, 150 190)') as the_line) As foo;

Result is like this:

POINT(51.5974135047432 76.5974135047432)

ST_Line_Substring (geometry a_linestring, float startfraction, float endfraction);

-It returns a LineString being a substring of the input one starting and ending at the given fractions of total 2d length. Second and third arguments are float8 values between 0 and 1. This only works with LINESTRINGs.

An Example: A linestring seen with 1/3 midrange overlaid (0.333, 0.666) --Return the approximate 1/3 mid-range part of a LineString as in Figure 9b.

SELECT ST_AsText (ST_Line_SubString (ST_GeomFromText ('LINESTRING (25 50, 100 125, 150 190)'), 0.333, 0.666));

Result is like this:

LINESTRING (69.2846934853974 94.2846934853974,100 125,111.700356260683 140.210463138888)

- To obtain length of the links in the restricted area

ST_Length(geometry)

-It returns the length of this link in its associated spatial reference.

After this step, we obtain link-node network for computing optimal path.

- Using Dijkstra algorithm to calculate the shortest path in the restricted area

- Output the optimal path and represent it on map

Above is the main procedure to build own wayfinding module using PostGIS with OSM data. We make the module as a java component for sharing. The wayfinding module is programmed in Java (dijkstra.jar). We package it as a jar

library file, which other users can use freely. We can use NetBeans IDE (or other IDE). Then, add the dijkstra.jar file and the PostgreSQL JDBC library to the Libraries for prototype developing.

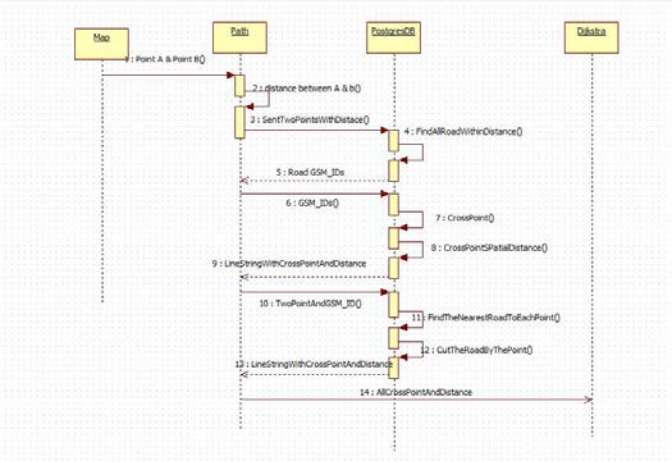
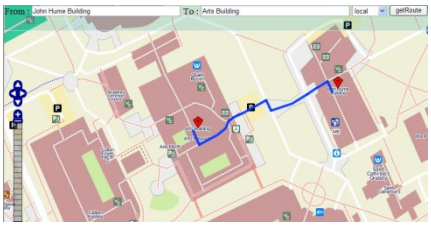
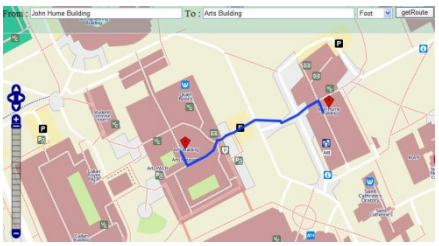


Figure 3. Work flow of the wayfinding module

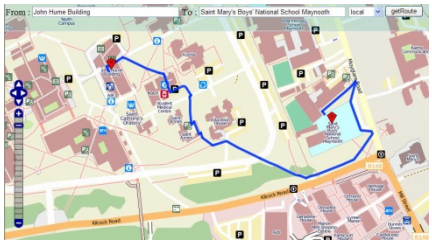
V. Analysis of the Built Wayfinding Module



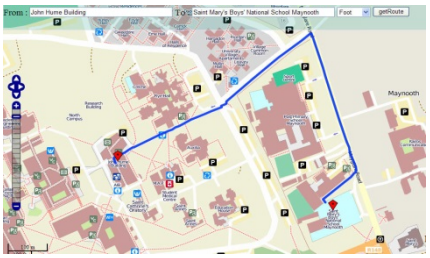
(a) From John Hum Building to Arts Building with own module



(b) From John Hum Building to Arts Building with CloudMade Routing API



(c) From John Hum Building to Saint Mary's Boys' National School Maynooth with own module



(d) From John Hum Building to Saint Mary's Boys' National School Maynooth with CloudMade Routing API

Figure 4. Prototype demonstration

We compare their features by result representations. From Figure 3a and Figure 3b, we can see our wayfinding module has similar path to that from CloudMade Routing API. The minor difference is the former provides centre-line of roads as part of the optimal path while the latter provides boundary of the road because it takes the area as walking area not a road segment. Both of the results are reasonable to the reality however ours is a little longer than that of the latter. Figure 3c and Figure 3d show another case. The former is more complicated though most pedestrians use this path. Figure 3d is simple and it is the shortest path though some us may think the path in Figure 3c is the shortest path by eyes. The major reason why results of CloudMade Routing API are more accurate is our wayfinding module adopts a lossy algorithm. Namely, when we build restricted areas, there may lose some useful nodes and links. This reflects the strategy of using space to exchange for time.

VI. Conclusions and Future Work

In this paper, we present a wayfinding module with implementing Dijkstra algorithm in a restricted area based on our own module for generating network topology. The restricted area is obtained by a built java component, which realized by a set of PostGIS functions. From prototype demonstration and comparison between the newly built wayfinding module and CloudMade Routing API, we find the result of the presented module is effective and efficient though its results might not be the shortest paths. However, this work provides foundation of realizing the special wayfinding requirements of the three cases mentioned in Section 1, which are future work of our research group.

Acknowledgment

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. The authors gratefully acknowledge this support and the support from the National Natural Science Foundation of China (No. 40801058).

References

[1] Millonig, A. and Schechtner, K., 2007. “Developing Landmark Based Pedestrian Navigation Systems”, IEEE Transactions on Intelligent Transportation Systems, vol.8 no.1, pp.43-49

[2] J. H. Zheng, A. Winstanley, A. S. Fotheringham and Z. Pan, “A Two-Level Path Planning Algorithm and Its applications for 2D Pedestrian Navigation with Open Boundary Areas”. Proceedings of LBS2009 Symposium, Nottingham UK, 3rd Sep. 2009

[3] M. Neteler and H. Mitasova, “OPEN SOURCE GIS: A GRASS GIS Approach (Third Edition)”, Springer Science+Business Media, LLC., USA, 2008

[4] CloudMade Web Pricing, EB/OL]. Availbale from http://cloudmade.com/pricing/web [Accessed on 17 Feb. 2013]

[5] OpneStreetMap,[EB/OL].Availbale from http://www.openstreetmap.org/ [Accessed on 25 Jan. 2013]

[6] PgRouting, [EB/OL]. Availbale from http://pgRouting.postlbs.org/ [Accessed on 25 Jan. 2013]

[7] M. Pulis and M. Attard, “Exploring the Shortest Route Options: Applying Environmental Indicators to Calculating Shortest Route”, [EB/OL]. Availbale from http://matthewpulis.info/paper.pdf [Accessed on 25 Jan. 2013]

[8] P. Neis and A. zipf, “Zur Kopplung von OpenSource, OpenLS und OpenStreetMaps in OpenRouteService.org”, [EB/OL]. Availbale from http://www.geographie.uni-bonn.de/karto/publications/pdf/conference/AGIT2008.OpenRouteService.FullPaper.pdf [Accessed on 25 Jan. 2011]

[9] CloudMade, [EB/OL]. Availbale from http://cloudmade.com/ [Accessed on 25 Jan. 2013]

[10] PostGIS, [EB/OL]. Availbale from http://postgis.refractions.net/ [Accessed on 25 Jan. 2013]

[11] PostgreSQL, [EB/OL]. Availbale from http://www.postgresql.org/ [Accessed on 25 Jan. 2013]