Landmark based shortest path detection by using Dijkestra Algorithm and Haversine Formula

Dr. P. V. Ingole, Mr. Mangesh K Nichat
Electronics and Telecommunication Dept. G. H. Raisoni College of Engineering and Management Amravati, India
Computer Science and Engineering Dept. G. H. Raisoni College of Engineering and Management Amravati, India

***Abstract*— In 1900, less than 20 percent of the world population lived in cities, in 2007, just more than 50 percent of the world population lived in cities. In 2050, it has been predicted that more than 70 percent of the global population (about 6.4 billion people) will be city inhabitants. There is more pressure being placed on cities through this increase in population [1].**

With advent of smart cities, information and communication technology is increasingly transforming the way city municipalities and city residents organize and operate in response to urban growth. In this paper, we create a generic scheme for navigating a route through out city. A requested route is provided by using combination of Dijkestra Algorithm and Haversine formula. Haversine Formula gives minimum distance between any two points on spherical body by using latitude and longitude. This minimum distance is then provided to Dijkestra algorithm to calculate minimum distance. The process for detecting the shortest path is mention in this paper.

Index Terms—Haversine Formula, Dijkestra Algorithm, Google Map,XML

I. Introduction

The aim of Paper is to find the route between two places within a city entered by user using the Junctions between Source and Destination junctions. The motto behind it is to improve navigation of user within a city; especially in India where Town Planning policy doesn't follow a standard rules for naming the different places. Most of the times an unknown person can't find even the most famous places within the city due to absence of significant identities. Hence the paper is intended to give an appropriate route to user by directing it through various junctions and roads which will be easily identified by the associated landmarks and a Google map.
The route is given in two parts as:

1) Text route containing route providing a junction to junction movement to user along with the appropriate directions and turnings guiding the user to get the exact intermediate junctions or landmarks
2) Google Map for exact requested route.

Paper uses client-server architecture. Communication between them is strictly in XML for flexibility. The client has user interface from where an input is taken in XML for processing. The server consists of a Java Processing Application and Database for it. The Database used by processing application is a Relational database containing whole information about city. The processing application after parsing request computes route between them with all necessary details with Latitude/Longitude for Google map and sends it as XML response. Client again parsing response gets it on User Interface with Google map processing done in JavaScript.

II. PROBLEM DEFINATION

The Aim of Paper is to find out the route in between two spots/junctions within a city entered by user by making use of the Junctions in between the Source and Destination spots/junctions. The main motto behind it is to improve the navigation of user within a city; especially in Indian cities where Town Planning policy doesn't follow a standard rules for numbering or naming the different spots or places. Most of the times an unknown person can't find even the most famous places within the city due to absence of naming boards or other significant identities. Hence the project is intended to give an appropriate route to user by directing it through various junctions and roads which will be easily identified by the associated landmarks provided with the route.

The requested route is given to user in terms of the junctions present in between the source and destination route along with landmarks and roads connecting them. The Landmarks used in the route may be significant Buildings, Statues, Roads, Complexes, Monuments, Temples, etc. The use of Landmarks adds an advantage of getting to the exact place having no significant identity while travelling through route supplied to user making the application friendly to the user unknown of the city to find out the route in between any two spots or junctions in the city.

The application is bound to give the shortest route providing a junction to junction movement to user along with the appropriate directions and turnings guiding the user to get the exact intermediate junctions (with their significant landmarks) or landmarks in particular areas in between two junctions/spots supplied by user.

LITERATURE SURVEY:

This paper contains “great circle distance” which represents the shortest path for distance modeling and optimal facility location on spherical surface. Great circle distances takes into consideration the geometrical reality of the spherical Earth and offers an alternative to widely held notion that travel over water can be exactly modelled by Euclidean distances. The need for geometrical presentation of the spherical earth becomes very relevant when we take into consideration an ever increasing junctions inside a city. The use of “Great circle distances” opens up another avenue for convergence of Navigation and Spherical Trigonometry into advancement of logistics and facility location. In this paper an evaluation of distance location using great circle distances is used to demonstrate the application of the concept.

This paper proposes and implements a method for performing shortest path calculations taking crowdsourced information, in the form of constraints and obstacles, into account. The method is built on top of Google Maps (GM) and uses its routing service to calculate the shortest distance between two locations. Users provide the constraints and obstacles in the form of polygons which identify impassable areas in the real world.

III. HAVERSINE FORMULA

The Haversine formula is an equation important in navigation, giving great-circle distances between two points on a sphere from their longitudes and latitudes. [4]

These names follow from the fact that they are customarily written in terms of the haversine function, given by haversin (θ) = $\sin^2$ (θ/2).

The haversine formula is used to calculate the distance between two points on the Earth’s surface specified in longitude and latitude.

$$d = 2r\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right)+\cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\psi_2-\psi_1}{2}\right)}\right)$$

d is the distance between two points with longitude and latitude (ψ,φ) and *r* is the radius of the Earth.

Translation to SQL statement[1]

3956 * 2 * ASIN (SQRT (POWER(SIN((orig.lat - dest.lat)*pi()/180 / 2), 2) +COS(orig.lat *pi()/180) *COS(dest.lat * pi()/180) *POWER(SIN((orig.lon - dest.lon) * pi()/180 / 2), 2))) AS distance

IV. DIJKESTRA ALGORITHM

This algorithm is used to find out shortest path between any two junctions. it makes the use of greedy loop to find the minimum distance. Following are the steps to implement Dijkstra’s algorithm:

1. Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes as unvisited. Set initial node as current.
3. For current node, consider all its unvisited neighbors and calculate their distance (from the initial node). For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6+2=8. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
5. Set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step 3.

4.5.1 Pseudo Code:

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:                // Initializations
        dist[v] := infinity                    // Unknown distance function from source to v
        previous[v] := undefined               // Previous node in optimal path from source
    dist[source] := 0                          // Distance from source to source
    Q := the set of all nodes in Graph
    // All nodes in the graph are unoptimized - thus are in Q
    while Q is not empty:                      // The main loop
        u := vertex in Q with smallest dist[]
        if dist[u] = infinity:
            break                              // all remaining vertices are inaccessible from source
        remove u from Q
        for each neighbor v of u:              // where v has not yet been removed from Q.
            alt := dist[u] + dist_between(u, v)
            if alt < dist[v]:                  // Relax (u,v,a)
                dist[v] := alt
                previous[v] := u
    return dist[]
```

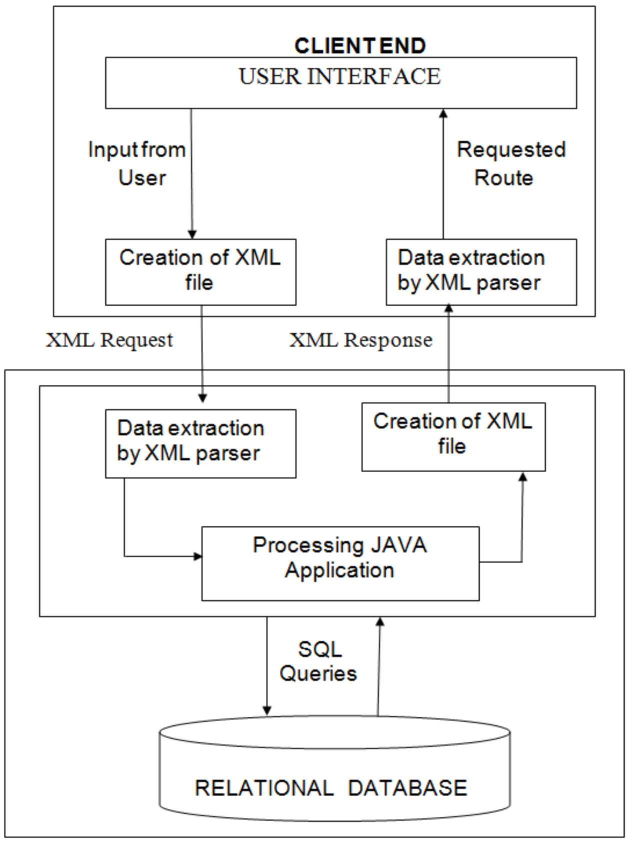
SYSTEM DESIGN

The Aim of the paper is to find out the route in between any two spots within a city entered by the user. This can be implemented using a client-server architecture where a request having two junctions as Source and Destination is sent from client to server and requested route is returned to client as a response from server.

The client-server implementation assumes that the user accesses the functional application remotely from client end to server one. This makes a clear idea of having client at one machine remotely accessing the application and server at the other. Therefore the design includes significant components shown in functional project design below:

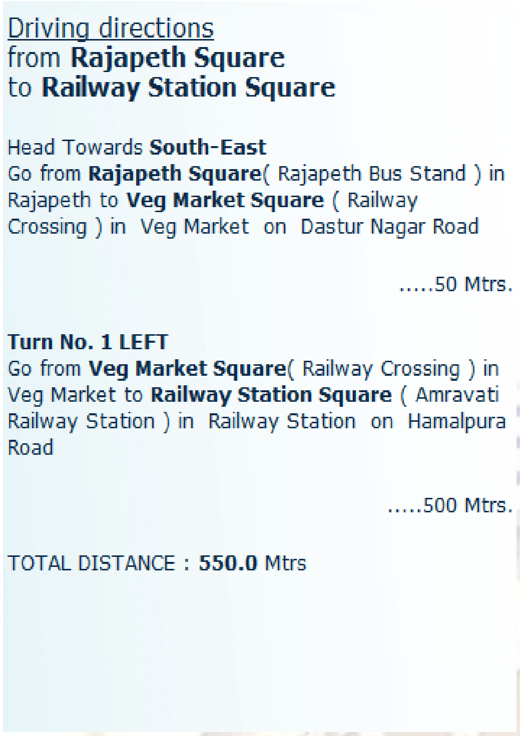
The client end consists of user interface from where an input is taken for processing. The server end consists of a Java Processing Application and Database for it. The processing application basically takes only start and end junctions and computes the route in between them with all necessary details having intermediate junctions with landmarks and roads in a particular area. The Database used by processing application is a Relational database containing whole information about city in terms of junctions, landmarks, roads and areas.

The input containing source and destination junctions for the requested route is sent to the server end as a request from client end. This request is embedded in a XML file can be called as XML request to be sent to server. At server on receiving a XML request; it is supplied to a XML parser for extracting necessary data i.e. source and destination junctions which are in turn supplied to Java Processing application as an input. This application computes a requested route (a shortest one) by interacting with the database using SQL queries to obtain necessary information for computation. For a computed route to be sent to client, it is again embedded into a XML forming a XML response. This response on receiving at client end is again sent to a parser to extract a route to be displayed to the user at to user interface.



SHORTEST ROUTE IN THE FORM OF TEXT ROUTE AND GRAPHICAL WAY:

A user has provision to know the shortest path from source to destination in two ways text based route and graphical route by using Google map. A text based route gives exact way from source to destination in the form of directions, turns, intermediate spots and distance between that spots. A path is given to user by using SQL query. At last it gives the total shortest distance from source to destination.



Driving directions
from **Rajapeth Square**
to **Railway Station Square**

Head Towards **South-East**
Go from **Rajapeth Square**(Rajapeth Bus Stand) in Rajapeth to **Veg Market Square** (Railway Crossing) in Veg Market on Dastur Nagar Road

.....50 Mtrs.

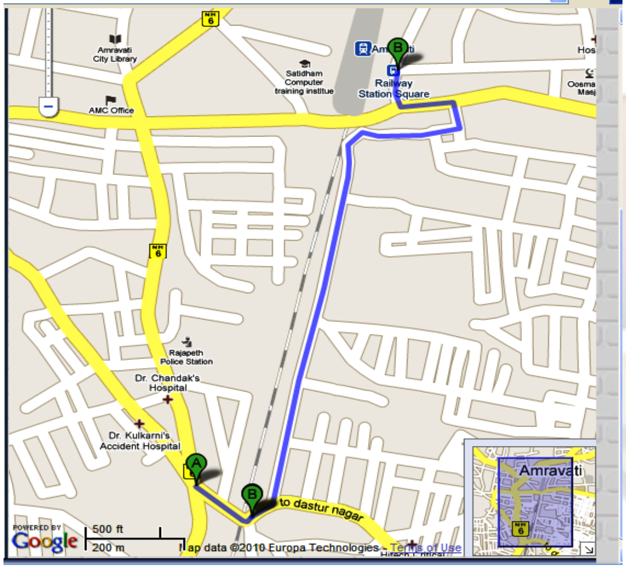
Turn No. 1 LEFT
Go from **Veg Market Square**(Railway Crossing) in Veg Market to **Railway Station Square** (Amravati Railway Station) in Railway Station on Hamalpura Road

.....500 Mtrs.

TOTAL DISTANCE : **550.0** Mtrs

Text Route

Graphical representation of shortest route is shown in figure. It highlighted the shortest route from source to destination. User can use both the techniques to easily know the route between source to destination. GPI provides different methods to access the highlighted route.



Graphical Representation

CONCLUSION

"Landmark Based Routing in Indian Cities" is bound to give the shortest route providing a junction to junction movement to user along with the appropriate directions and turnings guiding the user to get the exact intermediate junctions (with their significant landmarks) or landmarks in particular areas in between two junctions/spots supplied by user. The user also gets exact route with guidance of embedded Google Map.

FUTURE SCOPE

In this paper, we use Djkestra algorithm for determining shortest path between two junctions. But this is common algorithm to calculate the shortest path. Instead of Dijkestra algorithm we can use A* algorithm to calculate shortest path between two city. A* algorithm is combination of Dijkestra algorithm and Breadth First Search algorithm. In addition to that, A* is heuristic in nature.This System can be applied as a navigation syatem which can navigate through out city. Along with intracity shortest path detection ,we can implement same concept for intercity.

REFERENCES

[1] By Mr. Reid "Shortest distance between two points on earth" http://wordpress.mrreid.org/haversine-formula/ This is an electronic document. Date of publishing 20/12/2011.
[2] Samuel Idowu, Nadeem Bari, "A Development Framework for Smart City," Luleå University of Technology,9 Nov. 2012.
[3] Javin J. Mwemzi,Youfang Huang,"Optimal Facility location on spherical surfaces",New York science Journal,April 2011.
[4] Ben Gardiner,Waseem Ahmad,Travis Cooper, "Collision Avoidance Techniques for unmanned Aerial Vehicles",Auburn University, 08/07/2011.
[5] Simeon Nedkov,Sisi Zlatanova"Enabling bstacle Avoidance for Google maps",June 2011.
[6] Bing Pan,john C. Crotts and Brian Muller,"Developing Web Based Tourism Information using Google Map" Departemnt of Huminity and Tourism Mangement,Charston ,USA.
[7] ElinaAgapie.jason Ryder,Jeff Burke,Deborth Estrin,"Probable Path Interference for GPS traces in cities",university of California,2009.