Automatically determine route and mode of transport using a gps enabled phone

Himanshu Gilani

University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation
http://scholarcommons.usf.edu/etd/2897

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Automatically Determining Route and Mode of Transport Using a GPS Enabled Phone

by

Himanshu Gilani

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science Department of Computer Science and Engineering College of Engineering University of South Florida

Major Professor: Rafael Perez, Ph.D.
Ken Christensen, Ph.D.
Miguel A. Labrador, Ph.D.

Date of Approval
March 31, 2005

Keywords: J2ME, Latitude, Longitude, Speed, Longest Common Subsequence

© Copyright 2005, Himanshu Gilani
Acknowledgements

I am grateful to my major professor, Dr. Rafael Perez, for his continuous support and help.

I am also thankful to Sean Barbeau at Center for Urban Transportation Research whose experience was very helpful in completing this thesis.

Thanks are also due to my friend Prodip Hore, who suggested that I investigate the Longest Common Subsequence algorithm.

I am also indebted to my parents, Krishan Murari Gilani and Oma Rani Gilani, and my brother Deepak Gilani for their support and inspiration.
# Table of Contents

List of Tables ........................................................................................................................................... iii

List of Figures ............................................................................................................................................... iv

ABSTRACT ................................................................................................................................................. vi

Chapter 1 Introduction and Related Work ............................................................................................... 1

Chapter 2 GPS Phone ............................................................................................................................... 6
  2.1 GPS Technology ................................................................................................................................. 6
  2.2 2-D Trilateration ................................................................................................................................. 7
  2.3 GPS Accuracy ..................................................................................................................................... 8
  2.4 GPS Phone .......................................................................................................................................... 8
  2.5 Methods of Obtaining GPS Fix ........................................................................................................ 10
    2.5.1 Cell-of-Origin Method .................................................................................................................... 11
    2.5.2 Absolute-GPS Fix Method ............................................................................................................. 12
    2.5.3 Assisted-GPS Fix Method ............................................................................................................. 13
  2.6 GPS Data ........................................................................................................................................... 14
  2.7 Survey Tool Developed ....................................................................................................................... 14
    2.7.1 J2ME Platform ............................................................................................................................... 14
    2.7.2 Motorola Location API .................................................................................................................. 16
    2.7.3 Survey Tool .................................................................................................................................... 16
    2.7.4 Transmitting Data to SQL server ................................................................................................... 17
    2.7.5 User Interface of the Survey Tool ................................................................................................ 18

Chapter 3 Critical Point Algorithm ........................................................................................................... 20
  3.1 Need of Critical Point Algorithm ........................................................................................................ 20
  3.2 Critical Point Algorithm ..................................................................................................................... 20
  3.3 Results of Critical Point Algorithm .................................................................................................... 21
  3.4 Travel Speed ....................................................................................................................................... 25

Chapter 4 Mode Detection Algorithm ..................................................................................................... 26
  4.1 Travel Speed ....................................................................................................................................... 26
  4.2 Significance of Bus Mode Detection ................................................................................................... 27
  4.3 Bull Runner Bus Service ..................................................................................................................... 27
    4.3.1 Hours of Operation ......................................................................................................................... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.2</td>
<td>Bus Stop and Bus Route Database</td>
<td>28</td>
</tr>
<tr>
<td>4.4</td>
<td>Information Used for Bus Mode Detection</td>
<td>31</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Time of Travel and Hours of Operation</td>
<td>31</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Location of Bus Stops</td>
<td>31</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Bus Routes</td>
<td>31</td>
</tr>
<tr>
<td>4.5</td>
<td>Bus Mode Detection</td>
<td>32</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Mode Change Location</td>
<td>32</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Possible Bus Routes</td>
<td>33</td>
</tr>
<tr>
<td>4.5.3</td>
<td>LCS Matching of User Trip with Bus Routes</td>
<td>35</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Calculating LCS Matching Accuracy</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>Mode Detection Algorithm</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Experimental Data</td>
<td>42</td>
</tr>
<tr>
<td>5.2</td>
<td>Accuracy</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>Discussion</td>
<td>46</td>
</tr>
<tr>
<td>5.4</td>
<td>Conclusion</td>
<td>48</td>
</tr>
<tr>
<td>5.5</td>
<td>Future Work</td>
<td>49</td>
</tr>
<tr>
<td>References</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1 Factors Used For Selecting the Phone ............................................................... 9
Table 2.2 Attribute Values Obtained for Each GPS Fix .................................................. 14
Table 2.3 Meaning of Different Attribute Values in GPS Data ...................................... 15
Table 3.1 Percentage of Non-Critical Points .................................................................. 25
Table 4.1 Top Speed Values Used in Mode Detection Algorithm ................................. 26
Table 4.2 Hours of Operation Bull Runner Regular Service ........................................ 28
Table 4.3 Hours of Operation Bull Runner Extended Service ...................................... 28
Table 5.1 Sample Trip Data Used by the Mode Detection Algorithm ............................. 43
Table 5.2 Results of Mode Detection Algorithm .............................................................. 44
Table 5.3 Min, Max, and Avg Time between Two Consecutive GPS Fix in Seconds ....... 46
List of Figures

Figure 2.1 2-D Trilateration Using Three Satellites [6]......................................................... 7
Figure 2.2 Motorola i830 GPS Enabled Phone [10]................................................................. 10
Figure 2.3 Cell-of-Origin Method [11].................................................................................. 11
Figure 2.4 Absolute-GPS Fix Method [11]............................................................................ 12
Figure 2.5 Assisted-GPS Fix Method [11].............................................................................. 13
Figure 2.6 J2ME Architecture [13].......................................................................................... 15
Figure 2.7 GPS Data of a Sample Trip Recorded by the Phone.............................................. 17
Figure 2.8 Overall Network Developed for Transmitting Data on SQL Server............... 18
Figure 2.9 User Interface of the Survey Tool ........................................................................... 19
Figure 3.1 All Locations of a Trip ............................................................................................ 22
Figure 3.2 Flowchart of Critical Point Algorithm ................................................................. 23
Figure 3.3 Critical Locations of a Trip ..................................................................................... 24
Figure 4.1 Bull Runner Regular Service Routes................................................................. 29
Figure 4.2 Bull Runner Extended Service Routes .............................................................. 30
Figure 4.3 Finding Mode Change Location .............................................................................. 33
Figure 4.4 Different Routes Possible from Walk Segment.................................................. 34
Figure 4.5 Same Routes Possible from Walk Segment......................................................... 35
Figure 4.6 LCS Matching of a Bus Trip with the Bus Route .............................................. 36
Figure 4.7 LCS Matching of a Car Trip with the Bus Route .............................................. 37
Figure 4.8 LCS Matching having Good Accuracy ................................................................. 39
Figure 4.9 LCS Matching having Poor Accuracy ................................................................. 39
Figure 4.10 Mode Detection Algorithm ............................................................................ 41
Figure 5.1 Experimental Data ........................................................................................... 43
Figure 5.2 Accuracy of Each Mode .................................................................................... 44
Automatically Determining Route and Mode of Transport Using a GPS Enabled Phone

Himanshu Gilani

ABSTRACT

A system consisting of a GPS-enabled phone and a database has been designed and implemented. This system is capable of recording the route traveled by a user and determining the mode of transport (walk, bicycle, car or bus) used. The Java code running in the GPS-enabled phone automatically records location data, determines critical locations for the trip, and transmits the locations to a central database using the wireless capabilities of the phone. As the route information arrives at the database, it is processed by the mode detection algorithm that determines the mode of transportation being used by the individual for this route. The mode detection algorithm uses travel time, speed, location of bus stops and knowledge of bus routes. The system was tested on experimental data collected from 100 trips (25 trips for each mode of transportation). The correct mode of transport was detected on 97% of the trips. This system can be used by the transportation industry to replace paper-based travel surveys that have been shown to have a number of deficiencies. This includes the inability of the user to recall the events during the survey period, rounding-off errors in reporting time of activities, and requirement to convert data collected to electronic form for further analysis. The system developed will also reduce the burden on the user by automatically determining the travel mode used.
Chapter 1 Introduction and Related Work

In today’s fast-paced world, the transportation capacity of urban environments plays an important role. Many transportation infrastructures are stretched to their limits, as every day an increasing number of people travel on highways. The Transportation Demand Management (TDM) industry seeks to develop strategies for traffic and parking congestion, reducing pollution, increasing mobility of non-drivers and efficient use of existing transport resources. Increasing resources is not always an effective solution; adding capacity to an existing highway may reduce traffic for a particular highway but it will increase down-stream traffic, parking problems, crashes and environmental problems.

Effective solutions to these increased transportation demands require an in-depth understanding of the transportation patterns of the local area. While information about transportation infrastructure and services is readily available, the travel behavior of a population is not. This requires that travel researchers do comprehensive travel surveys. A good travel survey will give accurate information of travel patterns of populations allowing analysis of trip characteristics including start and end times, duration, distance, origin, destination, purpose of trip, and travel mode. These travel surveys can be broadly classified into two categories: Indirect Survey methods and Direct Survey methods.

Indirect Survey methods require independent observers to survey a target population. This approach has the advantage that it does not depend on the subject to
respond. However, this method only captures the evident behavioral characteristics and is limited in space and time. It is limited in space due to the inability of the observer to cover a large area, and limited in time as the target population being surveyed is not available at all times during the survey. To solve the problems of the TDM industry, indirect methods do not help much; thus direct survey methods are preferred.

Direct Survey methods involve interviews with users selected randomly in the target population. Traditionally, these surveys are conducted by paper-pen based personal interviews, postal mail, and phone interviews. During the survey, the user is supposed to complete a questionnaire based on his recollection. These methods thus depend on the ability of the user to recall the events during the survey period and are error prone. Due to its dependence on human memory, accurate spatial information is not available as exact routes taken during the survey are not reported. Users often do not report small and infrequent trips. Direct methods are also subject to rounding-off errors due to the inability of users to remember exact times and their tendency to round-off time while reporting. As a result, accurate spatial and temporal information is not available. In addition, due to the involved nature of direct surveys, interviews are conducted over a short span of 1-3 days. As the span of interview period increases, the probability that a user will not be able to complete the survey increases. This is due to boredom and fatigue. Sometimes a user may refuse to respond and give incorrect answers. Conducting interviews over large spatial areas is also not possible. Thus, these traditional direct survey methods lack adequate reliability. In addition, for long-term storage and future analysis, the data collected must be converted to electronic form. This process requires considerable effort and cost.
Most of the problems of direct survey methods stem from the fact that they put too much of a burden on the user doing the survey. These methods are passive in nature, as the survey data is recorded after the survey is over, which makes them less reliable. Direct surveys can be made more reliable if some kind of active method that reduces burden on the user doing the survey can be developed. With this in mind, the TDM industry seeks to develop computer assisted tools for conducting these types of surveys.

In 1997, the Federal Highway Administration of the U.S. Department of Transportation started a project of GPS-assisted data collection with Battelle Memorial Institute [1]. The main goal of this study was to prove the feasibility of the method. It used a GPS receiver connected by a serial cable to a handheld computer. This study focused only on car trips as both the GPS receiver and the handheld computer required power for operating. In recent years, Geologger [2] has been successfully used for travel surveys.

These studies [1, 2] proved that GPS-assisted methods have better reliability in collecting information about short and infrequent trips. They also demonstrated the ability of these methods to do accurate reporting spanned over multiple days. These systems were also able to report accurate spatial and temporal information of the trips. However, all this came with the cost that users were required to learn how to use the developed tool. In addition, it put a burden on them to carry a bulky GPS receiver wherever they went. The data collected still required manual manipulation even when in an electronic format since no real-time communication with the device was available [1, 2].
To eliminate the manual processing of data, PDA’s having GPS chips [3, 4] were used to develop survey tools that allow communication with the users. PDA’s, being small handheld computers, allow users to do active GPS-assisted data collection by asking them a small set of survey questions before, during and at end of the trips. The purpose of these questions is to obtain information such as purpose of trip, final destination, mode of transport, etc. These questions, however, increased the burden on the user being surveyed.

With advances in technology, GPS receivers started to become smaller in size and have become available in mobile phones. Mobile phones also have the capability of communicating with the users by asking small set of survey questions. One of the goals of this thesis is to determine the possibility of using a phone with GPS capability to record the survey data. Another goal of this research is to take a step towards reducing the burden of survey questions on the user by automatically determining answers to some of the survey questions. With this in mind, an algorithm for automatically determining the mode of transportation used during the trip was developed.

This work focuses on using a commercially available GPS-phone, Motorola i830, for the travel surveys. The user doing the survey will be given the phone and will be required to carry the phone during every trip. The phone will record the spatial and temporal trip data. It will then transfer all the data to a central database using packet data service. Such a device will not put any burden as people are currently used to carrying cell phones for personal communication. This will remove the requirement of carrying bulky GPS receivers or PDA’s for surveys. The data submitted by phone to the central database will then be analyzed on the server using the knowledge of the bus network of
that area to automatically compute the mode of transport. Automatic determination of mode of transport used will reduce the burden on the users to report this during the trip surveys. This system, which is tied wirelessly to a central database server, can be extended in the future using GPS/GIS overlays to automatically answer questions related to trip purpose and final destination. In addition, this system is also capable of running Artificially Intelligent algorithms, which provides feedback to the users by analyzing their travel patterns and suggesting improvements in travel behavior.
Chapter 2 GPS Phone

2.1 GPS Technology

The Global Positioning System (GPS) was developed by the U. S. Department of Defense to provide the military with a means of determining precise locations. It consists of a 24 geosynchronous-satellite constellation, 20,000 Km above the earth in six orbital planes, controlled by five ground stations located at Hawaii, Ascension Island, Diego Garcia, Kwajalein, and Colorado Springs. They constantly monitor the operational health of the satellites and transmit ephemeris constants and clock offsets [5] to the satellites.

For a GPS receiver to locate its position anywhere on the three-dimensional earth, it must receive signal from at least three of the satellites; however, four or more satellites increase the accuracy. Each satellite is broadcasting a specific signal, decodable by the GPS receiver that identifies the satellite. This signal contains information about location and time. All the satellites synchronize operation using atomic clocks, which are very accurate, so that all satellites transmit a repeating signal at the same time. These signals arrive at the GPS receivers at different times and it is possible to estimate the distance to the satellites by calculating the time taken by the signals to reach the receiver. By determining the distances to three or more satellites at the same time and using the satellite locations, receivers can calculate their own location i.e. latitude and longitude. This process is known as trilateration.
2.2 2-D Trilateration

Visualizing trilateration in three-dimensional space is not simple. It is easier to first describe trilateration process in two-dimensional space. To understand how trilateration works, consider GPS satellites to be at the centre of a circle with the radius being the distance of the satellite from the GPS receiver as shown in Figure 2.1. If the distance of one satellite is known, then the possible location of the GPS receiver is somewhere on the circle around the satellite. If the distances from two satellites are known, then the GPS receiver is located on one of the two intersecting points of the circles. Knowledge of a third satellite helps in determining at which intersection point the GPS receiver is located. Knowledge of additional satellites increases the accuracy of the calculated location. Figure 2.1 and [6] explains the trilateration process in two dimensions.

Figure 2.1 2-D Trilateration Using Three Satellites [6]

In three-dimensional space, GPS satellites can be visualized at the centre of spheres. If the distance from one satellite is known, then the possible location is anywhere on the sphere around the satellite. If the distance from two satellites is known,
then the GPS receiver is located at the intersection of two spheres. The spheres will intersect to form a perfect circle; the third sphere will intersect with this circle at two points. Only one of these two points will be on the surface of the earth and this will be the location of the GPS receiver.

2.3 GPS Accuracy

GPS receivers are passive devices that receive signals from the satellites and calculate the position. To receive signals, GPS receivers must be used outdoors and have a clear view of the sky. They consequently often fail to perform in urban areas where tall buildings and overhead bridges obstruct the signal. With the removal of Selected Availability by the U.S government in May 2000, GPS data for civilian uses is accurate to within 25 meters. However, the U.S Department of Defense reserves the right to distort the signal to an accuracy of 100 meters.

2.4 GPS Phone

In this research, a Motorola i830 phone was used for collecting the route data. This phone has an inbuilt GPS-chip and supports Java 2 Micro Edition platform, which allows the programming of this phone. This phone works on Nextel’s network, which has GPS capability. Other networks either do not have GPS capabilities or do not allow access to their GPS capabilities. For this reason, the Nextel Network was the best option for this work. The Nextel network supports Motorola phones using iden technology [7] for developing location-based applications. Within the iden phones, four phones were evaluated. The phones in consideration were Motorola i710, i730, i830 and i860. The following table shows the different factors used for selecting the phone.
Table 2.1 Factors Used For Selecting the Phone

* This price was published on Nextel's website on January 2005

<table>
<thead>
<tr>
<th></th>
<th>Floating-point Support</th>
<th>SDK Support</th>
<th>Price*</th>
</tr>
</thead>
<tbody>
<tr>
<td>i730</td>
<td>No</td>
<td>Yes</td>
<td>$124.99</td>
</tr>
<tr>
<td>i710</td>
<td>Yes</td>
<td>No</td>
<td>$89.99</td>
</tr>
<tr>
<td>i830</td>
<td>Yes</td>
<td>Yes</td>
<td>$149.99</td>
</tr>
<tr>
<td>i860</td>
<td>Yes</td>
<td>Yes</td>
<td>$299.99</td>
</tr>
</tbody>
</table>

Motorola i710 and i730 are similar phones with some minor differences. Motorola i730 supports Connected Limited Device Configuration (CLDC) 1.0 [8], which lacks floating-point capability while the i710 supports CLDC 1.1 [9] with floating-point capability. Since floating-point support was required for mathematical operations on GPS data, the i730 was removed from consideration. Furthermore, Motorola does not provide the Software Development Kit (SDK) for i710, and this phone was removed from consideration. Choices were thus limited to i830 and i860 phones, both of which were CLDC 1.1 devices and had SDK’s. The choice between these two phones was made based on cost, which made the i830 phone the final choice. As both phones are based on the iden technology, the software developed for i830 phone should run with little or no change on i860 and other future iden phones.
Following are the Motorola i830 device specifications and Figure 2.2 shows the Motorola i830 phone.

a. Screen Size: 130 X 130
b. Screen Color: 64K colors, 16-bit
c. J2ME Support: MIDP 2.0 (JSR-118) and CLDC 1.1 (JSR-139)
d. Networking Support: UDP, TCP, SSL, HTTP, HTTPS, Serial, SMS, and AGPS
e. Max Sockets: TCP: 24, HTTP: 8, UDP: 24
f. Heap Size: 1.1 MB
g. Midlet Installation Storage: greater than 2MB
h. Midlet File Storage: greater than 2MB
i. Recommended Midlet size: 500K

Figure 2.2 Motorola i830 GPS Enabled Phone [10]

2.5 Methods of Obtaining GPS Fix

The Motorola’s location API provides three options for obtaining a GPS fix. These methods differ in type of technology used, accuracy and response-times for getting a GPS fix.
2.5.1 Cell-of-Origin Method

In the Cell-of-Origin method, the location API provides the Serving cell latitude and longitude. This method, as shown in Figure 2.3, uses the location of the base station as the location of the mobile device that it serves. Thus, the mobile device will have the same location anywhere in the serving cell of the base station. Accuracy of this method thus depends on the serving cell area, which varies from 1 Km to 30 Km in different regions. If the mobile device moves to any area in which it is not able to establish contact with the serving cell, then this method will not report any location. Obviously, this method has poor accuracy. However, since the cell phone is in constant communication with the base tower that is serving it, the response time in getting a GPS fix is in the order of 1 to 5 seconds.

Figure 2.3 Cell-of-Origin Method [11]
2.5.2 Absolute-GPS Fix Method

The Absolute-GPS Fix method provides autonomous fix from satellites. In this method, time to first fix can increase to three minutes as the GPS receiver has to search the entire frequency space from -4 kHz to +4 kHz and entire code space (1 to 1,023 chips) to locate visible satellites. This method gives better accuracy compared to the cell-of-origin method as it does not depend on base station serving the mobile device. However, the response time is in the order of 1 to 180 seconds. Figure 2.4 shows Absolute-GPS Fix method.

Figure 2.4 Absolute-GPS Fix Method [11]
2.5.3 Assisted-GPS Fix Method

In the Assisted-GPS Fix method, the GPS receiver uses the assist data for locating satellites. The location server on cell phone network sends information to the GPS receiver to assist it in locating the satellites that are in range of the receiver. This information allows the GPS receiver to locate the satellites much faster and reduces time to fix compared to the Absolute-GPS Fix method. If the assist data is not available then fix proceeds without the assist data information. With the Assisted-GPS fix method, location API times out after 32 seconds. Figure 2.5 depicts Assisted-GPS fix method.

Figure 2.5 Assisted-GPS Fix Method [11]
2.6 GPS Data

For every location, the GPS Phone calculates its Latitude, Longitude, Speed, Heading, Altitude, Number of Satellites used, Altitude Uncertainty, Speed Uncertainty, Serving Cell Latitude, Serving Cell Longitude and Date-Time. These attributes can be divided into NMEA GPGGA-1083 standard and non-standard values [12]. Table 2.2 shows different attribute values obtained as part of each fix, and Table 2.3 defines the meaning of all these attribute values.

2.7 Survey Tool Developed

2.7.1 J2ME Platform

For developing survey tool, the Motorola i830 phone was programmed using Java 2 Micro Edition (J2ME) platform. The Application Programming Interface (API) on the J2ME platform is defined by Connected, Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP). Figure 2.6 shows the J2ME architecture.

Table 2.2 Attribute Values Obtained for Each GPS Fix

a. NMEA GPGGA-1083 Standard values

<table>
<thead>
<tr>
<th>Latitude</th>
<th>Longitude</th>
<th>Speed</th>
<th>Heading</th>
<th>Altitude</th>
<th>Satellite Num</th>
</tr>
</thead>
</table>

b. NMEA GPGGA-1083 Non-Standard values

| Altitude Uncertainty | Speed Uncertainty | Serving Cell Latitude | Serving Cell Longitude | Date | Time |
### Table 2.3 Meaning of Different Attribute Values in GPS Data

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latitude</td>
<td>The latitude of a place is the distance of the place to the equator, measured in degrees along a circle between the two poles.</td>
</tr>
<tr>
<td>Longitude</td>
<td>Longitude is defined as the location east or west in reference to the Prime Meridian, which is designated at 0 degrees longitude.</td>
</tr>
<tr>
<td>Speed</td>
<td>This value contains speed in km/hr at which user carrying the phone is moving.</td>
</tr>
<tr>
<td>Heading</td>
<td>This value contains direction in 0-359 degrees where user carrying the phone is moving.</td>
</tr>
<tr>
<td>Altitude</td>
<td>Altitude is the elevation especially above sea level or above the earth's surface. This value gives altitude of the current position in meters.</td>
</tr>
<tr>
<td>Satellite Num</td>
<td>Number of satellites used for obtaining GPS fix.</td>
</tr>
<tr>
<td>Altitude Uncertainty</td>
<td>This value gives the altitude uncertainty in millimeters as the measurement is taken.</td>
</tr>
<tr>
<td>Speed Uncertainty</td>
<td>This value gives the speed uncertainty in kilometers per hour</td>
</tr>
<tr>
<td>Serving Cell Latitude</td>
<td>Latitude value obtained from base station of mobile device.</td>
</tr>
<tr>
<td>Serving Cell Longitude</td>
<td>Longitude value obtained from base station of mobile device.</td>
</tr>
<tr>
<td>Date-Time</td>
<td>UTC Time value at which GPS fix was obtained.</td>
</tr>
</tbody>
</table>

### Figure 2.6 J2ME Architecture [13]

```
<table>
<thead>
<tr>
<th>MIDP Applications</th>
<th>OEM-Specific Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OEM-Specific APIs</td>
</tr>
<tr>
<td>MIDP</td>
<td></td>
</tr>
<tr>
<td>CLDC</td>
<td></td>
</tr>
<tr>
<td>Device Operating System</td>
<td></td>
</tr>
<tr>
<td>Mobile Device</td>
<td></td>
</tr>
</tbody>
</table>
```
J2ME has layered architecture in which CLDC, MIDP and OEM-Specific APIs reside on top of a device operating system. In this architecture, the CLDC defines Java language features and core library features of Java Virtual Machine (JVM). The MIDP provides the application life-cycle model for a mobile device and addresses issues such as user interface, persistent storage, and networking. OEM-Specific APIs address very specific application requirements by providing APIs for features not covered by the CLDC and the MIDP. The J2ME applications developed using CLDC and MIDP only are known as MIDP applications. On the other hand, applications that use CLDC, MIDP and OEM-Specific APIs are known as OEM-Specific applications.

2.7.2 Motorola Location API

Motorola location API is defined as part of the OEM-Specific API for the iden phones. The survey tool developed uses the location API for getting the GPS fix. Due to use of the OEM-Specific API, the survey tool developed will not run on all phones. However, it will run without any problem on the iden phones.

2.7.3 Survey Tool

During the user trip, the survey tool records the location of the user after every 3 seconds. For recording the location, Absolute GPS-Fix method is used. After recording GPS data, the survey tool filters critical locations of the trip using a critical point algorithm. Only critical locations are stored on the phone. The critical point algorithm is described in Chapter 3. Once five critical locations are available, the survey tool transmits data to a web server that stores the data in a central database. Figure 2.7 shows the GPS data of a sample trip recorded by the phone.
Figure 2.7 GPS Data of a Sample Trip Recorded by the Phone

2.7.4 Transmitting Data to SQL server

The Motorola i830 phone has a limited data storage space of 2 MB for application data. Since one GPS fix requires approximately 180 bytes, continuous location tracking at a rate of one fix every 3 seconds will use up all the available storage space in approximately 10 hours. In addition, Motorola i830 does not have any capability to connect to a database server for transmitting data. The only mechanism available is to transmit data to a web server using HTTP protocol [14]; the web server then stores the
data in the database. Due to this, Apache Tomcat 5.0.27 was deployed for receiving coordinates from the phone. The web server code is written using Java servlets, which waits for the phone to establish contact with it, and then it receives data. After receiving data, it makes JDBC connection to MS-SQL 2000 server and stores the data in appropriate tables. Figure 2.8 shows the overall network developed for transmitting data to the MS-SQL server.

![Diagram of network](image)

Figure 2.8 Overall Network Developed for Transmitting Data on SQL Server

### 2.7.5 User Interface of the Survey Tool

The survey tool developed has a User Interface (UI) that allows a user to start a trip activity. Figure 2.9 shows the User Interface of the survey tool. This involves selecting “Record Data” from the main menu of the survey tool. Upon selecting “Record Data,” the “Enter Trip Detail” form is displayed on the phone. In this form, the user needs to enter a trip name and press the “Start” button to begin recording trip data. After pressing the start button, the “Curr. Location” screen is displayed on the phone. This
screen displays latitude and longitude of the current location. Latitude and longitude information on this screen is updated as soon as new location is recorded. At any time, the user can press the “Stop” button on “Curr. Location” screen to finish the trip activity.

Figure 2.9 User Interface of the Survey Tool

a. Select Record Data from Main Menu.

b. Enter Trip Name

c. Scanning current Location

d. Displaying coordinate of current Location
Chapter 3 Critical Point Algorithm

3.1 Need of Critical Point Algorithm

The Motorola i830 phone records speed and direction at every location. This information is used together with location data to determine critical locations that are not very close and where there is a significant direction change. Such critical locations can be thought of as a set of points, which when abstracted by straight lines will form the path of the trip.

In the sample trip shown in Figure 3.1, there are 93 locations. However, not every location is important as at some locations more than one coordinate is recorded, and some locations are very close to previously recorded locations. This increases the storage space required for storing trip data. In addition, since the trip data is transferred using the wireless capabilities of the phone, transferring all the data to the web-server will incur a huge cost, as the cell-phone carrier charges for each byte sent. If the trip data can be reduced such that the original trip can be reproduced from it, then this will lower the data transmission costs and storage space required on the phone.

3.2 Critical Point Algorithm

During the trip, each new location is recorded every 3 seconds. Whenever a new location is recorded, a decision needs to be made whether or not the new location is a critical location in the trip. The first and last points (locations) of the trip are critical as they signify the starting and ending location of the trip. In addition, any point at which there is a significant direction change from the previous critical point and where the
speed is greater than minimum speed is a critical point. The critical point algorithm was originally developed by Barbeau et. al. [15]. Figure 3.2 shows the flow chart of the Critical Point Algorithm used for determining whether or not a new point is critical.

The algorithm depends on two parameters: “Minimum Speed” and “Significant Direction Change”. If the speed is less than Minimum Speed, this indicates that the user is moving slowly and has not traveled a significant distance. In addition, if the direction change between the last point and the current point is less than significant direction change, then this indicates that the user traveled approximately in a straight line. The following values of these two parameters are used in critical point algorithm on the phone:

a. Minimum Speed = 2 km/hr (0.7 m/s)

b. Significant direction change ≥ 10 °

3.3 Results of Critical Point Algorithm

There were 93 points (locations) in the trip shown in Figure 3.1. Applying the Critical Point Algorithm to this trip resulted in determining 17 points as critical. This means that about 82% of the points in the trip were non-critical points. Figure 3.3 shows the critical points of the sample trip (marked with blue color).

The result of the critical point algorithm shown in Figure 3.3 is simply an illustration of how the critical point algorithm helps in reducing trip data. By observing this Figure, it can be easily seen that most of the non-critical points lie on a straight line between two critical points. Such results were found on most of the trips. Table 3.1 shows the percentage of non-critical points calculated by critical point algorithm over 20 trips with 5 trips for each mode. By observing the table 3.1, it can be concluded that the
percentage of non-critical points varies from 40% to 80%. Overall, for trips in table 3.1, 68% of the points were non-critical points.

Figure 3.1 All Locations of a Trip
Figure 3.2 Flowchart of Critical Point Algorithm
Figure 3.3 Critical Locations of a Trip
Table 3.1 Percentage of Non-Critical Points

<table>
<thead>
<tr>
<th>Trip Mode</th>
<th>TripId</th>
<th>Number of Trip Points</th>
<th>Number of Critical Points</th>
<th>Percentage of Non-Critical Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>1</td>
<td>101</td>
<td>32</td>
<td>68.32</td>
</tr>
<tr>
<td>Walk</td>
<td>2</td>
<td>56</td>
<td>9</td>
<td>83.93</td>
</tr>
<tr>
<td>Walk</td>
<td>3</td>
<td>61</td>
<td>20</td>
<td>67.21</td>
</tr>
<tr>
<td>Walk</td>
<td>4</td>
<td>98</td>
<td>28</td>
<td>71.43</td>
</tr>
<tr>
<td>Walk</td>
<td>5</td>
<td>117</td>
<td>41</td>
<td>64.96</td>
</tr>
<tr>
<td>Bicycle</td>
<td>6</td>
<td>89</td>
<td>28</td>
<td>68.54</td>
</tr>
<tr>
<td>Bicycle</td>
<td>7</td>
<td>120</td>
<td>21</td>
<td>82.50</td>
</tr>
<tr>
<td>Bicycle</td>
<td>8</td>
<td>68</td>
<td>39</td>
<td>42.65</td>
</tr>
<tr>
<td>Bicycle</td>
<td>9</td>
<td>107</td>
<td>37</td>
<td>65.42</td>
</tr>
<tr>
<td>Bicycle</td>
<td>10</td>
<td>105</td>
<td>22</td>
<td>79.05</td>
</tr>
<tr>
<td>Bus</td>
<td>11</td>
<td>265</td>
<td>56</td>
<td>78.87</td>
</tr>
<tr>
<td>Bus</td>
<td>12</td>
<td>31</td>
<td>11</td>
<td>64.52</td>
</tr>
<tr>
<td>Bus</td>
<td>13</td>
<td>82</td>
<td>31</td>
<td>62.20</td>
</tr>
<tr>
<td>Bus</td>
<td>14</td>
<td>92</td>
<td>17</td>
<td>81.52</td>
</tr>
<tr>
<td>Bus</td>
<td>15</td>
<td>202</td>
<td>29</td>
<td>85.64</td>
</tr>
<tr>
<td>Car</td>
<td>16</td>
<td>89</td>
<td>23</td>
<td>74.16</td>
</tr>
<tr>
<td>Car</td>
<td>17</td>
<td>111</td>
<td>15</td>
<td>86.49</td>
</tr>
<tr>
<td>Car</td>
<td>18</td>
<td>155</td>
<td>22</td>
<td>85.81</td>
</tr>
<tr>
<td>Car</td>
<td>19</td>
<td>40</td>
<td>8</td>
<td>80.00</td>
</tr>
<tr>
<td>Car</td>
<td>20</td>
<td>200</td>
<td>53</td>
<td>73.50</td>
</tr>
</tbody>
</table>

3.4 Travel Speed

The mode detection algorithm that is described in Chapter 4 uses speed information to determine mode of transport. Since the speed information of non-critical locations will not be available on the server, it will not be a good idea to use only the speed information of critical locations, as in this case, mode detection will be based on 20-30% locations of the original trip. To consider speed of all the locations in a trip, the software on the phone calculates the travel speed over the critical point segment formed by two consecutive critical points. This is done by calculating the distance between two critical points by adding the intermediate distance of all the non-critical points. Then the time difference between two critical points is used to calculate the travel speed between two critical points.
Chapter 4 Mode Detection Algorithm

4.1 Travel Speed

Travel speed can be used for identifying walk and bicycle mode, as both these modes have a limitation on possible top speeds. As a result, travel speed gives the most important information for determining the mode of transport. If the travel speed of a user trip is below top speed of the walk or the bicycle mode at all times, then the walk or the bicycle mode is possible. However, travel speed needs to be used carefully as both bus and car can travel below the top speed of a walk/bicycle mode due to traffic and speed restrictions.

Possible top speeds of different modes were determined statistically by using 10 trips for each mode. For use in mode detection algorithm, top speeds of walk and bicycle modes were chosen 2 km/hr above their possible top speed. The value of 2 km/hr was added to the possible top speed to make sure that no user trip is misclassified if the user travels slightly above possible top speed. Table 4.1 shows the top speed values used by the mode detection algorithm.

Table 4.1 Top Speed Values Used in Mode Detection Algorithm

<table>
<thead>
<tr>
<th></th>
<th>Top Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>10 km/hr</td>
</tr>
<tr>
<td>Bicycle</td>
<td>22 km/hr</td>
</tr>
<tr>
<td>Bus/Car</td>
<td>&gt; 22 km/hr</td>
</tr>
</tbody>
</table>
4.2 **Significance of Bus Mode Detection**

Walk mode can be easily detected using only travel speed information since the speed at which a user walks is far lower than the speed of other modes. However, travel speed does not allow making a decision between bicycle, car and bus modes. As a result, the mode detection algorithm developed attempts to do bus mode detection before bicycle and car mode detection to simplify the decision-making. If the mode detection algorithm is able to conclude that the user trip is a bus trip, then it does not need to test for determining the possibility of bicycle or car mode. If the bus mode detection fails, then either a car trip or a bicycle trip is possible. Due to exclusion of bus mode, it becomes easy to make a decision between car mode and bicycle mode using travel speed. Due to the significant role played by bus mode detection in determining mode of transport, it is described in the next section.

4.3 **Bull Runner Bus Service**

The mode detection algorithm developed requires knowledge of a bus network on which a user will be taking bus trips. As a result, for use in the mode detection algorithm, knowledge of the Bull Runner bus service that operates at the University of South Florida (USF), Tampa was collected. Data collected include hours of operation, bus stop locations, and bus routes.

4.3.1 **Hours of Operation**

The Bull Runner bus service is divided into regular service and extended service. Under regular service, five routes operate on weekdays. The Bull Runner extended service operates in evenings and on weekends. Under the extended service, two routes
that operate in regular service are not serviced. In addition, routes being serviced have extended routes to cover the users of routes that are not in operation. Table 4.2 and 4.3 illustrate the hours of operation for Bull Runner regular service and extended service, respectively.

Table 4.2 Hours of Operation Bull Runner Regular Service

<table>
<thead>
<tr>
<th>Routes</th>
<th>Days of Week</th>
<th>Hours of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D and E</td>
<td>Mon – Fri</td>
<td>7:00am-5:30pm</td>
</tr>
<tr>
<td>A, B, C, D and E</td>
<td>Sat-Sun</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Table 4.3 Hours of Operation Bull Runner Extended Service

<table>
<thead>
<tr>
<th>Routes</th>
<th>Days of Week</th>
<th>Hours of Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AX, CX and DX</td>
<td>Mon – Thu</td>
<td>5:30pm-midnight</td>
</tr>
<tr>
<td>AX, CX and DX</td>
<td>Fri</td>
<td>No operation</td>
</tr>
<tr>
<td>AX, CX and DX</td>
<td>Sat-Sun</td>
<td>2:30pm-9:30pm</td>
</tr>
</tbody>
</table>

4.3.2 Bus Stop and Bus Route Database

A bus stop database having locations of all the bus stops was prepared. In addition to location information, each bus stop location has information about bus routes that service it. Also, each bus stop stores information about where in the bus route this stop is located. A database having routes of all the buses was also prepared. The routes were prepared such that there is a location in a bus-route after every 30 meters. Figure 4.1 shows the routes of the Bull Runner regular service, and Figure 4.2 shows the routes of the Bull Runner extended service. From Figures 4.1 and 4.2, it can be observed that having a location in a bus route after every 30 meters allows the streets traveled by the user to be outlined with sufficient accuracy.
Figure 4.1 Bull Runner Regular Service Routes
Figure 4.2 Bull Runner Extended Service Routes
4.4 Information Used for Bus Mode Detection

Various types of information such as time of travel, hours of operation of buses, location of bus stops, and path of bus routes are used together for identifying the bus mode.

4.4.1 Time of Travel and Hours of Operation

By comparing time of travel with hours of operation, it can be determined whether or not the bus mode is possible. Naturally, if buses are not in operation when the user is taking the trip, then the bus trip is not possible. In addition, for Bull Runner service, using time of travel and hours of operation routes that are in operation (regular or extended) can also be determined. This reduces the number of bus routes that need to be tested. If buses are not in operation at time of travel, then the bus trip is not possible.

4.4.2 Location of Bus Stops

If time of travel and hours of operation indicate possibility of bus mode, then the bus routes possible can be determined by using location of bus stops. For determining possible bus stops, assume at the start of each trip there is a small walk segment. Details of how this walk segment is found are provided in later sections. During the walk segment, the user walks to the bus stop location from where he or she will catch the bus. By comparing all the locations in a walk segment with all bus stop locations in the bus stop database, bus stops reachable from the walk segment can be identified. If no bus stop is identified, then the bus trip is not possible.

4.4.3 Bus Routes

If any bus stops are reachable from the walk segment at the start of a user trip, then the user can take trip on bus routes that service these bus stops. For making a
decision whether the user took a bus trip or not, the mode detection algorithm matches the user trip with the bus routes. If the matching results provide sufficient accuracy, then the mode detection algorithm concludes that the user was taking a bus trip.

4.5 Bus Mode Detection

Bus mode detection proceeds in four steps:

a. Determine mode change location.

b. Find possible bus routes on which bus trip can originate.

c. Perform LCS matching of a user trip with possible bus routes.

d. Calculate percentage matching of a user trip with the bus route.

4.5.1 Mode Change Location

For bus trips, users walk to a bus stop location where they board the bus. This can be observed as 4-step process:

a. Walk to a bus stop.

b. Wait for a bus to arrive at the bus stop.

c. Get on to the bus.

d. Get off the bus at the destination bus stop.

According to these steps, a bus trip can be divided into a walk segment and a bus segment. The bus segment starts at a location where the walk segment ends. This is a mode change location as at this location users change form walk mode to bus mode. It is difficult to find the exact mode change location; however, it is possible to find a close location using the travel speed. The user can go above top speed of the walk mode, thus the mode change location is the first location from the beginning of the user trip where
the travel speed is more than walk top speed. However, the mode change location found is not the exact location as the bus usually travels some distance before exceeding walk top speed.

Figure 4.3 illustrates the concept of mode change location. In this Figure, a location having a travel speed 12 km/hr is mode change location; it is the first location in the user trip with a speed greater than 10 km/hr (top speed of walk mode).

Figure 4.3 Finding Mode Change Location

4.5.2 Possible Bus Routes

Once the mode change location is known, the mode detection algorithm compares all the locations in the walk segment formed by it with a bus stop database to find bus stops in a 40-meter distance from the locations in the walk segment. This provides a list of possible bus stops that are reachable from the walk segment. Hence, if a bus trip is possible, then it must originate from one of these bus stops.

Using the list of possible bus stops and the bus stop database, a list of possible bus routes can also be determined. For each bus stop in the list of possible bus stops, a look up operation can be performed in the bus stop database to determine which bus routes a user can catch. This look up operation also gives the location where the bus stop is
located in the route. If a user can possibly take the same bus route from more than one bus stop, then the list of possible bus routes will contain that route from the bus stop that occurs chronologically earlier in the route. Figure 4.4 and 4.5 show examples to illustrate how a list of possible bus routes is determined.

In Figure 4.4, bus stop numbers 1 and 2 are reachable from the walk segment. Bus stop number 1 services route A and bus stop number 2 services route B. As a result, in this case both routes A and B are possible.

![Figure 4.4 Different Routes Possible from Walk Segment](image)

List of possible bus stops = \{1, 2\}  
List of possible bus routes = \{A, B\}

List of possible bus routes = \{Route A from stop 1, and Route B from stop 2\}

In Figure 4.5, bus stop numbers 1 and 2 are reachable from the walk segment. However, in this case route A services both stops. By accessing the bus stop database, it was determined that route A services bus stop location 1 before 2. As a result, route A from bus stop location 1 is a possible bus route.
4.5.3 LCS Matching of User Trip with Bus Routes

After determining possible bus routes, a user trip is matched with all possible bus routes. The Longest Common Subsequence (LCS) algorithm [16] is used for matching the user trip with the bus route. The LCS algorithm uses dynamic programming approach to give an optimal solution for matching elements of two sequences. The result of the LCS algorithm is a longest common subsequence between two sequences.

For matching a user trip with the bus routes, the LCS algorithm is slightly modified. This modification allows the LCS algorithm to work with two sequences: bus route and user trip, both having latitude and longitude coordinates as its elements. With this modification, the LCS algorithm considers locations in the bus route and user trip as matched, if the distance between them is less than or equal to 20 meters. If more than one location in the user trip has a distance less than 20 meters with a location in a bus route,
then the location in the user trip with the least distance from the location in the bus route is considered as matched.

The distance of 20 meters provides flexibility in LCS matching by trying to match a user trip loosely with the bus route. Distance value less than 20 meters can be used but this will reduce matching flexibility by doing tight LCS matching. Figure 4.6 and 4.7 show examples to illustrate how LCS matching is done.

Figure 4.6, shows the type of matching between a bus trip and a bus route. In this case, since the user travels along the path of the bus route, locations in user trip gets matched frequently with locations in bus route.

Figure 4.7, shows the type of matching between a car trip and a bus route. In this case, as the car is not traveling along the path of bus route, few locations in the user trip are matched with locations in the bus route.
4.5.4 Calculating LCS Matching Accuracy

LCS matching provides only the matching information of locations between the user trip and bus routes; some locations in the user trip might not be matched with locations in the bus route. As a result, matching information alone is not sufficient to determine whether the path of the user trip is being matched with the bus route. To make this decision, a percentage of path matched between the user trip and the bus route is calculated. This is done with a heuristic function that operates on consecutive pairs of LCS matched locations.

A pair of LCS matched locations consists of a location in a user trip and a bus route that were matched. Between two consecutive pairs of LCS matched locations, there might be some locations in the user trip and in the bus route that were not matched. The heuristic function developed uses all the unmatched locations in the user trip and the bus route to calculate the trip distance and route distance between consecutive pairs of LCS matched locations.
If the trip distance and the route distance between two consecutive LCS matched locations have a difference of less than 500 meters between them, then all the trip locations between two LCS matched trip locations are considered matched with the bus route. Using this heuristic function, the mode detection algorithm calculates the number of trip locations matched with the bus route. The number of trip locations matched is then used to calculate the percentage of the user trip that was matched with the bus route. This percentage is called as LCS matching accuracy.

The mode detection algorithm uses the LCS matching accuracy for making decisions. If the accuracy is more than 70%, then it considers the user trip to be matched closely with the bus route. This means more than two-thirds of the user trip was matched with the bus route. If the user trip has an accuracy greater than 70% with more than one route, then the route that has maximum accuracy with the user trip is considered as the route used to take the trip. Figure 4.8 and 4.9 show two examples to illustrate how the LCS Matching Accuracy is calculated.

Figure 4.8, shows the LCS matching accuracy calculated when the user followed the bus route. In this case, due to accurate LCS matching throughout the bus route, all the locations in the user trip were matched with the bus route. Such a case is possible when the travels by bus.

Figure 4.9 shows the LCS Matching accuracy calculated when the user did not follow the bus route. In this case, no trip location between location ‘b’ and ‘j’ was matched with a bus route location. This resulted in a low LCS matching accuracy of 33.33 %. Such a case is possible when the travels by car.
Figure 4.8 LCS Matching having Good Accuracy

distance (a→b→c) ≈ distance (1→2→3→4→5) → 3 points matched

... distance (j→k) ≈ distance(20→21→22→23) → 2 points matched

Percentage = 100% (11 out of 11 points)
Conclusion: Mode determined as Bus-Trip

Figure 4.9 LCS Matching having Poor Accuracy

distance (a→b) ≈ distance (1→2→3→4→5) → 2 points matched
distance (b→c→d→e) ≠ distance (5→...→20) → 8 points not matched
distance (e→f) = distance(20→21→22→23) → 2 points matched

Percentage = 33.33% (4 out of 12)
Conclusion: Bus-Trip not possible
4.6 Mode Detection Algorithm

User trip data arrives at the server in a group of five critical locations. If the server waits for the user trip to end, then the mode information will not be available until the trip ends. To determine the mode during the trip, the mode detection algorithm is run every time new data arrives at the server. Thus, the mode detection algorithm is predicting the mode as soon as the data comes to the server. Figure 4.10 shows the flowchart of the mode detection algorithm.

The mode detection algorithm uses the travel speed sent to the server to calculate the percentage of the trip that is below walk and bicycle top speed, called walk percentage, and bicycle percentage, respectively. If the walk percentage is greater than 90% then the algorithm concludes walk mode. If the walk mode is not possible, the algorithm next tests for the possibility of a bus mode using the LCS algorithm. If the LCS matching is more than 70%, then the bus mode is determined. If the bus mode is not possible, then the algorithm tests whether the bicycle percentage is greater or less than 90%. If the bicycle percentage is above 90% then the algorithm concludes the bicycle mode; otherwise, the car mode is concluded. The car mode conclusion is based on the assumption that not many people use other types of vehicles such as motorcycles, transit rail, etc. Due to this assumption, it is safe to conclude the car mode if all other modes are not possible.
Figure 4.10 Mode Detection Algorithm

1. Start
2. Calculate percentage of trip below walk top speed (wp) and below bicycle top speed (bp)
3. If wp >= 90, goto Yes
   - Test for presence of bus-stops in starting segment of trip
     - If bus-stops found, goto Yes
       - Do LCS Matching of Trip with Bus-routes originating from bus stops in starting segment
         - Calculate Percentage LCS Matching (lcsPercentage)
           - If lcsPercentage >= 70, goto Yes
             - Mode = Bus
           - No, goto No
             - Mode = Bicycle
         - No, goto No
           - Mode = Car
4. End

41
Chapter 5 Discussion and Conclusions

5.1 Experimental Data

For testing the mode detection algorithm, experimental data was collected for 100 trips with 25 trips each for walk, bicycle, bus, and car mode. All the trips were taken by this author in the USF area. For all the trips, a record was maintained that describes the mode of transport used for that trip. This information helps in calculating the accuracy of the mode detection algorithm.

Before starting the trip, the author waited for the GPS receiver to obtain the first fix. This was done as the survey tool developed uses the Absolute GPS fix method that requires time in the order of 1 to 3 minutes for the first GPS fix. After the first fix, as locations of the satellites are known to the GPS receiver, subsequent GPS fixes require times from 1 to 15 seconds. As a result, the GPS data will not be recorded for 1 to 3 minutes at the beginning of the trip if the trip is started before the first fix is acquired by the GPS receiver. In addition, the roofs of cars and buses block the GPS signal. This blockage may result in not being able to record any data if the first GPS fix is not acquired before the trip initiates.

For each trip, all the GPS attribute values described in Table 2.2 were recorded by phone and sent to the server. However, only latitude, longitude, travel speed, heading, date, and time were used by the mode detection algorithm. Table 5.1 shows the type of data used by the mode detection algorithm, and Figure 5.1 shows one experimental trip of each mode.
Table 5.1 Sample Trip Data Used by the Mode Detection Algorithm

<table>
<thead>
<tr>
<th>Trip ID</th>
<th>Critical Point Number</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Travel Speed</th>
<th>Heading</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>28.05879</td>
<td>-82.41545</td>
<td>0</td>
<td>0</td>
<td>10/8/2004</td>
<td>17:28:0</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>28.05959</td>
<td>-82.41510</td>
<td>2</td>
<td>1</td>
<td>10/8/2004</td>
<td>17:38:10</td>
</tr>
</tbody>
</table>

Figure 5.1 Experimental Data
5.2 Accuracy

The mode detection algorithm developed correctly identified the mode of 97 out of 100 trips showing an overall accuracy of the 97 percent. Table 5.2 shows the result for each mode and Figure 5.2 depicts the bar graph showing accuracy of each mode. The misclassified trips were due to the lack of a signal while recording GPS data, limitations of the mode detection algorithm, and errors in the bus stop database used by the algorithm.

Table 5.2 Results of Mode Detection Algorithm

<table>
<thead>
<tr>
<th>Mode</th>
<th>Total Trips</th>
<th>Estimated Correctly</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>25</td>
<td>25</td>
<td>100%</td>
</tr>
<tr>
<td>Bicycle</td>
<td>25</td>
<td>24</td>
<td>96%</td>
</tr>
<tr>
<td>Bus</td>
<td>25</td>
<td>23</td>
<td>92%</td>
</tr>
<tr>
<td>Car</td>
<td>25</td>
<td>25</td>
<td>100%</td>
</tr>
<tr>
<td>All Modes</td>
<td>100</td>
<td>97</td>
<td>97%</td>
</tr>
</tbody>
</table>

Figure 5.2 Accuracy of Each Mode
All the walk trips were identified correctly. These results show that the travel speed alone is sufficient to determine the walk mode. Use of the LCS algorithm for determining bus trips also demonstrated good results with only two bus trips misclassified as car trips. In addition, the strategy of doing bus mode detection prior to bicycle and car mode detection was also successful, as this eliminated misclassification of car trips as bus trips. The success of this strategy is shown by 100% accuracy in car mode detection. However, two of the bus trips were misclassified as car trips and one of the bicycle trip was misclassified as the bus trip.

One of the bus trips was misclassified as a car trip, as no bus stop was reachable from the walk segment at the start of the trip. As a result, the LCS algorithm for matching the trip with the bus route did not start. This was due to an error in the bus stop database used for the bus mode detection. Due to this error, correct bus stop location was not available in the bus stop database.

Another bus trip was misclassified as a car trip due to LCS matching accuracy slightly lower than 70%. In this trip, there were 202 locations. For the first 170 locations, the mode detection algorithm predicted the correct bus mode, but due to poor LCS matching, in the next 32 locations, LCS matching accuracy dropped below 70 %. This resulted in misclassification of bus trip as car trip.

One bicycle trip was also misclassified as bus trip. The route for this trip matched a bus route. The misclassification can be attributed to the availability of a bus stop in the walk segment at the start of the trip which made conditions favorable for bus mode detection.
5.3 Discussion

For recording a trip, the user needs to wait until the first location is recorded by the receiver. This wait is not due to the limitation of the survey tool developed but due to the limitation of the GPS receivers; as they need to search the entire frequency space of -4 kHz to +4 kHz to locate the visible satellites. However, once the GPS receiver obtains the first fix, subsequent fixes require an average of 4 to 6 seconds. Thus, after the first fix, continuous location tracking is possible.

The time to fix during the trip may increase from an average of 4 to 6 seconds if the user travels in a urban area with tall buildings. In addition, electrical wires, overhead concrete bridges, and trees also block GPS signal. In some cases, the environmental conditions may also increase the average time between subsequent GPS fixes. Table 5.3 shows the minimum (min), maximum (max), and average (avg) time* between two consecutive GPS fixes for 10 trips.

Table 5.3 Min, Max, and Avg Time between Two Consecutive GPS Fix in Seconds

<table>
<thead>
<tr>
<th>Trip Id</th>
<th>Min Time</th>
<th>Max Time</th>
<th>Avg. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>86</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>61</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>69</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>59</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>26</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>69</td>
<td>5</td>
</tr>
</tbody>
</table>
Detection of walk mode and bicycle mode can fail in special cases that force the rider to go above the top speeds of their respective modes. These special cases may result if user decides to run or if the bicycle goes down hill.

The mode detection algorithm will misclassify the bus mode if it lacks a GPS signal in the walk segment at the start of the trip. Thus, the mode detection algorithm may fail to find bus stops reachable from the walk segment; as a result, LCS matching will not be initiated. In addition, bus mode detection may fail if a user gets on the bus away from a regular bus stop. In this case, as the user got on the bus at a non bus stop location, the mode detection algorithm will fail; as it will not be able to find bus stops that are reachable from the walk segment. Due to similar reasons, bus mode detection will fail if there is an error in the bus stop database.

Bicycle and car trips may get misclassified as bus trips if the route taken by the user resembles the route of a bus originating form the walk segment. This misclassification is due to the bias of the mode detection algorithm as it tries to detect a bus mode before bicycle and car mode and if the route of the trip is matched with a bus route, the algorithm rejects the possibility of bicycle and car mode.

The detection of car mode is based on the assumption that not many people will use a mode of transport other than walk, bicycle, bus and car. If any other mode of transport such as motorcycle, transit rail, etc is used then the algorithm will misclassify the mode as car mode.

The mode detection algorithm developed can be easily extended for determining the user trip mode in a different geographical area. The only requirement is knowledge of bus stop locations and bus routes for use in the mode detection algorithm.
5.4 Conclusion

Results of this research demonstrate that

a. GPS phone can successfully track a trip.

The survey tool developed as a part of this research can successfully collect spatial trip data. This system can be used by the transportation industry to replace paper-based travel surveys that have a number of deficiencies. These include the inability of the user to recall the events during the survey period, rounding-off errors in reporting time of activities, and requirement to convert data collected to electronic form for further analysis. In addition, the survey tool developed also eliminates the need of costly and bulky PDA’s for the survey.

b. Mode Detection Algorithm can successfully determine mode of transport subject to having knowledge of the Bus Network in the database.

The developed mode detection algorithm demonstrated good results on experimental data of 100 trips collected in the University of South Florida campus area, which operates the Bull Runner bus service. Use of the LCS algorithm for bus mode detection, prior to bicycle and car mode detection using the travel speed makes the mode detection process robust. The mode detection algorithm developed is not limited to the University of South Florida area, and can be used in other geographical areas if knowledge of bus network of that area is available. Results of the mode detection algorithm also eliminate the need for users to report the mode of the trip; thus, it helps to reduce the burden on the user doing the survey.
5.5 Future Work

Reverse geo-coding information obtained from a Geographical Information system (GIS) overlay can be used for enhancing the mode detection algorithm. Information from a GIS overlay can be used to identify bicycle tracks, one way roads, and highways. In addition, knowledge of roads can be coupled together with the speed limits of each road segment and used to increase the accuracy of the mode detection algorithm.

Possible enhancement to the system could eliminate problems where trips have loss of signal for small periods. In such trips, missing data can be interpolated using knowledge of road segments from the GIS overlay.

In the future, the algorithm can also be extended to detect modes of transport used in multi-mode trips. In multi-mode trips, more than one mode of transport is used during one trip. The challenge in this problem is to determine a possible mode change location. This will require knowledge of Park-&-Ride and bike-rack locations that can hint the algorithm to check for the possibility of a possible mode change near these locations. The possibility of mode changes can also be tested at every bus stop on the bus route.
References


Microsystems website, Available online at

[14] “Hypertext Transfer Protocol - HTTP/1.1, RFC 2616,” Available online at

Increasing Transit Ridership by Automatic Collection (TRAC) of Individual Travel
Behavior Data and Personalized Feedback", Center for Urban Transportation Research at
the University of South Florida, April 2005.