GPS CaPPture: A SYSTEM FOR GPS TRAJECTORY COLLECTION, PROCESSING, AND DESTINATION PREDICTION

Terry W. Griffin

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

May 2012

Griffin, Terry W. GPS CaPPture: A System for GPS Trajectory Collection, Processing, and Destination Prediction. Doctor of Philosophy (Computer Science), May 2012, 116 pp., 8 tables, 44 illustrations, bibliography, 120 titles.

In the United States, smartphone ownership surpassed 69.5 million in February 2011 with a large portion of those users (20%) downloading applications (apps) that enhance the usability of a device by adding additional functionality. A large percentage of apps are written specifically to utilize the geographical position of a mobile device. One of the prime factors in developing location prediction models is the use of historical data to train such a model. With larger sets of training data, prediction algorithms become more accurate; however, the use of historical data can quickly become a downfall if the GPS stream is not collected or processed correctly. Inaccurate or incomplete or even improperly interpreted historical data can lead to the inability to develop accurately performing prediction algorithms. As GPS chipsets become the standard in the ever increasing number of mobile devices, the opportunity for the collection of GPS data increases remarkably.

The goal of this study is to build a comprehensive system that addresses the following challenges: (1) collection of GPS data streams in a manner such that the data is highly usable and has a reduction in errors; (2) processing and reduction of the collected data in order to prepare it and make it highly usable for the creation of prediction algorithms; (3) creation of prediction/labeling algorithms at such a level that they are viable for commercial use. This study identifies the key research problems toward building the CaPPture (collection, processing, prediction) system.

ACKNOWLEDGMENTS

It would not have been possible to write this doctoral dissertation without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

I would like to give my gratitude to Dr. Yan Huang, my major professor, for her advice, unsurpassed knowledge, and of course, her patience. Above all, I would like to thank Rachel for her personal support and great understanding.

My parents, Don and Christa Griffn have given me their unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice. Blake, Dax, Desiree, and Mackenzie - thank you! After many nights of questioning my current path, it was the children that helped me persevere.

I am very grateful to Catherine Stringfellow for providing the push and assistance I needed, at the most critical time, to get the job done. The expertise of Nelson Passos and Richard Simpson was instrumental in getting me through much of the perplexing subject matter. I also thank Tina Johnson. We spent many hours driving from Wichita Falls to Denton and back; her friendship made the many hours of driving in a car bearable. I would like to thank Stephanie Deacon for her kindness and support. She consistently goes above and beyond the call of duty. Lastly, I would like to thank Ranette Halverson for encouraging me (and assuring me that I could do it) to start this whole crazy process of earning a PhD.

CONTENTS

CHAPTER 1

INTRODUCTION

In the world as we know it, hand-held computing devices have penetrated our lives. This is tolerated mostly due to the fact that many of the technologies that these devices provide create a comforting level of convenience. In fact, society expects more and more from these small, life augmenting devices. The simple cell phone transformed from just a "phone" into a powerful "smart device" that today has as much computing power as a desktop computer from a decade ago[1]. In fact, a smart device offers quite a bit more functionality than a decade old desktop in a remarkably smaller form. Aside from the common uses (phone, camera, video recorder), "smart devices" are even more powerful with integrated chip sets such as global positioning systems (GPS) to measure geospatial location and accelerometers that can measure a devices orientation.

With the power of these ubiquitous devices growing, users are expecting the software and applications accompanying such devices to provide services to match. On October 22, 2010, for the first time ever, a geo-locating service was used from space [73]. NASA astronaut Douglas H. Wheelock, commander of *Expedition 25*, checked in to the International Space Station, using a geo-location service called "foursquare," showing that even astronauts in space enjoy the ubiquity of hand held devices and the geo-location/social software they provide.

The number of worldwide mobile broadband-enabled subscriptions is rapidly growing and will hit the one billion mark in 2011. According to the latest market data released by ABI Research, there was already over five billion mobile subscriptions globally at the end of 2010. In the United States alone, smartphone ownership surpassed 69.5 million in February of 2011 with a large portion of those users (20%) downloading applications (apps) that enhance the usability of a device by adding additional functionality [66]. A large percentage of apps are written specifically to utilize the geographical position of a mobile device. In an

[1]The average CPU speed and RAM size in a desktop computer in the year 2000 was 1Ghz and 512Mb respectively, the same as the average leading smart devices in 2010.

article written by Andrew Lipsman, outlining the top ten mobile trends of 2010, three of the top ten categories directly relate to utilizing the geographic position of a user and their mobile device. The three overarching trends are 1) location based services, 2) social media, and 3) mobile advertising [2]. Despite the fact that these mobile trends directly support the need for location prediction/labeling, the glaring lack of existing applications that do so is astounding. As of this writing, a single app called "Succotash" is available on the Google Android® platform. "Succotash" allows "friends" to see each others future locations [32]. Industry expects a 28% growth or 6.6 billion wireless subscriptions by the year 2016 (twice the current percentage of users being mobile broadband-enabled) [69]. With the number of individuals becoming broadband-enabled, the question to ask is: "Is there a potential market for mobile location prediction applications?"

While there is a huge upswing in worldwide mobile broadband subscriptions and overarching trends in mobile broadband usage that lean towards location, there is a complete lack of available apps to fulfill these mobile trends. There is software that indeed utilizes location sharing (Foursquare, Facebook, etc.), but the list of available predictive apps is non-existent (or rather 1 item long). John Krumm wrote an article titled "Ubiquitous Advertising: The Killer Application for the 21st Century" [55] showing that ubiquitous advertising is already happening, however, one feature lacking in ubiquitous advertising is the ability to predict location or behavior. Taking all of these factors into consideration, it is not a far stretch to realize that there is a potential revenue generating market that needs filling by researchers. There is plenty of current research in developing prediction models, but this is not translating into commercial software, which would imply that the models themselves are not accurate enough to be implemented commercially.

One of the prime factors in developing prediction models is the use of historical data to train such a model. With larger sets of training data, prediction algorithms become more accurate; however, the use of historical data can quickly become a downfall if the GPS stream is not collected or processed correctly. Inaccurate or incomplete or even improperly interpreted historical data can lead to the inability to develop accurately performing prediction algorithms. As GPS chipsets become the standard in the ever increasing number of mobile devices, the opportunity for the collection of GPS data increases remarkably.

In a book published July 2011, Jiawei Han states that: “[c]omputing with spatial trajectories is still a fairly young and dynamic research field.” Furthermore, “with the rapid development of wireless communication and mobile computing technologies and global positioning and navigational systems, spatial trajectory data has been mounting up, calling for systematic research and development of new computing technologies for storage, preprocessing, retrieving, and mining of trajectory data and exploring its broad applications. Thus, computing with spatial trajectories becomes an increasingly important research theme” [116].

This recent quote reinforces the goal of this study, which is to build a comprehensive system (see figure 1.1) that addresses the following challenges: (1) collection of GPS data streams in a manner such that the data is highly usable and has a reduction in errors; (2) processing and reduction of the collected data in order to prepare it and make it highly usable for the creation of prediction algorithms; (3) creation of prediction/labeling algorithms at such a level that they are viable for commercial use. This study identifies the key research problems toward building the CaPPture (collection, processing, prediction) system. It gives a comprehensive review of previous related work presented in three sections, one for each of the three major parts of the CaPPture system. An overview of the system is presented next along with the investigations and methodologies to address each key research issue. Finally, the current progress and future work in regards to this Ph.D. dissertation are presented.

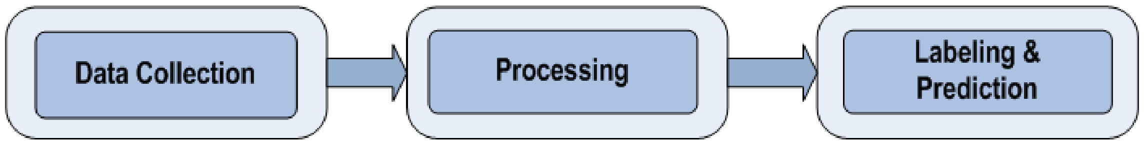


Figure 1.1. A generic framework.

CHAPTER 2

LITERATURE REVIEW

This chapter shows the related work done in the three sub-areas related to the proposed system: CaPPture (collection, processing, prediction). This is a software system that encompasses the entire life-cycle of generating prediction algorithms from GPS traces. Each element of CaPPture taken separately cannot reproduce the synergy that is obtained by the cooperation of each individual sub-area.

2.1. Collection

Data collection "is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes" [1]. The data collection component of research is common to all fields of study but is paramount in any system that attempts to make predictions based on GPS traces.

2.1.1. Related Work

GPS receivers are used in a barrage of contemporary research projects. These projects range from enhancing travel surveys [9, 10, 11, 26, 36, 107], to identifying a subject's mode of travel [87, 91, 100, 112, 113, 114, 115], along with a variety of other research activities [22, 92]. Even with the popularity of GPS as a tool for spatial data collection, there is a small minority that find GPS to be inadequate. In fact LaMarca et al. claim that GPS "has poor user-time coverage and long gaps because satellites are cut off indoors or under cover where many people spend the vast majority of their time" [60]. Their study showed that the average time associated with GPS coverage was a mere 4.5%, whereas GSM and 802.11 coverage were 99.6% and 94.5%, respectively. What is the motivation behind using GPS traces if they only account for approximately 4.5% of user-time coverage? It is because 4.5% is an extremely low estimate in an extremely urban area; for example Seattle. In actuality, GPS receivers have no problem receiving signal as long as the receiver has at least partial view of the sky. The GPS satellite constellation consists of 24 satellites (plus some spares) orbiting in six different planes. Each of these planes are inclined 55° from the Earth's equatorial plane. The satellites are positioned in their respective planes in such a manner that from almost any place on Earth, at least four satellites are above the horizon at any time [8]. With this configuration, every receiver is nearly always guaranteed to be in view of the minimum number of four satellites needed to get an accurate fix[1] (assuming there are no obstructions). If enough satellites are in view, an accuracy within two meters can be achieved (5-10 meters is a realistic expectation [106]). This spatial accuracy coupled with GPS satellites' extremely accurate clocks allows for great representation of a user's mobility.

The focus of CaPPture is to identify travel and destinations. Can we identify a user's route? Can we assign a purpose on a location where a user stops? Can we predict where the user might travel to next? These are all questions that can only be answered by starting off with the collection of GPS traces. What a user does when a GPS receiver can no longer track satellites in the sky (e.g. entering a building) is beyond the scope of this research. Table 2.1 illustrates that GPS technology has the necessary qualities that will meet the needs of the research for the proposed system.

Technology	**Precise Location**	**Low Infrastructure Demands**	**Low Power Usage**	**Scalable Coverage**	**Works Indoors**
GPS	✓	✓		✓	
GSM		✓	✓	✓	✓
802.11 WiFi					✓
Bluetooth			✓		✓

Table 2.1. Comparison of existing location technologies[110]

GSM (802.11 to a smaller degree) is a strong contender for spatial data collection as shown by Bahl et al. and LaMarca et al. in [12, 60]. There are many drawbacks to using GSM only

[1]Fix is defined as: The determining of the position of a ship, plane, etc., by mathematical, electronic, or other means.

data. First, the actual location of the GSM base station may be unknown, and GSM cell data does not provide precise location information. Cells can span kilometers in diameter, thus being in a particular cell may not provide enough resolution in terms of location. In addition, base station coverage overlaps, so several cells whose base locations are significantly far apart may be observed from one location. Finally, cell signal strength is unreliable reflections, people, and other obstacles may attenuate cell signal in non-linear ways [110].

The GPS data itself may have drawbacks. A GPS receiver's accuracy can be affected by a multitude of irregularities. A list of some minor inaccuracies in positioning are summarized in table 2.2. The effect of some of the smaller errors (tropospheric effects, calculation/rounding errors) is surmountable, but some of the larger errors do have an effect on accuracy, especially when multiple errors are coupled together.

Cause of Error	**Amount**
Ionospheric effects	±5 meters
Shifts in the satellite orbits	±2.5 meters
Clock errors of the satellites' clocks	±2 meters
Multipath effect	±1 meters
Tropospheric effects	±0.5 meters
Calculation- and rounding errors	±1 meters

Table 2.2. Summary of possible GPS errors [35]

.

There is a bigger problem called the warm start/cold start problem [26, 91, 96]. A cold start happens when a GPS receiver has been turned on and is trying to get a fix with no current satellite or location information. The time to get a location fix associated with a "cold start" ranges from approximately 30 seconds to 2 minutes depending on weather conditions, view of sky, and hardware used. A warm start is when a GPS receiver is powered on (or restarted) and has some satellite and location information with which to work. In this case, a location fix can usually be obtained in less than 10 seconds. To account for systematic and random GPS errors during the collection phase, the data must be processed.

Processing GPS data encompasses the following: filtering, smoothing, interpolation, and map-matching. These are all discussed in section 2.2.

2.1.2. Contribution

The contribution of the this system in the area of collection is a design of an intelligent system for locating, labeling, and logging (ISL^3) [45]. This is a software "tool" that maximizes the initial quality of data being collected. It takes many of the active efforts of data collection out of the users hands, and effectively replaces these efforts with intelligent choices based on past behavior. This collection tool is discussed further in chapter 4.

2.2. Processing

Processing is defined as "repairing or putting through a prescribed procedure". Each feature individually performs an essential task that ultimately determines unfavorable attributes and either identifies them or removes them. One prime reason for the processing of a collected set of GPS streams is to replace the impossible task of visually inspecting the collection [48].

2.2.1. Related Work

The prescribed procedure for the processing of GPS data is made up of a series of essential features including: 1) filtering, 2) smoothing, 3) interpolation, and 4) map-matching. In general, each processing procedure is important as mentioned in several studies [3, 6, 14, 21, 31, 54, 110]. These processing features will be described individually in the sections that follow.

Filtering. In signal processing, a filter[2] is a device or process that removes from a signal some unwanted component or feature, where signal is defined as: "any time-varying or spatial-varying quantity." In processing a GPS signal, a filter can be used for different applications at various times in the processing step. When analyzing a raw GPS stream,

[2]Smoothing and filtering are two approaches to the same problem, but from different areas. Filtering comes from signal processing, while smoothing comes from statistics. In the context of this study some assumptions with the prevalent semantics of each are taken. A Filter is considered a process in which portions of the data are "eliminated" or "rejected", whereas "Smoothing" is a process that "averages" or "corrects" data values.

each point has various attributes associated with it such as: speed, bearing, time, satellites in view, etc. In the simplest sense, filtering GPS data can use one or more of these attributes to filter a portion of the stream. Axhausen and Schussler filtered based on altitude level, speed and acceleration [91], and Upadhyay et al. used point attributes to identify and filter erroneous points [100]. Similarly, speed was used to identify and filter when a user was not moving [6, 43, 97], and signal strength attributes were used to identify and filter out points with a weak satellite fix [43].

Filtering GPS streams is not limited to the removal of data by identifying one or more point attributes that do not fall within (or outside of) some previously determined set of parameters (e.g. speed < 1 mph). The filtering step can be applied later when individual attributes are less of a concern, but aggregate features are important. An aggregate feature of GPS data might be a common route that a user travels, or a location that a user visits often. Aggregate features are made up of groups of points. A route is a grouping of spatially and temporally adjacent points where there is a distance (greater than some threshold) between the first and last point. A location is similarly made up of a group of spatially and temporally adjacent points where there is a lack of distance between the first and last point (meaning the user was stationary).

Using aggregate features, Kang et al. filtered a location if the user was not stationary long enough [50]. Krumm and Froehlich both filtered routes that met the following criteria: 1) trip time too short, 2) trip length too short, and 3) contained too few GPS points [33, 54]. In fact, Krumm concludes that having the ability to predict a driver's destination could be used to filter out entire portions of collected GPS data, allowing algorithms to concentrate solely on data concerning locations or situations associated with a particular drivers' behavior [53].

When raw GPS streams are recorded, there is always some degree of error in some or all of the GPS points. A point that deviates in value (spatially or temporally) from its neighboring points in such a manner that it is "numerically distant," it is labeled as an outlier (also referred to as noise). Noise deviates considerably from its neighboring data and must be managed. Filtering is one technique to identify and manage outlying GPS points. Greenfeld mapped points to an arc to identify and filter outlying GPS points [37]. Another approach to handling noise within the data is to use a technique called "smoothing".

Smoothing. In the context of the system, filtering and smoothing are quite different. To smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while eliminating noise. As previously discussed, a GPS receiver is not guaranteed to receive error free data, and is almost always sure to have some margin of error. Some points will be easily identified and removed by the filtering step, but this is not always the case. In many instances, points fall just inside the threshold parameters and are not filtered. These points, however, could possibly be statistically distant enough from neighboring points to alter any conclusions from analyzing that portion of the stream. This can be rectified by the implementation of a smoothing algorithm. Kanagal and Deshpande state that in some instances "Smoothing does not significantly improve upon the results (reduce variance) of the filtering step and in those cases filtering alone guarantees sufficient modeling accuracy with good performance" [49]. This is not always the case though, and any points that are missed by filtering, but are still statistically distant, must be dealt with by smoothing.

Jun et al. studied three smoothing methods comparing the data recorded by a vehicle speed sensor (VSS) and GPS unit and concluded that a method called the modified discrete Kalman filter had the best results in matching the smoothed GPS data to the actual VSS data [48]. Similarly, Quddus et al. used a Kalman filter to smooth GPS vehicle positions [84]. Reddy et al. had problems inferring transportation modes for motorized transport, biking, and running and concluded that incorporating smoothing algorithms would help increase the performance of their prediction model [87]. Ogle et al. used a smoothing technique called "T4253H-smoothing"[3] to match GPS calculated speeds to speeds collected by Distance Measuring Insruments (DMI) and found the reduction in variance of the GPS speeds to be acceptable and appropriate for their project [78].

[3]T4253H-smoothing is a standard method for smoothing a time series in the data handling software SPSS.

Interpolation. GPS receivers are appropriate tools to supplement the collection feature of the CaPPture system. Smoothing deals well with data that is noisy, but data that has unfavorable attributes must be filtered, which may leave gaps. Additionally, no receiver is invulnerable to signal loss, and therefore, data interpolation can help replace the missing data.

One focus of GPS interpolation is during signal acquisition and obtaining/maintaining a location "fix". Recall that a GPS receiver is in constant communication with an array of orbiting satellites. Each satellite has an extremely accurate clock (much more so than the receiver itself). The signal that is received from each satellite contains the unique identifier, precise timing data, and the ephemeris[4] data about the remaining satellites.

Each satellite transmits its unique identification code (with other location data) in 1-millisecond intervals. The receiver compares its internally generated code against the repeating unique identifier from each satellite and looks for any lag from the expected 1-millisecond interval. Any deviation is assumed to be the travel time of the GPS signal from space. Once the travel time (ΔT) is determined, the receiver then calculates the distance from itself to each satellite using the following formula: Distance = (ΔT) x Speed of Light.

The inaccuracies of the GPS receivers' clock, coupled with its reception of signals from multiple sources at any one time, gives a margin of error. Figure 2.1 shows an example of the possible margins of error given an inaccurate clock. A receiver in view of additional satellites will benefit with better accuracy, but the additional satellites cannot overcome clock error. This is when the GPS receiver will start to interpolate its position by analyzing the overlap error of individual clocks. Not only does this interpolation give the receiver its correct location, but it also allows the receiver to correct its own internal clock error.

[4]An ephemeris is a table of values that gives the positions of astronomical objects in the sky at a given time or times.

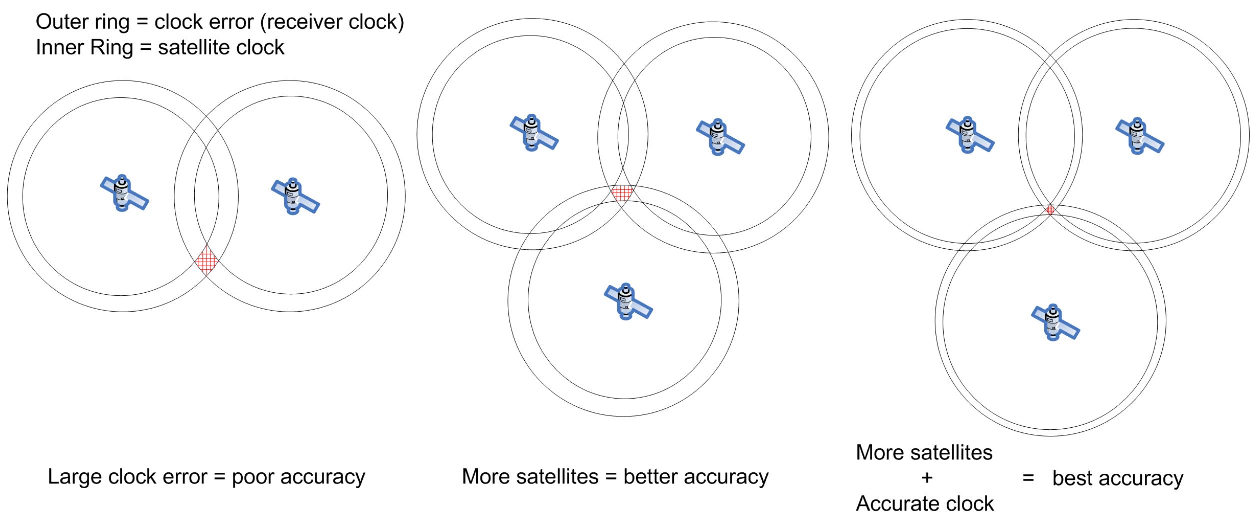

Outer ring = clock error (receiver clock)
Inner Ring = satellite clock
Large clock error = poor accuracy
More satellites = better accuracy
More satellites + Accurate clock = best accuracy

Figure 2.1. Satellite footprint overlap - showing clock error.

Most GPS receivers will interpolate position for a short period of time when the signal is totally lost or has deteriorated. This happens when there is a low number of satellites in view, i.e. the HDOP[5] is high. When a receiver keeps picking up and losing satellites in view, it may depend a little too much on its built in clock. Typically the satellites in view of the receiver will help make adjustments to the local time so that its positioning can be accurate. If the timing is off even by a small fraction, accuracy suffers (see table 2.2). When position cannot be accurately calculated, the receiver may provide latitude, longitude, altitude, heading, and speed, all by interpolation from the immediately preceding valid records. In most cases, this occurs for only small periods of time, but does have a negative impact, creating records that may be unreliable [96].

The proposed system does not deal with the interpolation of signal by the receiver itself, instead it concentrates on filling in the gaps from filtered data and loss of signal. Gaussiran et al. created a software suite that supports GPS research [34]. A portion of their library of mathematical algorithms includes Lagrange interpolation and Runge-Kutta interpolation [35]. Schuessler and Axhausen express the need for an interpolation solution to the "cold start/warm start" problem [91], and Stopher et al. had mild success with

[5]HDOP (Horizontal Dilution Of Precision) is a measure of how well the positions of the satellites are arranged. Higher HDOP values are cause by the satellites being at high elevations.

interpolating values lost during a "cold start/warm start" when using cars as the mode of transportation. Their motivation was to create a better estimate of when travel began and to correct the trip length in both distance and time [96].

Wiehe et al. interpolated position in the event of signal loss by imputing missing data values between two logged points. Their method was to insert values at 5-minute intervals (depending on the time gap). "If 2 temporally adjacent points bounding a period of missing data were within 30-meters (the distance used to determine the 2 points were in the same or similar location), we assigned the missing GPS data point to the earlier point. For data points more than 30 meters apart, we imputed in 2 ways. In one case, we used the point which was closer to home and, in another, used the point which was further from home [105]." They used varying time lapse cut-offs for imputation and found similar results with all methods.

After a GPS stream is attuned as much as possible using the above methods, there is still one more step that could allow the GPS stream to become even more reliable. The method is known as map-matching. This is the process of taking one or more points from a GPS stream and "matching" it to an underlying road network. If one is walking or biking they will not be bounded by an underlying road network, and map-matching would not be useful. Even though there is some research dealing with mobile pedestrians [72], in the context of the proposed system, this work will concentrate on GPS streams that are collected while in an automobile, and the only reason for determining the mode of transportation would be to eliminate a specific portion of the GPS data from the research. In addition, map-matching can also be broadly categorized into two distinct types: "online" and "offline". Online map-matching deals with matching points to an underlying road network in real time. A prevalent example is the common use of GPS devices as vehicle navigation systems. Offline map-matching deals with the processing of GPS data "a posteriori," and is focused on the analysis of the data set. The scope of this proposal focuses on the analysis of GPS data streams and will exclude the investigation of online map-matching. "Map-matching" (unless explicitly labeled) for the remainder of this thesis will be referring to "offline" map-matching.

Map-matching. When raw GPS streams are recorded, there is always some degree of error in some or all of the GPS points. This is the general motivation for the filtering techniques mentioned above, which is to remove points or features that do not meet a set of predefined parameters. Filtering, however, is not just limited to the removal of points or features, it can also be used to aid in map-matching GPS points. This is a process in which points in a GPS stream are "adjusted" so they are "matched" to an underlying GIS definition of a road network (this is defined more vigorously in section 3). This was done by Zhou et al. [120] and Carstensen [18]. An example of a raw GPS stream juxtaposed with a map matched version of the same GPS stream can be seen in figure 2.2.

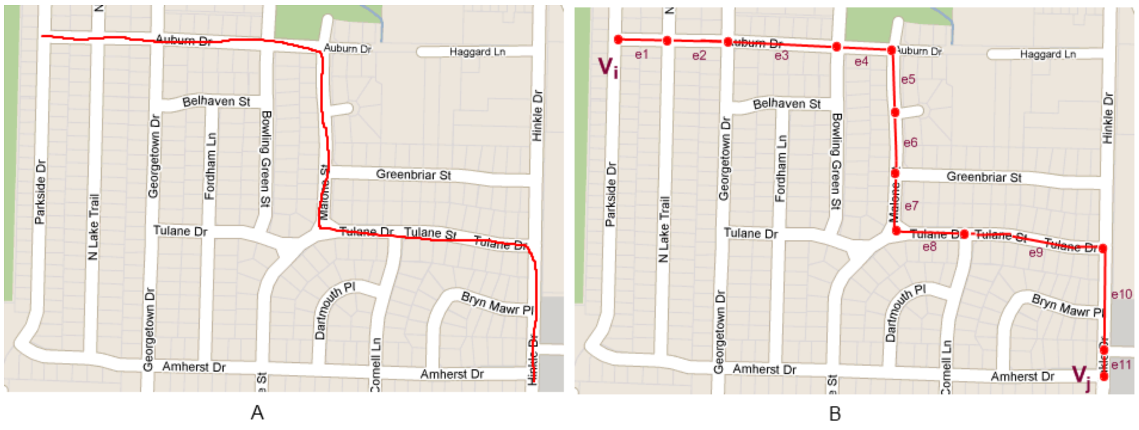


Figure 2.2. Part A shows a raw GPS trajectory T with its Map Matched counterpart in B.

In conventional map-matching algorithms the ideas are closely related. They all extract candidate road segments, then evaluate each candidate based on heading, connectivity, closeness, tracking length, etc., and finally they choose the best candidate [61]. For a conventional map-matching algorithm to be helpful, it must perform two tasks: 1) the correct road link must be unambiguously identified and 2) all the identified links must together form a meaningful travel route [120].

Although map-matching techniques usually have the same goal, this does not mean they may use different approaches. Procedures for map-matching vary from those using very simple search techniques [52] to more advanced approaches. These more advance approaches for map-matching algorithms can be categorized into four groups: geometric [13], topological [20, 37, 47, 85, 104] , probabilistic [77, 111], and other advanced techniques like the Kalman Filter, Extended Kalman Filter, fuzzy logic, and Belief Theory [27, 46, 82, 83, 98, 108, 117]. The proposed methodology section discusses a novel approach to map-matching, where an "off the shelf" approach gives a simple solution to a complex problem. It describes an efficient algorithm that uses web service driving directions to ensure logged GPS points are adjusted to an underlying map structure. This important step in GPS processing will allow the normalization of routes and stops which in turn will increase performance for any prediction models.

Marchal et al. [70] developed an offline map-matching algorithm that uses only the coordinate data that is collected by the GPS. It does not use a DR (dead reckoning) device, nor is heading or speed data considered. The road network used by the algorithm is represented by a directed graph $G(V, E)$ where V is the set of vertices or nodes and E is the set of edges or links. Q_i is the set of coordinate points (x_i, y_i) from the GPS data where $i = 1 \dots T$. The algorithm starts by finding the set of N nearest links from the first GPS point which is Q_1. For each GPS point in Q_i, the N nearest links are added to the previous link thus forming multiple paths. Finally, the difference (error) between the points in Q_i and the paths is calculated and scored. The path with the lowest score is chosen to be the best estimation of the actual route that was traveled.

Yin and Wolfson [109] developed a weight based offline map-matching algorithm. In addition to the coordinate data collected by the GPS, the algorithm uses heading data. The distance between the GPS coordinate data and nearby road network links (edges in a graph) is calculated as well as the heading difference between the GPS coordinate data and nearby road network links. These calculated values are used to assign weights to define how well a coordinate point matches a specific link. The route (series of links) with the lowest total weight is considered to be the map matched route.

Alt et al. [4] implemented a map-matching algorithm which uses Frechet distance to match a route formed by GPS data points to a sequence of edges in a road network graph

$G = (V, E)$. The Frechet distance is commonly described as follows: Suppose a person is walking a dog, the person is walking on one curve (route) and the dog on another. Both are allowed to control their speed but they are not allowed to go backwards. Then the Frechet distance of the curves (routes) is the minimal length of a leash that is necessary for both to walk the curves from beginning to end.

Ramer, Douglas, and Peucker [25] developed the Ramer–Douglas–Peucker (RDP) algorithm for reducing the number of coordinate points required to represent a curve. The algorithm relies solely on a distance metric, ϵ, which determines whether or not one of the original points in the curve will be included in the reduced point representation. The value ϵ represents the perpendicular distance (see figure 2.3) from a line segment in the route, to another point in the route being considered for preservation.

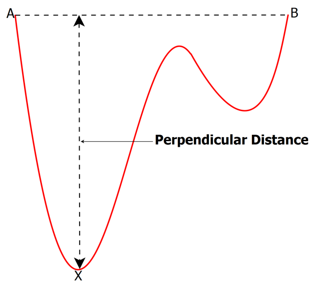


Figure 2.3. X is the point with the greatest perpendicular distance from the line created by A,B.

If the perpendicular distance is less than ϵ, then the point is preserved and included in the result. In other words ϵ determines the “granularity” of the resulting representation. By changing its value, qualities of the final reduced point route can be adjusted to a be a “fine grained” or a “course grained” representation. The smaller the value of ϵ, the closer the reduced point route will be to the original route and in turn, more points from the original route will be included. Figure 2.4 shows that the progression from A$\rightarrow$F adds more points to the resulting representation as the the value of ϵ changes. Initially in 2.4(A), only the first and last points are in the Peucker result set (this is the minimum result), and by 2.4(F), nearly all the points in the original curve are in the final representation.

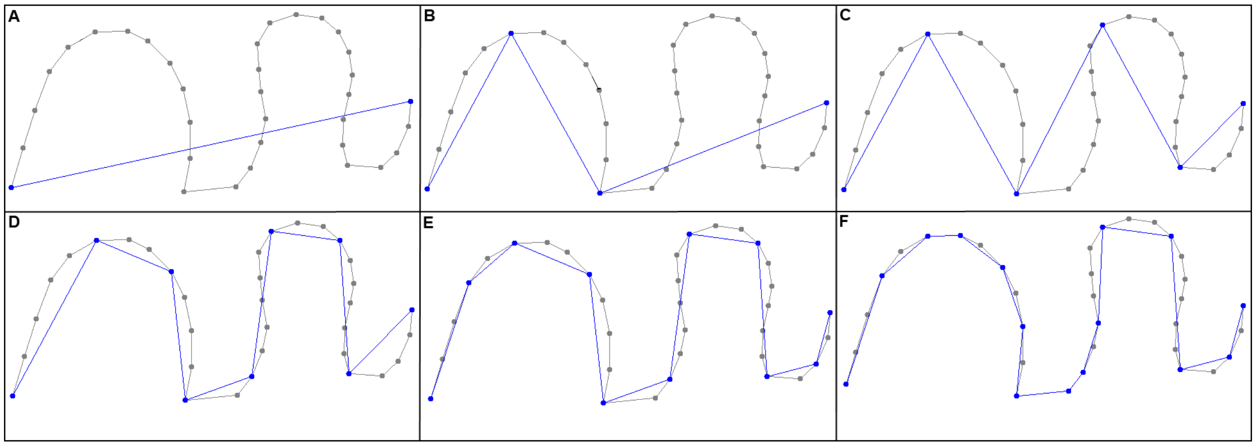


Figure 2.4. RDP algorithm showing the progression from A $\rightarrow$ F of the resulting granularity of a route by changing the value of ϵ.

Just by adjusting the value of ϵ, one can change the resulting representation of the curve, but one cannot determine with any consistency how many points, or which points, will be included in the reduced representation. It is important that the improved point selection algorithm achieve a reduction comparable with the original RDP algorithm (described in algorithm 1), but also use an intelligent selection criteria when assigning points to the resulting representation. A greedy algorithm using a simple threshold is altogether not sufficient.

Algorithm 1: Original Ramer-Douglas-Peucker algorithm

```
   Input: $P = \{p_0, \ldots, p_n\}, epsilon$
   Output: $P' = \{p'_0 \ldots, p'_n\}$ where $P' \subseteq P$
1  dmax = 0
2  index = 0
3  for $i \leftarrow 2$ to $(length(P) - 1)$ do
4      $d \leftarrow$ PerpendicularDistance$(P_i, Line(P_1, P_n))$
5      if $d > dmax$ then
6          $index \leftarrow i$
7          $dmax \leftarrow d$
8  if $dmax \geq epsilon$ then
9      $P'_1$ = RamerDouglasPeucker$(P[p_1, \ldots, index], epsilon)$
10     $P'_2$ = RamerDouglasPeucker$(P[index, \ldots, p_n], epsilon)$
11     $P' \leftarrow P'_1 \cup P'_2$
12 else
13     $P' = P_1, \ldots, P_n$
14 return $P'$
```

2.2.2. Contribution

The contribution to the processing portion of the proposed CaPPture system is to create a novel map-matching technique, that is, a routing-based map-matching approach for standardizing identified routes in a collected set of GPS trajectories. An improved version of the RDP algorithm plays an important role in this novel map-matching technique. Chapter 5 describes this approach and results of the implementation in further detail.

2.3. Labeling and Prediction

Predicting the location of a mobile user is an interesting and challenging problem. Ubiquitous electronics, such as cell phones and GPS devices, have the potential to act as resourceful augmenters for everyday activities and assist the user by making contextual suggestions. These contextual suggestions have been coined with a few terms, one of the more common terms being "location based services" (LBS). LBS is defined as: "services that integrate a mobile device's location or position with other information so as to provide added value to a user [90]." LBS is not strictly defined to take into account an individual's purpose for being at a specific location, but would be greatly augmented by this ability.

Before truly useful suggestions can be made based on context, the context must be labeled or classified by some predictive model. Together, a user's current location and purpose for being at that location form one of the most useful contextual attributes used to make appropriate suggestions. The main goal of the system is to label a user's current location, and in addition to this, predict that user's next location and label it with a purpose.

A major step in the creation of a prediction model is to analyze the previously collected data by using some statistical procedure. One known procedure that can discover groups of data that are similar is clustering.

2.3.1. Clustering

Clustering is an important tool, and one of the first steps in unsupervised learning and prediction. It deals with finding structure during the collection and processing of unlabeled data. Clustering is the process of organizing objects into groups whose members are similar in some way. A cluster is therefore a collection of objects (points) which have similarity between them and are dissimilar to the objects belonging to other clusters. Researchers have applied a variety of clustering algorithms when analyzing location data and trying to discover spatio-temporal patterns [28, 30, 58, 59, 79, 89, 102].

Clustering requires techniques to statistically classify the data (points). “Statistical classification is a supervised machine learning procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items (referred to as traits, variables, characters, etc.) and based on a training set of previously labeled items” [88]. If the system is going to do any type of classification, it must first be put through a step where one or more of these existing characteristics can be identified within the data.

K-Means clustering is a partition-based method of clustering, which aims to partition a data set of size n into k clusters (partitions) [67]. The value for k is decided a priori, and the centroid location for each cluster (partition) is also decided at the beginning of the algorithm. Placing each centroid, in such a manner so that they are as far away as possible from each other, is a good starting point. The algorithm is described as below:

(1) Place initial group of K centroids into the space.
(2) Assign each object to the group that has the closest centroid.
(3) When all objects have been assigned, recalculate the positions of the K centroids.
(4) Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Ashbrook and Starner used a variation of the well-known K-Means clustering algorithm to learn a user‘s significant locations from location history data [6, 7].

K-Means clustering in general has several drawbacks. First, the number of clusters must be specified before clustering begins. This is a difficult task, especially between users who might have very different behaviors (e.g. very different frequented location behavior). Second, all points are included in the final clustering results. This allows K-means to be quite sensitive to noise. “A single noisy or uninteresting location reading far from other points can pull a cluster center toward it much more than it should, because the squared-distance error term heavily weights distant outliers [119].” Finally, the K-Means algorithm is nondeterministic: the final clustering depends on the initial placement of the centroids.

Spectral clustering is also a partition-based clustering technique that is based on spectral graph theory. The idea is to define an objective function for partitioning a graph into sub-partitions and to minimize an objective function that defines the “goodness” of a cut. Ng et al. cover several methods for measuring the “goodness” of a cut [75]. Nurmi and Koolwaaij implemented a version found in [101]. They also used duration and cell count information as a post-processing step to prune out any irrelevant clusters. The algorithm can take advantage of small scale locality information to produce spatially tight clusters using only coordinate data. The authors claim that one drawback of this algorithm is its expensive memory requirements. This is only applicable to mobile phones where the architecture obviously has restricted resources [76].

Kang et al. developed a time-based clustering algorithm to extract places using Place Lab [60] to collect traces of locations. A new cluster is discovered when the user goes beyond a certain distance from the previous cluster and stays in place more than some time threshold. The algorithm does not consider revisiting previous locations. More simply, each time it discovers a place, it is a different place. This also makes it difficult to discover places that are visited with high frequency, but short dwell time. Finally, this method requires continuous location data collection with very fine intervals, and thus large storage. The authors claim it can be run on a mobile device as a background process, but with continuous logging and distance calculations being done, this technique is likely to be power intensive [50].

Duration based grid clustering detects peaks in the duration as a function of latitude and longitude, which are discretized into a grid. A cluster is then formed by connecting neighboring grid points that have a relative duration larger than a threshold percentage. Heuristic graph clustering uses GSM cell transitions to induce a directed weighted graph, which can then be used to identify places. Frequent transitions is a clustering technique that takes advantage of the underlying GSM infrastructure and clusters frequent transitions between cells. This assumes that, while stationary, a user will be handed off between overlapping cells frequently, thereby identifying a stay [76].

DBScan (density-based spatial clustering of applications with noise) is a popular density-based clustering algorithm upon which much research has been based [15, 30, 89, 102]. It is a density-based algorithm, because it uses two parameters to control how close each point needs to be within the cluster. Epsilon, or ε, measures the density of the cluster and minPts measures the minimum number of points within the cluster needed to qualify for consideration. It starts with an arbitrary starting point that has not been visited, and calculates its ε-neighborhood. If it contains a count larger than minPts, a cluster is started. Otherwise, the point is labeled as noise. This does not mean that this point is ignored for the rest of the process. It can later be included in another cluster, if it is within that clusters' ε-neighborhood.

If a point is found to be part of a cluster, its ε-neighborhood is also considered to be a part of that cluster. This means that every ε-neighborhood for every point in a cluster is added together to create a collective ε-neighborhood. This portion of the algorithm is what allows DBScan to "grow" and find arbitrarily shaped clusters. This process of finding points within the ε-neighborhood continues until no more points are within reach. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise. Both Griffin and Laasonen et al. use density-based clustering similar to DBScan to recognize personally important locations [59, 38].

Zhou uses density-based clustering (DBScan) [30] to reduce the systematic noise in the data and replaces logged points when a carrier is still within the centroid of the cluster, allowing for better recognition of location where a user is stopped [120]. Another example of density-based clustering is shown by Ashbrook and Starner, who use a cut-off parameter $t(\gamma)$ to determine whether the user stays long enough within a specified radius. If the duration of a stay exceeds the value of the cut-off parameter, the location is determined to be a place of interest (or a stop) [7]. A variation of this work is presented by Toyama et al. [99] who use multiple values for the radius parameter.

A similar approach has also been adopted by Zhou et al. [118], who use a modified DBScan algorithm together with temporal preprocessing for inferring places. The temporal preprocessing ensures that the places are really visited frequently enough and the modification to the DBScan algorithm is needed to cope with signal errors. Zhou et al., this time, uses a Density-and-Join-based, called DJ-Cluster, that requires at most a single scan of the data [119].

OPTICS (Ordering Points To Identify the Clustering Structure) is an algorithm for finding density-based clusters in spatial data. OPTICS is similar to DBScan, but with an improved ability to find clusters in data of varying density. It does this by ordering all the points in a linear fashion, so that points that are spatially close together become neighbors in the ordering [5].

Toyama and Hariharan use a type of agglomerative (hierarchical) clustering because of the ability to specify the spatial scale of the clusters, rather than the number of clusters or the number of points contributing to a cluster, neither of which are known a priori [41].

There are many different clustering algorithms in many different categories, including density-based, partition-based, and hierarchical, just to mention a few. This section concentrated on the more dominant clustering algorithms as they pertain to the problem of finding significant locations. K-means and DBScan are the most studied clustering algorithms (with many variations), with the majority of current research leaning towards variations of density based algorithms and some agglomerative (hierarchical) methods. An interesting point is the number of clustering algorithms using a GSM infrastructure to identify significant locations by using some type of "transition" between cells to either cluster frequency or build a transition graph [76, 95, 110]. One obvious point is that clustering has a wide range of uses and is not limited to the problem of finding significant locations, as will be seen in subsequent sections.

2.3.2. Route Prediction

Unlike a trajectory, which is simply a set of spatial and temporal contiguous points, a route can be defined as a GPS trajectory taken between two known destinations. There can also be multiple routes between the same two destinations, and route A→B is not necessarily the reverse of route B→A. Drivers tend to be very consistent when choosing their routes between origins and destinations [33] making it possible to create route prediction algorithms.

Predicting a drivers route is useful for many reasons, such as warning a driver about upcoming problems regarding traffic, alerting the driver about upcoming points of interest, or suggesting more efficient routes. "One of the most innovative applications of end-to-end route prediction is for improving the efficiency of hybrid vehicles. Given knowledge of future changes in elevation and speed, a hybrid control system can optimize the vehicles charge/discharge schedule" [33]. Also, one can never discount the importance of advertising. Knowing a drivers route could be an effective basis for targeted advertisements.

Laasonen et al. present a novel adaptive framework for recognizing personally important locations. They divide activities into locations, bases (clusters of locations), routes, and areas. To determine areas, a simple form of density clustering is used. In their route prediction model, only cell (GSM) transitions were addressed. They evaluated the route prediction accuracy by calculating the next base after each cell transition (unless the user actually arrives at a base) and then compared their prediction results to the actual base that was reached next. They found their method to be just above 50% accurate [59]. Additionally, Laasonen devised another algorithm that analyzes entire routes by using an improved clustering technique in which distinct physical routes correspond to clusters of cell sequences [58].

Patterson et al. presented a method for modeling a traveler through an urban environment. The model simultaneously determines the users' mode of transportation and also the route most likely taken. A graph was used to represent the state space where given a graph $G = (V, E)$, the set of edges E represented straight road sections and foot paths, and V represented either an intersection, or accurately modeled, a curved road as a set of short straight edges. To estimate the users' mode of travel, Bayes filters were used. The authors used Expectation-Maximization to estimate the values of the parameters to further strengthen the prediction model. The authors were successful in predicting mode of transportation, but were only about 50% successful when predicting user routes [80].

Persad-Maharaj et al. propose an algorithm called "Path Maker" for creating and maintaining an archive of user trip records that are converted to polylines. The "Path Maker" algorithm then stores a representative version of the polyline as a polygon that has a five meter buffer around the original polyline. Spatial calculations are more efficient with geometries that have stored and indexed spatial area. To do route matching, the polyline is selected against the spatial database of polygons looking for intersections. Just as in most route matching algorithms, problems arose when the current route did not completely match any of the stored routes. In the context of the "Path Maker" algorithm, the polyline did not fall entirely within one of the polygons in the database. The authors were marginally successful in their prediction efforts and with improvements the route matching algorithm could be useful [81].

Froehlich and Krumm use the Multiperson Location Survey [56] data and a similarity algorithm to predict a users next route. The data is first segmented into routes and location stops and then the similarity algorithm clusters the routes together using a dendrogram clustering technique to form like-routes. These routes are then stored in a similarity matrix for comparison later. The authors concluded that drivers are very regular but their algorithm had trouble effectively matching routes; however, partial route matching, and possibly the inclusion of other features such as frequency and recency of travel, would improve the results [33].

2.3.3. Destination Prediction

Destination prediction methodologies infer a users' trip destination through a variety of methods. The use of wireless communication devices have become pervasive enough to allow the effective creation and verification of these prediction models. The need for destination prediction is not obvious and borders on the infringing privacy; however, judging by the types and number of devices people are willing to carry around, it is clear that there is a desire to utilize the predictive albeit invasive nature of these small computing devices. Applications that enhance a users' daily routine with supplemental information appear preferable to the possible loss of privacy. If an application can predict a users destination, then the application can relay relevant information such as nearby points or available services, to a user before they reach that location.

Liao states that "when predicting a persons destinations from raw GPS measurements, the system may have to infer street, mode of transportation, and travel route in order to make reliable predictions of destinations [62]." These are just a few of the attributes that can be used to build a destination prediction model. Ashbrook and Starner used Markov models trained to predict a next or most likely destination candidate based on recently visited destinations by using clusters of GPS data as inputs [7]. Karbassi et al. developed an algorithm that makes use of probability histograms taken over a period of three years for creating discrete daily time intervals. A bus passenger can then be matched to one of five fixed bus routes using these time intervals. Depending on the time of day, their algorithm anticipates that a bus will travel on a certain route, and in a certain direction [51].

Simmons et al. use a Hidden Markov Model (HMM) to capture a sequence of driver actions used to navigate a route and deduce the destination based on a present and next state transition model [93]. Marmasse and Schmandt also used a HMM to predict users destinations in their "comMotion" research. They found that when comparing a Bayes classifier to histogram modeling, the histogram modeling performed better. They attribute this to the small amount of training data where Histogram Modeling tends to outperform its counterparts [71].

Krumm and Horvitz present a methodology known as *predestination*, that not only predicts a users destination, but also takes into consideration previously unvisited locations. This is not taken into consideration in the previously mentioned prediction models. The method used by Krumm and Horvitz divides "space" into a 40x40 two-dimensional grid of cells measuring 1km wide and 1km high. The goal is to predict the cell in which the driver will conclude their trip. This is also how they handle previously unvisited locations. Each cell has a probability associated with being the destination $P(D = i|X = x)$, where D is the random variable representing the destination, and X is a random variable representing the vector of observed features from the current trip the user is on. The features can range from a variety of things including: “time of day”, “day of week”, and “ground cover”. With three different variations of their model, the median prediction error was about 2km at the halfway point of the trip [57].

Project Lachesis proposed a number of rigorously defined data structures and algorithms to assist in the analysis and identification of location histories (trip stops). Hariharan and Toyama focused on the creation of a location clustering algorithm that was used to model transitions between clustered locations using a Markov Model. The primary contribution of this research, was, however in its rigourously defined data structures for the storage and analysis of location information [41].

2.3.4. Significant Locations & Labeling

Many location-aware applications have recently started incorporating the notion of a “personally defined place.” Weilenmann and Leuchovius proposed “everyday positioning” to obtain personal places, such as “I’m waiting where we met last time” for location-based services [103]. *DeDe* is a location aware mobile messaging application, allowing messages to be delivered to places of interest that are meaningful to individual senders and recipients. The delivery of the message is activated when the recipient spatially comes into contact with the meaningful location [119]. *Place-Its* is a location-based reminder system, allowing users to create brief personal reminder messages that are essentially “delivered” to a significant location that is defined for a specified group of users [94]. The key challenge faced by these applications is to understand what constitute a users’ personally meaningful place, and establishing the relationship from the physical locations to the personally defined location established in the collection of a users’ history. Also known as personal place acquisition, this discovery process is an essential requirement for location-aware applications and remains an open research challenge [119]. Although there have been a number of interesting place discovery systems [6, 43, 63, 71] we know of no large scale quantitative empirical studies that evaluate how well these systems work [7].

Ashbrook and Starner propose a route recognition model that builds a Markov model of movement between important locations [6]. Simmons et al. use a Hidden Markov Model (HMM) to capture a sequence of driver actions used to navigate a route and deduce the destination based on a present and next state transition model. While training an HMM in general requires large amounts of data, it is feasible for this model because the structure of the model is known a priori. It is the road map of the area (limited number of segments), and in addition to this the statistical model will be relatively small because a single user utilizes a small fraction of the roads in any region [93].

2.3.5. Contribution

The prediction portion of the proposed system utilizes a novel classification model to automate trip purpose labeling by using aggregate data obtained from clusters of trips stops. This technique is discussed further in chapter 6.

CHAPTER 3

GPS CaPPture SYSTEM OVERVIEW

This chapter presents an overview of the CaPPture system architecture. Key research problems concerning a system that makes predictions based on GPS traces are identified and described. Methods to address each of the key research problems in collection, processing and prediction and the implementations and results of those methods are presented in the three succeeding chapters.

3.1. System Overview

Figure 3.1 shows the general architecture of the proposed CaPPture system. The system is divided into 3 major modules: 1) collection & processing, 2) map-matching, and 3) labeling and prediction. Each module is discussed in more detail in the subsequent chapters addressing the key research problems related to this thesis.

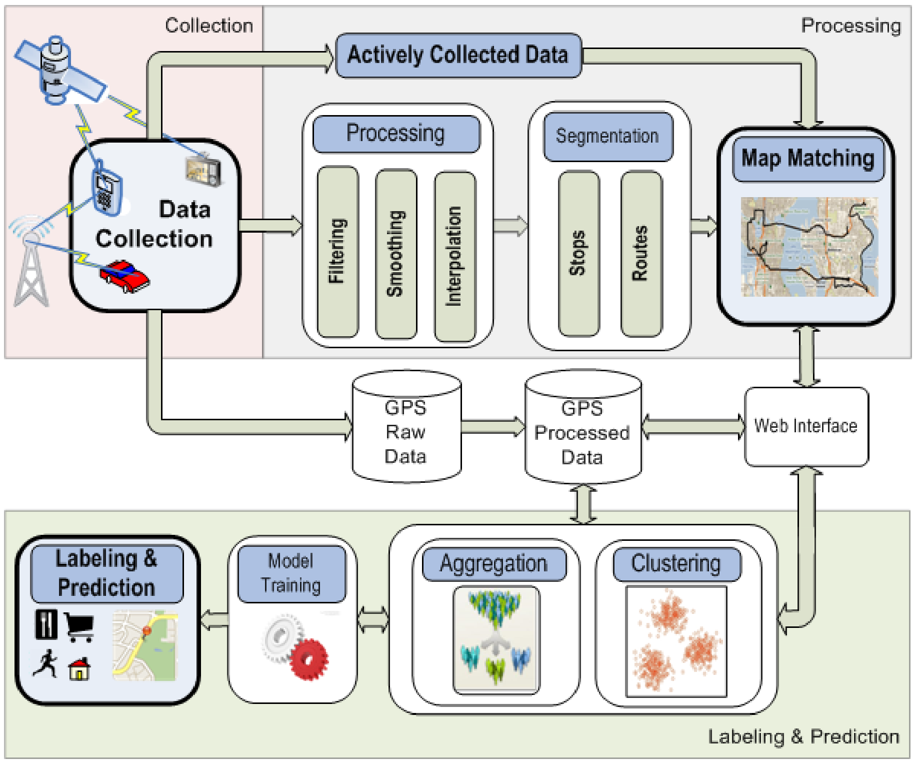


Figure 3.1. System architecture.

One of the proposed design strategies to help strengthen the integrity of the CaPPture system is to base a large portion of its design on the software architecture: model view controller (MVC) [16] in which the design of a system is divided into the following:

- Model: The model contains the bulk of the logic for an application and also represents the "state" of an application. *Models* include: data access, validation rules, and aggregation logic.
- View: The view encapsulates the interface to the user. The implementation language is irrelevant, however, in a web based system this is usually created using a combination of html and javascript. The interface creates the functionality that allows a user to interact with the system via a *controller*.
- Controller: The controller contains the control-flow logic. It interacts with the *model* and *views* to control the flow of information and execution of the application. This separation of roles allows the creation and maintenance of the system to be done faster and with flexibility. For example, by separating the *views*, the logic driving the filters and prediction models can be developed without dependance on the visual aspect of the system. This is quite important due to the fact that normal applications have logic and visualization intertwined.

Figure 3.2 is a graphic representing each role in the MVC architecture along with the relationship between each. The solid line represents a direct association, and the dashed line an indirect association. The responsibility of each role is emphasized by its literal size in the graphic. For example, the representation of the model appears larger in the graphic than the other roles. This is because it typically contains the bulk of the logic for an application. The representation of the controller appears to be "skinny" because its job is simply to link models and views together. This is not placing an importance on any of the individual roles, it is simply representational of the amount of logic in each role.

Where the MVC architecture would strengthen the CaPPture system is by creating a series of well defined classes (models) that would ensure the verification and validation

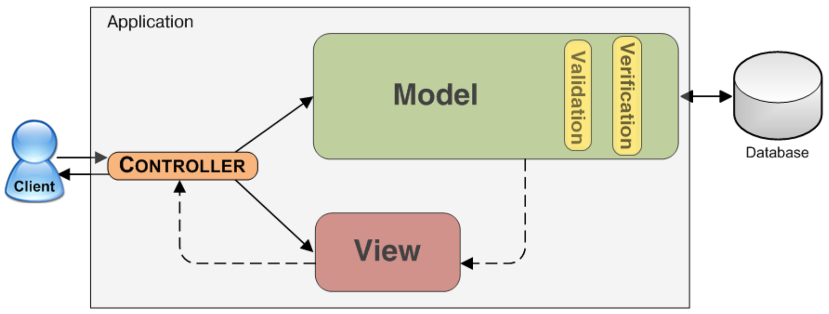


Figure 3.2. Model view controller.

of the data being committed to the database. The first module in the CaPPture system is the collection of GPS data. Regardless of the hardware or formatting of the data being collected, it is quite important that the data is stored using rigorously defined structures in the assurance that any entity basing results on the integrity of the data is not compromised.

To ensure the integrity of the data, the collection models would be partly based on the work done by Hariharan and Toyama in "Project Lachesis: parsing and modeling location histories [41]", in which the authors present a series of rigorously defined data structures and algorithms for representing (storing) and analyzing GPS traces. These data structures and algorithms are explored further in chapters 4, 5, and 6.

The collection module will receive raw GPS data collected by a user in various formats (see appendix A for recognized formats). The goal of collection is to create a pristine processed GPS database that stores raw and processed GPS data in such a manner as to allow seamless access and manipulation of the data. Further discussion of the database design and implementation will be presented in chapter 4.

Attached to the processed database is a verification module built using a web based interface. This was developed using an Ajax Programming Model on top of a system utilizing an MVC type architecture in [16]. The data is then processed, depending on the method used for collection. Raw data collection requires further attention by the processing and segmentation modules, before it is sent to the map-matching module.

If the data is collected using the proposed collection software, it can bypass the processing and segmentations modules because that functionality is built in to the software itself. In the context of this system, the map-matching module will be used as a refinement step to make the final adjustments to each GPS point ensuring they are usable and correct. Another purpose behind the map-matching module is to remove large portions of the collected data without jeopardizing the integrity of the data. This step performs the bulk of the logic behind all of the filtering. It turns a raw GPS trajectory into a reduced (by removal of unnecessary points) and adjusted (points are snapped to the road network) route. This is achieved by using a modified Douglass-Peucker algorithm [25]. Details of the improved Douglass-Peucker curve reduction algorithm will be discussed in chapter 5.

The interface provides a series of views that allow a user to "manage" routes and stops. Two significant problems in GPS stream segmentation are 1) identifying two contiguous stops that should be one (usually because of GPS error showing movement when there is none), and 2) missing the identification of a stop due to the stop length being below the given time threshold. The effect of this last error snowballs by appending two separate routes together, creating a single longer route, rather than two distinct routes. Both of these problems could be alleviated when using the proposed data collection software.

After the GPS data is processed and verified, the remaining modules perform the tasks necessary to start training prediction models. The clustering module allows a user to choose the most appropriate clustering algorithm by using previously implemented algorithms (DB-scan, K-means, Dendogram), or inserting a new implementation of an appropriate algorithm. The system will allow the clustering of one of two features: stops or routes. The clustering and aggregation modules can write back to the database storing values to be retrieved for use later. The last two modules are used for model training and finally prediction/labeling. The training module currently utilizes the c4.5 decision tree algorithm. C4.5 will be discussed in detail in chapter 6 (Labeling & Prediction).

3.2. Key Research Problems

Key research problems are identified and discussed below:

(i) *Data Collection.* The ensuing key research problems rely strongly on reliably collected GPS data. An entirely passively collected data set will not have the confidence that data has when using a small amount of interactivity from a user for which data is being collected. The proposed system easily guides the user through the data collection process which will result in a segmented and labeled set of GPS data streams.

(ii) *Design and Implement a series of rigorously defined data structures.* This is done accurately representing the collected data while concurrently identifying the key features:

- Stay - is a single instance of an object spending a minimum amount of time in one place.
- Destination - is any place where one or more objects have experienced a stay.
- Trip - A single trajectory between two adjacent stays (made by a single object).
- Path - is a representation of the description of a set of trips between destinations.

(iii) *Identify and Implement improved filtering and smoothing techniques.* GPS streams will always have errors, even ones collected using reliable high quality hardware and software. Finding an effective combination of smoothing and filtering algorithms will ensure a high quality data set. Chosen algorithms have to be developed/configured properly as to not remove or adjust GPS data points with indiscretion.

(iv) *Develop an Offline Map-matching method for data quality improvement.* The map-matching method needs to be the final step in removing any and all errors from the collected set of points. By creating a robust offline map-matching method, the data being used in the development of a labeling model will be substantially reduced and error free, thereby allowing all concentration to be focused on the labeling model without questioning the accuracy of the training data.

(v) *Design and Develop algorithm for labeling Trip Purpose:* The final goal in the system was to develop a robust labeling model. By correctly labeling a users current or next location, each of the overarching mobile trends mentioned in the introduction could have substantial support with software development. It is apparent through the lack of available apps that current location labeling algorithms are deficient in their goals leaving a research void that needs study.

3.3. Proposed Methods

The following sections describe the targeted approaches used by the proposed system in the process of finding solutions to each of the key research problems.

3.3.1. Collection and Processing

There are many proprietary and open source standards for recording and representing location data. Appendix A shows a large portion of the many possible variations that different devices and companies use. In a stark contrast, the GPS satellite constellation orbiting earth sends back data in a very standard format. It is the software on a GPS receiver that determines the type and format of the data that is stored (if it is stored at all). Along with location data, a GPS receiver can determine many additional features dealing with the strength and quality of a signal. Although these additional features can be beneficial, they are just an appurtenance to the basic location data. Keeping with this view, the proposed system uses data structures and definitions that do not depend on extraneous location data, starting with a *GPoint*:

Definition 2: GPoint

```
If a point $\rho$ is a spatial coordinate containing:
    $([lat], [lng])$
Then we define a GPoint $\rho_t$, to be a triple:
    $([\rho_t.lat], [\rho_t.lng], [\rho_t.t])$
containing both a location and a time.
```

Building on the *GPoint* type, we use a typical object oriented design pattern and derive a *Trajectory* data structure to create a container of points:

Definition 3: Trajectory

```
**Given:**
    GPoint vector:  $T = \emptyset$
    Time interval:  $I = \{0, \ldots, n\}$
    Start time:  $t$ where $t$ is a non-decreasing value,
                    such that:$t_b \geq t_a \;\forall\; b > a$, where $a, b \in I$
1 foreach ($t \in I$) do
2     $T \leftarrow T \cup p_t$        where $p_t$ = GPoint at time $t$
```

A container of points, or "Trajectory", is the basic structure behind the proposed systems approach to data collection. Simply put, a trajectory is a spatially and temporally ordered set of points that is associated with an individual. By recording multiple trajectories for multiple individuals and storing them in a manner that allows for easy retrieval and manipulation, we now have the start of a collection.

One of the major problems in previous methods of data collection was the emphasis they placed on user interaction. This proposed system will reduce the need for active user interaction by making intelligent decisions based on a users' history of travel. This moves the data collection effort more towards the passive end of the scale.

Providing an intelligent collection system requires that the algorithms guiding the collection system have knowledge of advanced data structures beyond that of a simple location point, along with the capacity to identify and pull those advanced features from the data. GPS streams that are collected 100% passively tend to have a high number of errors. These errors range from signal loss, to the cold-start / warm-start problem. GPS streams must be processed to handle these expected errors. Incorporating algorithms in the collection process, that identify and pull advanced features from GPS data streams, will require a small amount of real time processing. A description of the processing techniques used in the collection process along with an implementation of these same techniques are discussed in chapter 4.

The pre-processing of GPS data requires the extraction of many features from raw trajectory data. One of the most important features is the identification of “stays” or “destinations” [99]. A *stay* is a single instance of an individual spending time in one place. The threshold of how much time one must be immobile before a *stay* is recognized, is debatable and will be analyzed in future work. A *destination* is any place where one or more individuals have experienced a *stay*.

Accurate identification of *stays* is critical considering that the identification of *trips* is dependant on a *stays* accuracy. A *trip* is defined as a single users *trajectory* made between two *stays*. An example of a trip can be seen in figure 3.3.

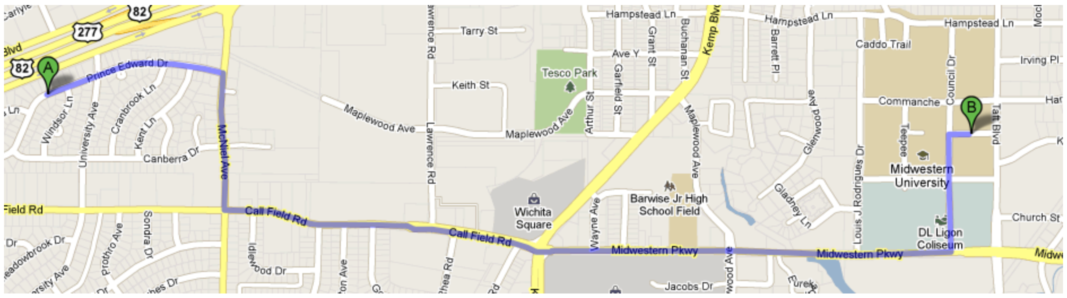


Figure 3.3. Example trip between the identified stays A and B.

3.4. Post-Processing

The emphasis of the post-processing portion of the CaPPture system is a robust map-matching algorithm used as a filtering step. To define map-matching more rigorously, this study builds on previous definitions and adds two more terms “Road Network” and “Path”. These are defined as follows:

Taking the definitions for trajectory, road network, and path, map-matching can now be defined as:

Figure 3.4 part A shows a raw path and part B shows the map-matched path with the vertices and edges of the underlying road network identified.

The map-matching approach proposed by this system first identifies key waypoints in a user’s GPS trajectory using a modified Peucker curve reduction algorithm [25]. Subsequently, it sends the key waypoints to a black-box driving directions service that returns a

Definition 4: Road Network

A road network is a directed graph: $G = (V, E)$ where:

$\boldsymbol{E}$ = the set of edges representing each road segment.

$\boldsymbol{V}$ = the set of vertices representing intersections and terminal points of road segments.

Each road segment e is a directed edge that is associated with:

$e.eid \rightarrow$ id

$e.v \rightarrow$ speed

$e.l \rightarrow$ length

$e.start \rightarrow$ starting point

$e.end \rightarrow$ ending point

Definition 5: Path

Given two vertices V_i, V_j in a road network G, a path P is a set of connected road segments that start at V_i and end at V_j:

$$P : e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_n$$

where

$$e_1.start = V_i, e_n.end = V_j, e_k.end = e_{k+1}.start, 1 \leq k < n$$

Definition 6: Map-matching

Given a road network G and a raw GPS trajectory T, find a path in G in which each $t_i \in T$ are matched with the closest $e_i \in G$.

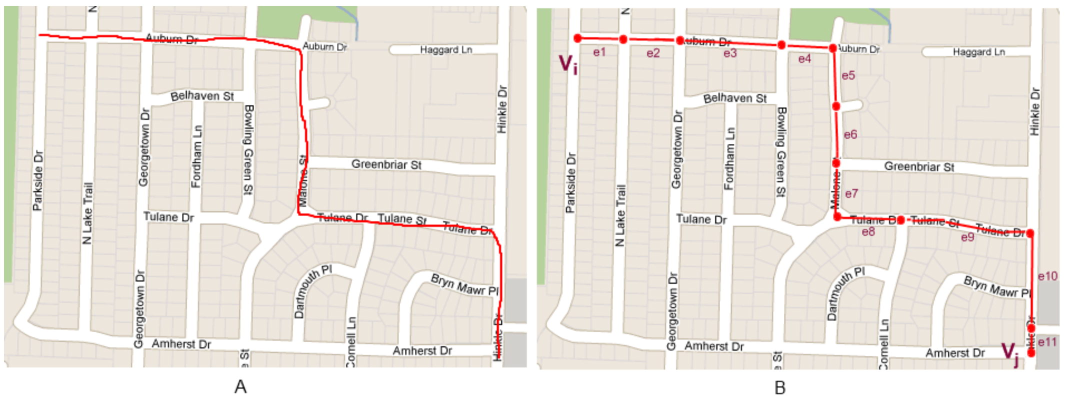


Figure 3.4. A) Raw path. B) A map matched path between V_i and V_j.

map-matched route utilizing each of the key waypoints. Essentially, it re-builds the route by interpolating and returning a map-matched minimum set of points needed to represent the route between each of the given key points. This methods implementation is described further in chapter 5.

3.4.1. Trajectory Segmentation

The goal of the collection and processing portions of the CaPPture architecture is to provide a pristine data set, by segmenting each trajectory into stops and routes, to aide in the creation of labeling and prediction algorithms.

3.4.2. Labeling & Prediction

This integral part of the processing of a users' travel data results from the identification of each stop that a user makes. The goal of the labeling and prediction portion of the CaPPture architecture is to label each stop with a purpose, or predict the next stop a user will make. A typical stop has three components. The first is simply a point, or the location in which the user stopped. The second is a timestamp, which indicates the time the user arrived at the stop. The last is length of stay, or how long the user stayed at the stop. The labeling problem is defined below:

Definition 7: Labeling

Given:

A collection of $S = \{s_1, s_2, \ldots, s_n\}$ stays where:

s_i is a pair $\{[p_t], [l]\}$

t is a timestamp

p_t is a GPoint at time t

l is a length of stay

Find:

$s_i.label$

Where:

$label \in \{home, work, shopping, \ldots, eating\}$

By clustering spatially similar stops together, aggregate data for each cluster can be calculated by averaging or identifying temporal values. The proposed system would use these aggregate values to train prediction algorithms. Some of the aggregate values that would be identified are (but not limited to): “Average Time of Day”, “Average Length of Stay”, “Earliest Arrival Time”, or “Latest Leaving Time”. This proposed method has an implementation that is described further in chapter 6.

CHAPTER 4

COLLECTION AND PRE-PROCESSING

Currently, there is a functional data collection system called ISL^3 - *Intelligent System for Locating, Labeling, and Logging* that has been developed by Griffin et al. [45], which helps create a filtered data set by utilizing a reduced amount of user interaction to create a segmented set of GPS data streams. This software assists a user to identify spatially important locations with the click of a button, and in turn segment GPS traces into routes with minimal interaction. A drawback is the .NET platform on which the ISL^3 was initially implemented on. Only being available on a Microsoft Windows© enabled device with the .NET framework installed not only limits the data collection capabilities of the software, but also limits the number of devices it can be implemented on. With these problems in the previous generation, efforts were made to create an Android© version of the software and will be covered at the end of this chapter.

4.1. Motivation

A large collection of data will allow researchers to derive the parameters required to train prediction models and also get the trends of user behaviors in general. One key component of any prediction model is a user's history from which the models are based upon. The reliability of a collected data set will significantly affect one's ability to ascertain a users history which in turn will impact the overall performance of a prediction model. In addition, researchers tend not to share these types of sensitive data sets, typically due to privacy issues. Therefore, it becomes necessary for researchers to have a convenient and correct tool to ease this burden of data collection. In recent history, many researchers have used GPS devices to collect a user's location history. Location data can be either passively collected without a user's input or actively collected where a user provides labels to their destinations. Passive data collection requires no user attention. However, the post labeling (or classification) of destinations was done by hand in a highly interactive manner. The contribution of the tool described in this paper is in that it allows researchers to easily collect data that has a correct labeling of a user's destination in a manner that requires little interaction and places some control over the quality of the resulting logged data.

The tool went through a three generation change and each generation will be discussed in detail in the sections to follow. What will be addressed first is a prominent error inherent in GPS data that deals with the calculation of speed. This will be followed by a discussion of the parameters used to guide logging methods. These logging parameters are various configurations of speed, time, and distance. Finally, this chapter will discuss the details of each generation of logging and its accompanying resulting data collected.

4.2. GPS Speed Error

Because of the inherent signal error, and accuracy limitations in all devices (especially devices not utilizing differential GPS or DGPS[1]), speed is very often calculated to be above zero when in fact a device is stationary, particularly when calculating speed indirectly from positional data differences and corresponding time differences (see definition 8).

Definition 8: Speed and Average Speed.

Given:

`A trajectory:` T

`An object traveling along` T `traveling the distance` Δl

`A time interval:` $\Delta t = t_2 - t_1$

The speed of an object is defined:

$$v = \lim_{\Delta_t \to 0} \frac{\Delta l}{\Delta t} = \frac{dl}{dt}$$

Then the average speed of an object is defined:

$$\hat{v} = \frac{\int_{t=0}^{T} v dt}{T} = \frac{\int_0^L dl}{T} = \frac{L}{T}$$

A countermeasure to ensure that points do not get logged while currently within

[1]DGPS is an improvement to traditional standalone GPS that provides improved location accuracy from the 15-meter nominal GPS accuracy to about 10 cm in a best case scenario.

a *stay* is to place a filtering threshold on logged points while stationary. By raising the assumed speed value of zero to some value between 3-5 mph, many of the erroneous points can be eliminated during the filter step. The outer circle of Figure 4.1 shows the typical range of GPS errors. Errors within the expected range of accuracy, can show a device to be in motion even when it is not. For example, the device (in figure 4.1) is at the center of a circle with diameter of 20 meters, or approximately 65 feet. It is not unlikely to have two consecutive GPS points contain enough error to be logged at a distance similar to points A1 and A2 (almost 60 feet apart). A two second logging interval would give a calculated speed of approximately 20 mph! In fact, segments B (7.8 feet apart) and C (19.7 feet apart) provide a calculated speed of 2.65 mph and 6.7 mph respectively. Both of these segments are well within the expected range of error that all GPS units suffer from.

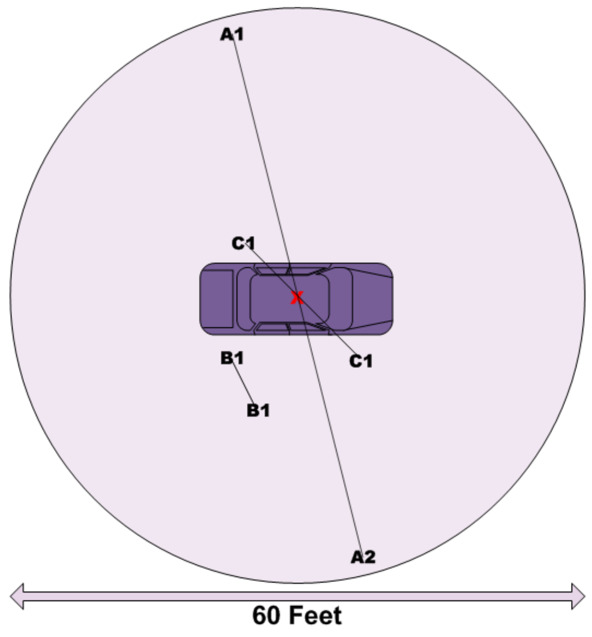


Figure 4.1. Even small errors falling within the normal error range can be detrimental to certain logging methods.

Modern standalone GPS devices implement a digital PLL (phase-lock loop) type receiver which continuously tracks the carrier frequencies of up to 12 satellites every second. The difference between the known satellite frequency as compared to the frequency when it reaches the receiver is known as "doppler shift" (see figure 4.2). The figure shows how the wavelength for the source and observer can change. The effect is amplified by the movement of one or both parties. Since the doppler shift is directly proportional to the velocity of the receiver along the direction of the satellite, regardless of the distance to the satellite, the movement of either party does not matter. With multiple satellites being tracked, it is possible to determine the 3D velocity vector of the receiver, and achieve a very accurate speed measurement known as "doppler-speed". The accuracy of doppler-speed was shown to be approximately one order of magnitude more accurate than the traditional positional data methods. The doppler-speed in a one hour sample averaged 0.063 mph whereas the traditional speed measurement was 0.551 mph for a stationary GPS unit [19].

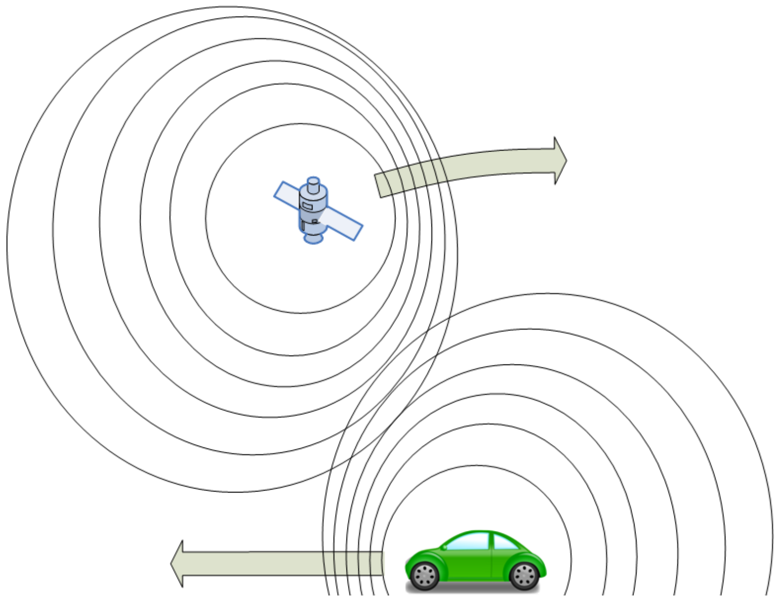


Figure 4.2. The Doppler effect is the change in frequency of a wave for an observer moving relative to the source of the wave.

Froelich et al. stated that stationary satellite receivers would sense drift and be fooled into thinking that the GPS device had begun moving even though the vehicle was parked (Figure 4.3). "These faulty readings could continue for as long as 10-15 minutes at a time, though they tended to come in shorter spurts" [33]. One method to combat these faulty readings while stopped is to set a minimum speed threshold, whereas the device would not record unless it sensed movement above this minimum speed. "Static navigation" is one such method implemented by GPS makers at the firmware level.

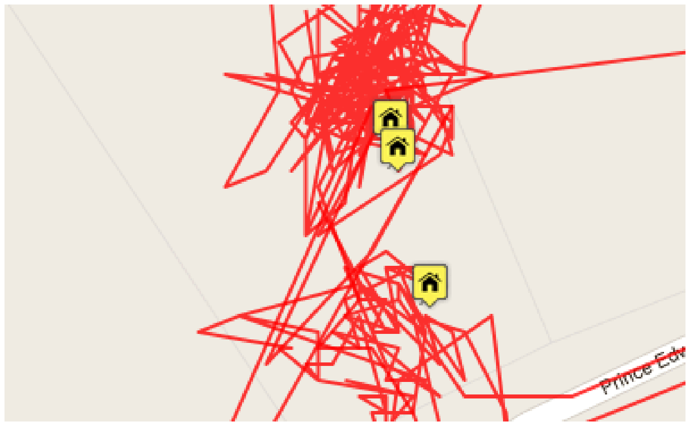


Figure 4.3. Erroneous data from a vehicle logging data despite being stationary. This particular figure shows multiple stops at a single location.

"Static navigation" is another method for improving GPS accuracy while remaining stationary. This method will "lock" the position when the device is determined to be at very low speed. This is a low level filter. Low level in the scope that it is implemented directly in the firmware of the device. It is created to cancel out the drifting that results from the natural inaccuracies of GPS. When "static navigation" is enabled by a device residing in some type of automobile, it will improve the accuracy of *stay* information despite the fact that it will not be 100% accurate. Wearable devices, or when a subject is carrying a device while in pedestrian mode, will have lowered accuracy when "static navigation" is on. The device will update its location infrequently as a result of the speed threshold mandated by the filter.

4.3. GPS Logging Methods

For a GPS receiver to log data, it must have signal reception from a minimum of 4 GPS satellites in the array of over two dozen available. A loss of GPS signal can be defined as the absence of signal from the minimum required number of satellites. Having signal reception from less than 4 satellites does not allow for accurate trilateration of a users position. If in fact there is available signal, GPS devices log data utilizing four methods: 1) continuous: or logging points continuously at a predetermined interval, 2) movement only: logs points when the GPS device senses motion, 3)distance: logs points at preset spatial intervals and 4) adaptive: a hybrid method which combines facets from each of the previous methods, coupled with a small set of refining parameters, into a single collection method. Each method has benefits and drawbacks, however all receivers suffer from some common downfalls when it comes to calculating land based speed as discussed previously in section 4.2.

4.3.1. Continuous Logging

This method logs points at regular discrete time intervals (definition 9). The time interval can be viewed as a threshold, and when the threshold is reached, a point is logged. The time threshold is some predetermined interval of time that can be set to any discrete value. In a setting where a very detailed history is needed, the threshold is set to a value between 1 and 5 seconds. The positive result of such a small interval between points, is that it allows for a detailed analysis of a users history with little or no gaps in a trajectory. As previously stated, every GPS stream has errors, but with a small time interval, the ability to filter bad points is available without large loss of detail.

Definition 9: Continuous Logging

Given:

`Discrete Time Threshold:` $I \in \mathbb{N}$

`Trajectory:` $T = \emptyset$

`A time interval:` $\Delta t = \rho_{n-1}.t - \rho_n.t$

Then:

$$(T = T \cup \rho_n) \iff (|\Delta t| \geq I)$$

The downside of a small time interval is the demand it places on the device in regards to memory constraints and battery life. Every point logged requires an I/O operation, and in turn places added demand on the devices battery. Another obvious side effect is that logging data more often uses more storage. When local storage capacity is filled, the contents must be transferred off the device, otherwise logging of data ceases. Even with the "WiFi" or "GSM" capabilities of current devices allowing them to transmit data wirelessly, and in turn save local storage, the cost to the battery must be taken into consideration.

4.3.2. Movement Only

When a unit utilizes this method of logging it places a tuple in a log file at discrete time intervals, only when the device senses the current speed to be over some threshold, implying that the device is moving (definition 10). Motion is detected by using the universally accepted definition of speed seen in definition 8.

If the speed over some interval Δl at Δt time, is over some minimum threshold then a point is logged, otherwise nothing is recorded. This can be beneficial in two ways, first, it is space saving, and second, it provides easier trajectory segmentation. It also has its downside; stationary devices still continue to sense small movements influenced by typical GPS error.

Definition 10: Movement Only Logging

Given:

`Speed Threshold:` $S \in \mathbb{N}$

`Trajectory:` $T = \emptyset$

`A distance:` $\Delta l = dist(\rho_{n-1} - \rho_n)$

`A time interval:` $\Delta t = \rho_{n-1}.t - \rho_n.t$

Then:

$(T = T \cup \rho_n) \iff (\frac{dl}{dt} \geq S)$

The obvious benefit of movement only logging is the savings achieved with smaller log files. However, the obscure benefit from this logging method is the inconspicuous segments within the log file that are devoid of logged points. This seems counter intuitive because the more data describing a behavior or action implies a better understanding of said behavior or action. However, the segments devoid of points actually represent small temporal containers that correspond to significant locations within the trajectory, providing a natural delimiter to the actual detailed route information. Identification of the natural delimiters allows for improved and accelerated route segmentation. By calculating the length of each of the temporal containers, the length of each *stay* can be deduced, giving additional insight to the behavior patterns of an individual[2]. Furthermore, the logged points between each *stay* provide a detailed representation of a *trip*.

If the accuracy and confidence in GPS signals were strong, movement only logging would be a very desirable logging method. However, the inherent error that GPS streams contain make it extremely difficult to determine a device is actually stationary with any high probability. Much of this inaccuracy comes from the methods in which speed is calculated within a GPS device. Utilizing approaches such is "doppler-speed" and "static navigation" would improve the method of movement only drastically.

4.3.3. Distance

This method, as implied by the name, adds a tuple to the log file when the current distance is greater than some distance threshold previously established (see definition 11). This method was not utilized in the collection portion of the CaPPture system, but is worth mentioning. Similarly to continuous logging, one advantage of distance logging is the ability to control the granularity of the resulting log file. By creating a small threshold, less than 50 feet, a very detailed history of activity would be achieved.

[2]It must be mentioned that a *stay* is a temporal gap above some threshold, and not all temporal gaps indicate that a driver is stopping with purpose

Definition 11: Distance Logging

Given:

`Distance Threshold:` $L \in \mathbb{N}$

`Trajectory:` $T = \emptyset$

`A distance:` $\Delta l = dist(\rho_{n-1} - \rho_n)$

Then:

$$(T = T \cup \rho_n) \iff (\Delta l \geq L)$$

A seeming obvious fact is that by changing the distance threshold, the resulting log file would be representative (in size) of the spatial distance assigned to the threshold. What is not obvious is when using the distance method is the by-product of achieving a pseudo-adaptive type log file. The adaptive method (discussed in detail next), uses parameters to adjust the type of logging and frequency of logging based on the current conditions. Distance logging does achieve something similar to the adaptive method. When a device is stationary, the distance threshold is never met, and the logging frequency goes to zero (similar to movement only), but when the device is in motion, the distance threshold is met consistently and the frequency of logging could increase dramatically. In fact, depending on the distance threshold, distance logging (when in motion) mimics continuous logging.

This does not mean that distance logging is without its downfalls. It succumbs to the same inherent GPS errors that all other logging methods do. Even when a device is stationary, due to errors, it may calculate distances over the threshold and record tuples in the log file, forcing the need for additional filtering.

4.3.4. Adaptive

One answer to the problems of each of the previous methods of logging is to use an adaptive approach. This approach combines the favorable portions of each method, while avoiding faulty ones. Adaptive logging can be configured to utilize any logging method for any specified allotment of time or duration, with any number of combinations available. The configuration would depend on the projected usage of the logged data. If memory and power is not an issue, continuous logging would always give the greatest detail of historical activity. However, adaptive logging, if configured correctly, could give the minimum necessary logged tuples increasing or decreasing in detail as needed.

As an example, one approach is to log data as if in "movement only" mode until the device stops. At this time, instead of not logging data at all, the interval at which points are logged is greatly reduced, but continues to log. For example: if the logging interval is set to one second, while the device is in motion the granularity of the log files will be in one second intervals, however, while the device is stationary, the granularity of the log files may increase greatly to every 300 seconds (5 minutes). Adaptive logging is discussed further in future work.

Regardless of the logging method used, the quality of the data collection is dependant on the ability to identify portions of the resulting collection in which data that should be there, is either missing totally, or so highly inaccurate, it is of no use. For example Froelich et al. created a set of filters that worked in succession to identify and eliminate false data. "Filtering, however rudimentary, is an essential part of our work. Before running experiments with our route prediction algorithms we had to ensure that a vast majority of our trip data was valid" [33]. Just as important as filtering the false data is the identification of missing data. How the missing data is identified is highly dependant on the logging mode being utilized.

4.4. Identifying Signal Loss

4.4.1. Continuous Logging

Continuous logging records a single record at a pre-determined interval. Signal loss can be determined by first finding the time difference (time-gap) between two consecutive tuples in the data set. If the difference is greater than some preset time interval (Δt), then a tuple was not recorded when it should have been, and signal loss has occurred.(see definition 12). Of course each logging device has different methods when creating log files.

Definition 12: Continuous Logging Signal Loss
Given:
`Discrete Time Threshold:` $I \in \mathbb{N}$
`Trajectory:` $T = [\rho_1, \rho_2, \rho_3 ... \rho_n]$
Then:
$\Delta t = \lvert \rho_n.t - \rho_{n+1}.t \rvert$ `where` $t =$ `time.`
$f(\Delta t) = \begin{cases} No\ Loss & : \Delta t \leq I \\ Loss & : \Delta t > I \end{cases}$

It is possible for devices to log tuples even though the acquired signal would result in an inadequate entry. In this case, determining signal loss would be impossible without either additional information (for example: number of satellites in view, dilution of precision), or a set of heuristics based on contextual information to identify signal loss. A set of heuristics to help determine signal loss would be to compare the values of consecutive tuples in regards to distance, speed, acceleration, and heading. If any of the compared values reaches some predetermined threshold, then signal loss is identified. In this case the definition changes as seen definition 13.

Definition 13: Continuous Logging Signal Loss

Given:

`Speed Threshold:` $S \in \Re$

`Acceleration Threshold:` $A \in \Re$

`Heading Threshold:` $H \in \Re$

`Distance Threshold:` $L \in \mathbb{N}$

`Trajectory:` $T = [\rho_1, \rho_2, \rho_3 ... \rho_n]$

Then:

$\Delta\sigma = |Speed(\rho_n.t) - Speed(\rho_{n+1}.t)|$ `at time` t

$\Delta\alpha = |Accel(\rho_n.t) - Accel(\rho_{n+1}.t)|$ `at time` t

$\Delta\nu = |Head(\rho_n.t) - Head(\rho_{n+1}.t)|$ `at time` t

$\Delta l = |Dist(\rho_n.t, \rho_{n+1}.t)|$ `at time` t

$\delta = (\Delta\sigma > S) \wedge (\Delta\alpha > A) \wedge (\Delta\nu > H) \wedge (\Delta l > L)$

$$f(\delta) = \begin{cases} No\ \ Loss & : \delta = 0 \\ Loss & : \delta = 1 \end{cases}$$

4.4.2. Movement Only Logging

It is difficult to determine signal loss when using logging methods that do not record data at a continuous regular frequency. The indication that a gap in time exists between two contiguous tuples is not necessarily indicative of loss of signal; it could be attributed to the type of logging method and is simply a space saving or power saving technique. The way to recognize signal loss in a movement only scenario is to calculate the distances between consecutive points, and if the respective distance for each point pair is over a predetermined threshold, there is loss of signal (see definition 14).

Definition 14: Movement Only Signal Loss

Given:

`Discrete Distance Threshold:` $D \in \mathbb{N}$

`Trajectory:` $T = [\rho_1, \rho_2, \rho_3 ... \rho_n]$

Then:

$\Delta l = Dist(\rho_n, \rho_{n+1})$

$$f(\Delta l) = \begin{cases} No\ \ Loss & : (\Delta l \leq D)) \\ Loss & : otherwise \end{cases}$$

The general idea is that if a device is motionless, it will not log points. Once the device senses motion, it will again start logging points at a regular interval. Depending on the maximum speed of the object the device is logging data for, there is a corresponding maximum distance two contiguous logged tuples can be apart. Above this distance, the result can only be signal loss.

4.4.3. Distance Logging

Signal loss when utilizing distance logging can be identified in a similar fashion to movement only logging. (see definition 15). When the object being tracked is on the move, tuples should be logged at discrete intervals predetermined by the distance threshold. If the object is stopped, no points will be logged, but the distance between each point pair will be within the threshold. When the distance between two contiguous points is greater than the prescribed threshold, then the indication is loss of signal.

Definition 15: Distance Signal Loss

Given:

`Discrete Distance Threshold:` $D \in \mathbb{N}$

`Trajectory:` $T = [\rho_1, \rho_2, \rho_3 ... \rho_n]$

Then:

$$\Delta l = Dist(\rho_n, \rho_{n+1})$$

$$f(\Delta l) = \begin{cases} No\ \ Loss & : \Delta l \leq D) \\ Loss & : otherwise \end{cases}$$

Identifying signal loss would be a trivial problem given the number of values embedded in the the NMEA (National Marine Electronics Association) strings generated by GPS most chips. Just as the research supporting the CaPPture system looks for methods to reduce the amount of data necessary to represent behaviors, while still maintaining accurate portrayal of said behaviors, log files also look for space saving or power saving techniques when generating logs. Therefore, an abundant amount of GPS data is stored in formats that require processing to identify loss of signal. In fact, some widely used formats remove the temporal component of GPS data rendering it useless for certain uses. Others reduce it to its basic components [latitude,longitude,time], requiring many processing and filtering steps

One method of acquiring pristine pre-processed GPS data is to create and control the mechanism in which it is collected. The next section discuss three generations of logging methods, each improving the effectiveness of the control mechanism and its resulting data.

4.5. The First Generation

A typical first generation GPS data-logger is something similar to the *Genie GT-31/BGT-31* [65] as can be seen in figure 4.4. This is a passive logger, with a small firmware operating system that allows certain parameters to be set for a minimal amount of control over the logged data.

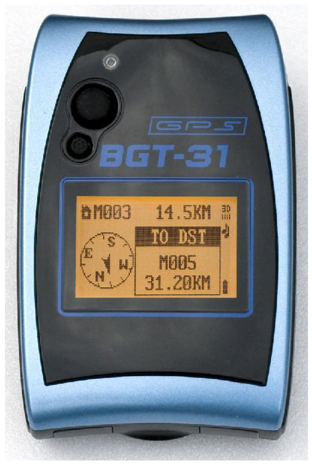


Figure 4.4. Bgt-31 Data Logger

Some of the more important features are highlighted in the table 4.1. Even with the control given by the built in firmware, this still is a passive logger and much of the motivation behind the number of filters and processing steps incorporated into the CaPPture system.

Table 4.1. Features of BGT-31

Logging Interval	The logging interval can be changed from 1 to 60 seconds.
Minimum Speed	This is a threshold setting from 1-50mph. No points will be logged at speeds below this value.
NMEA Sentences	Choose from a variety of NMEA sentences.
SD-Card	The device will log directly onto a variable sized micro-SD card.

A description of the available NMEA sentences can be seen in table 4.2. These are not proprietary to the BGT-31; NMEA is a format defined by the National Marine Electronics Association for interfacing marine electronic devices, and is the widely recognized standard for formatting GPS location information.

Subset of Available NMEA sentences	
GGA	Global Positioning System fixed data.
GLL	Geographic position - latitude/longitude.
GSA	GNSS DOP and active satellites.
GSV	GNSS satellites in view.
RMC	Recommended minimum specific GNSS data.
VTG	Course over ground and ground speed.
ZDA	Date and time.

Table 4.2

The BGT-31 data logger is one the of premier GPS devices used in competitive water sporting events such as wind surfing and sail boating. It is used because of its accuracy in calculating speed by way of utilizing doppler-speed. However, as advanced as the firmware is in allowing a very customizable configuration that goes beyond the short description of features in table 4.1, it still is a passive device. Passively collected data must be dealt with by way of real time filters or a large amount of post processing, that includes a large amount of user interaction by way of labeling, confirming, and filtering directly by the user whom collected the data. However, with a small amount of interaction during the collection process, the resulting logged data can be many times more usable.

4.6. The Second Generation

One of the major problems in previous methods of data collection was the emphasis placed on user interaction after the data was collected. A user would typically place a passive logging device inside their vehicle, in addition to maintaining a travel diary (written accounts of arrival times, departure times, and trip purpose). The second generation of data collection implemented by the CaPPture system is known as ISL^3. This implementation reduces the need for active user interaction by making intelligent decisions based on a users' history of travel. This moves the data collection effort away from written diaries in conjunction with GPS loggers, towards the passive end of the scale, by giving the user a "travel diary" type interface via software. Figure 4.5 shows where on the scale the second generation of data collection falls.

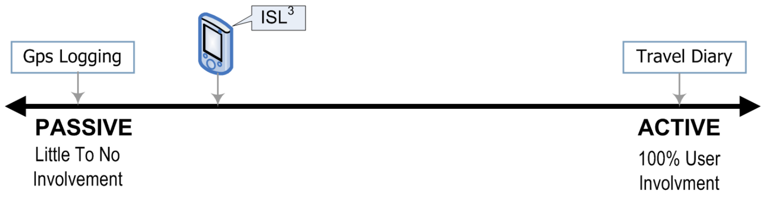


Figure 4.5. By making the data collection process as passive as possible while still gaining useful information is a vital beginning to a pristine data set.

Figure 4.6 shows the architecture for ISL^3. The implemented portion of the architecture uses a handheld device coupled with the .NET framework. The .NET framework allows for the creation of a user interface, communication with the GPS receiver, and maintenance of the GPS library. The 1^{st} and 2^{nd} generation of the architecture as implemented, required frequent downloads of the collected data onto a central server. The 3^{rd} generation allows for the use of either a WiFi hotspot or a GSM network for the seamless upload of data to the central server at set intervals (discussed in section 4.7). The removal of the download step will improve the convenience of using any data collection software.

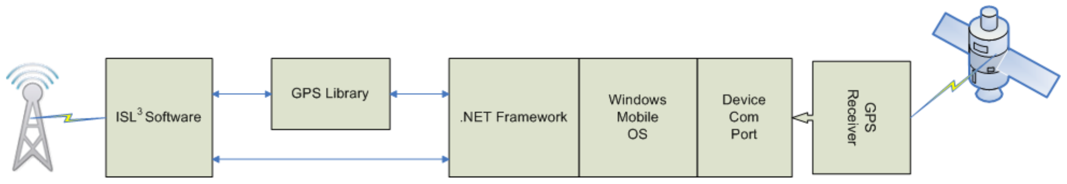


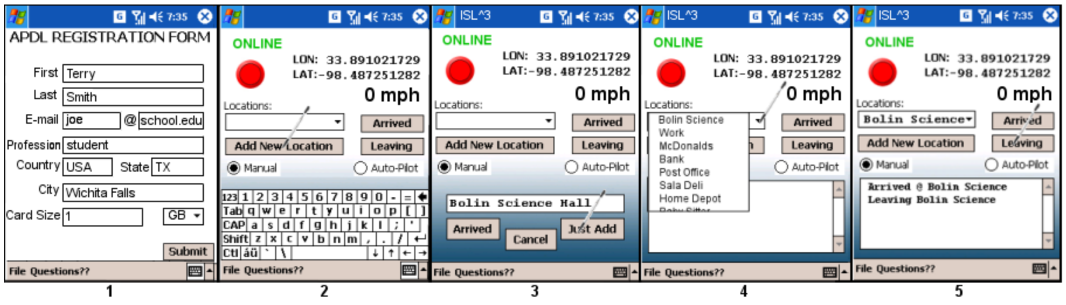
Figure 4.6. ISL^3 2^{nd} Generation Architecture

The user interface for ISL^3 allowed for users to raise the quality of a passively collected GPS stream, into a data set that had defined and labeled *stays*, which in turn allowed for each route to be identified. When a user arrived at a location, they could actively choose "Arrived" or use the "AutoPilot" mode and be automatically checked in when arriving. By identifying each user stop, routes could then be identified with high accuracy. Stops can be identified programmatically, but there is always room for error. For example, what is a decent threshold for minimum stop time? Most researchers choose values from 2-5 minutes, but it is entirely feasible to perform a task in under two minutes (bank drive-through for example). It is also feasible to be stopped for over two minutes without performing a task (stop light). With just a small amount of user interaction, the collected data raises its quality from a passively collected error riddled data set to a high quality low error data set. One type of GPS error that must be addressed because of the impact it plays on GPS logging methods, is "speed". Speed, its accuracy, and the methods in which it is calculated have an adverse effect on data collection process and impact nearly all types of logging methods and accompanying filters. ISL^3 displays an easy to use and intuitive interface for users. This plays into the overall purpose of the software itself, which is to give the everyday user the ability to easily collect a complete and fully labeled travel diary. To start the program the user clicks on the "play" button. As soon as the GPS enters in contact with the array of GPS satellites, the longitude and latitude of the current location are displayed at the right upper corner (see figure 4.7). The concept is simple: 1)Add destinations, 2)Record arrivals and departures in respect to those destinations.

To add a destination, the user simply presses the "Add Destination" button (see figure 4.7 - 2), showing the keyboard allowing the user to type a short descriptive name for a destination. This adds the destination to a *Destinations* file that contains a unique id for the destination and user, along with the coordinates of that destination. This file populates a drop down list that is accessible on the main screen (see figure 4.7 - 4). This drop down list is continuously sorted, where the top of the list is the destination closest to a users current location.

When the user needs to record arrivals and departures, they simply press the *Arrived* or *Leaving* button depending on the circumstance. The action is recorded in the *Actions* file with the specified destination. To specify a destination the user selects it from the drop down list. In addition to a record being appended to a file, visual assurance is given to the user in the form of a log entry on the users screen (see figure 4.7 - 5).

Initially the interaction by the user is relatively low, but will diminish as the users typical destination are entered. A typical location entry consists of adding each location as


APDL REGISTRATION FORM

First: Terry
Last: Smith
E-mail: joe @ school.edu
Profession: student
Country: USA State: TX
City: Wichita Falls
Card Size: 1 GB

ONLINE
LON: 33.891021729
LAT:-98.487251282
0 mph

Locations: Bolin Science, Work, McDonalds, Bank, Post Office, Sala Deli, Home Depot

Bolin Science Hall

Arrived @ Bolin Science
Leaving Bolin Science

Figure 4.7. 1)User registration. 2)Adding a new location into the system when arriving. 3) Adding a new future location. 4)Choosing which location the user arrived. 5)User leaving location.

it is encountered (meaning a record would be logged in the *actions* file), but the software does have an option to "Just Add" a location. This allows a user to populate the list with their typical locations before they are actually visited. The one drawback to this method of destination entry is that the destination will have no Lat-Lon pair associated with it until the first time it is physically visited. Consequentially that location would not bubble to the top of the list as the user approaches it (for reasons explained earlier). ISL^3 has two modes of operation: 1)Standard and 2)Autopilot. Standard mode operates as previously discussed. Autopilot mode requires slightly less interaction.

Autopilot mode works much like a GPS logger in the fact that it senses motion. If the "Autopilot" senses that the user is not moving (based on a distance function between contiguously logged points), it starts a timer. When the timer exceeds some predefined threshold, the "Autopilot" will record an arrival at that destination. When the "Autopilot" senses movement, based on the same distance function, it records the user leaving the destination in the Actions file.

4.6.1. Architecture Problems of the ISL^3 Software

A smart device communicates with some form of GPS antennae to obtain its location information. Smart devices are not limited to using GPS antennas to obtain location information. However, the original software depended on one being present, meaning GSM was not available in that version. When the device receives a GPS stream, it receives it from a communications port (comm port). Even if a GPS antennae is embedded within the smart device, it still communicates via a comm port. Each brand of hardware will determine which comm port is used for GPS antenna communication, and the choice of which port to use is platform dependant. To ensure a more robust software package, comm port scanning is necessary. The application scans through the ports of the handheld device to identify the correct port receiving the GPS stream. To do so, the application attempted to utilize the native functionality of the .Net framework. This should allow the system to perform *try* and *catch* blocks in an attempt to identify the correct comm port; however a peculiar “feature” of crashing arose when attempting to *try* and read an unutilized port using .Net functionality. Some “outside the box” methods did allow the correct port to be identified and communication established. This left ISL^3 with raw NMEA data strings and no native parsing capabilities. The “GPS Library” shown in the architecture (figure 4.6) was a layer of filters and conversions turning the raw data into usable location information.

The main program manages most aspects of the software. Many of the management choices are predetermined by a configuration file. It determines which data are filtered and passed by the GPS library to the main program. The configuration file also determines the frequency and format in which records are logged. Other features that are stored in the configuration file consist of the maximum allocated memory for storage, a radius to help in determining destination arrival, and maximum destination definitions. All of these parameters did allow ISL^3 to be highly configurable, but still not comparable to the programming API's and SDK's currently available.

4.6.2. Hardware Used

ISL^3 is designed to work on .Net Compact framework 2.0 architecture or above. The application was tested using two type of Pocket PC handheld device. The first handheld was the T610 model from the AIRIS manufacturer. The AIRIS T610 has an integrated GPS chip embedded in the device. The second Pocket PC is a WAYPOINT PDA coupled with an external GPS device via a Bluetooth connection. Both devices were equipped with Windows mobile 2005 operating system. The WAYPOINT PDA had shown a better battery life saving and a quick response in getting satellite connection. However after a long period of inactivity, the Bluetooth connection was frequently lost which forced the user to restart the ISL^3 program. The Integrated GPS enabled device seemed more suitable for auto-pilot mode. As long as the device was connected to external power, it remained on and functional. During the testing phase, the AIRIS T610 was set up for a week in a car, on auto-pilot mode. With no interaction, the data collection completed successfully without interruption.

4.6.3. Managed Files

ISL^3 produces four different files: 1)User File- contains information about the user of the software. 2)Destinations File- is used to store added destinations by the user. 3)Actions File- contains the activities of the user. 4)Log File- stores the raw GPS data. The division of stored information was designed in such a way as to ease the extraction of important information from the collected data set, along with reducing storage requirements for the device in use. A relationship between the main data files can be seen in figure 4.8.

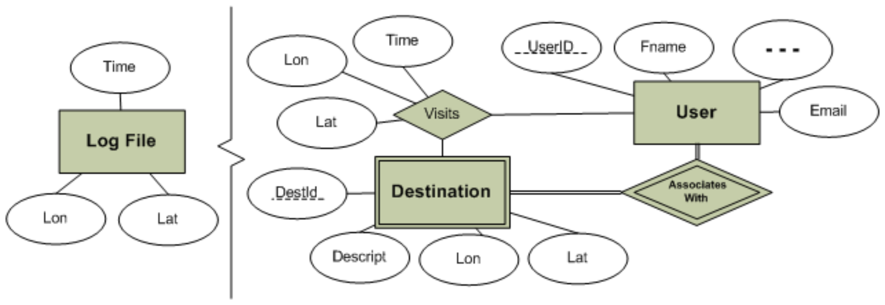


Figure 4.8. ISL^3 ER-Diagram

4.6.4. Data Collected

The ISL^3 software was tested by four individuals in the North Texas area. The users all recorded data continuously every 5 seconds as long as the unit was on. The collected data is summarized in table 4.3 with an aerial 3D view of user activity shown in figure 4.9.

User	Logged Records	Destinations Defined	Activities Labeled
User 1	1855	6	98
User 2	50040	29	256
User 3	14193	22	418
User 4	11153	22	310
Totals	**77241**	**79**	**1082**

Table 4.3. Summary of collected data by ISL^3.

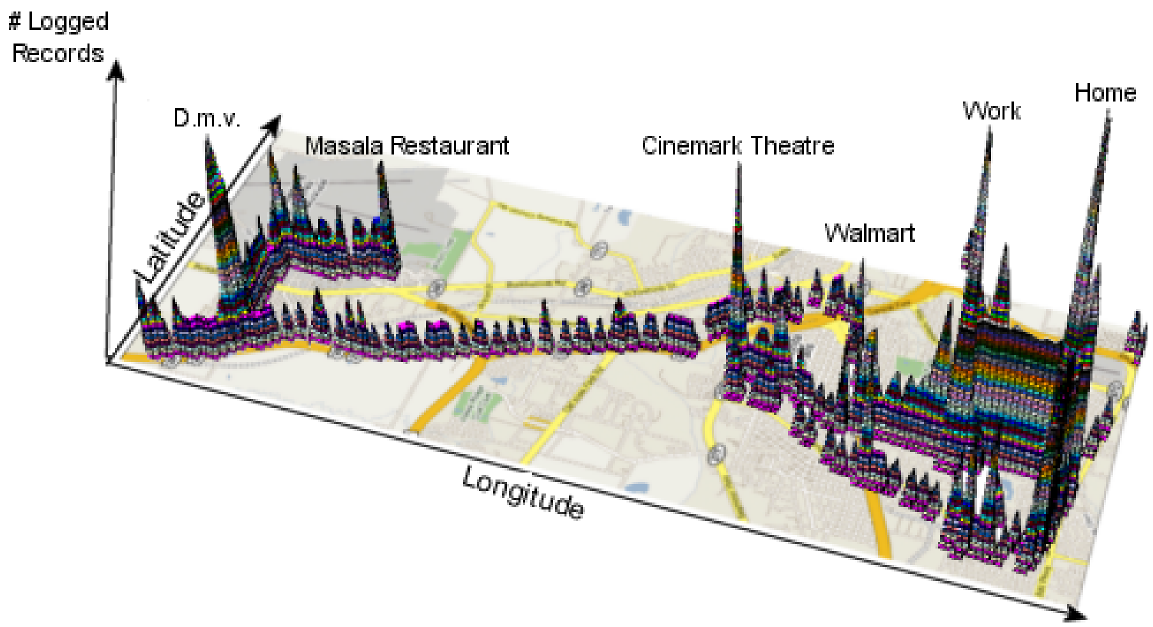


Figure 4.9. Wichita Falls Area Visualization (log scale).

4.6.5. Improvements to ISL^3

The 2nd Generation of ISL^3 (see figure 4.6) targets the Smartphone class of handheld devices. This will help satisfy four goals: 1) To reach a much larger audience of potential data collectors. 2) Get easier access to internet based technologies. 3) Implement real time functionality to our software. And most importantly 4) Design and implement location/destination prediction algorithms.

According to CTIA (International Association for the Wireless Telecommunications

Industry) over 250 million individuals in the United States own cell phones. When factored against the latest U.S. census, this places cell phone usage at over 80 percent. So by simply changing platforms we gain a potentially large subject base. In addition to a very large volunteer base, smartphone technologies are typically accompanied with access to the internet. This alleviates the storage problem on the local device, and allows the software to send data to a centralized location. This centralized location could be used to also return real time data to the individual giving them access to any number of mass user based applications. Directly related to this is the potential improvement in prediction algorithm research. With access to large amounts of correct and labeled user histories, it is possible that researchers will create more accurate and robust prediction algorithms that in turn will improve location based research as a whole.

4.7. The Third Generation

The third generation of the CaPPture data collection system was motivated by the outbreak of cloud computing (albeit in the not to near past). At the time of the 2^{nd} generation of the systems collection software, cloud computing existed but was more of an idea than a fully implemented distributed computing paradigm. The "cloud" in recent years is a well understood concept and has many well documented API's to allow software the functionality to was not once available.

The third generation of the CaPPture system takes advantage of the available scaffolding pieced together by many cloud friendly services, and mobile operating systems built to take advantage of WiFi, GSM, and GPS. In addition to large public API's like "Dropbox©", additional open source projects like "github" create a social addition to software development. In fact the third generation of the CaPPture logging software is based on an open source code base "GPSLogger for Android" authored by Mendhak and released under the GPL v2 license [74]. This software provided the basic functionality necessary to utilize the processing power of contemporary hand held devices, and the network capabilities necessary to remove the physical download process from the collection device. The breakdown of the collection model is very simple, mostly due to the fact that many of the libraries are available, either natively or as a third party additions, to the Android Software Development Kit (SDK). The simplicity of the 3rd generation software can be seen in figure 4.10.

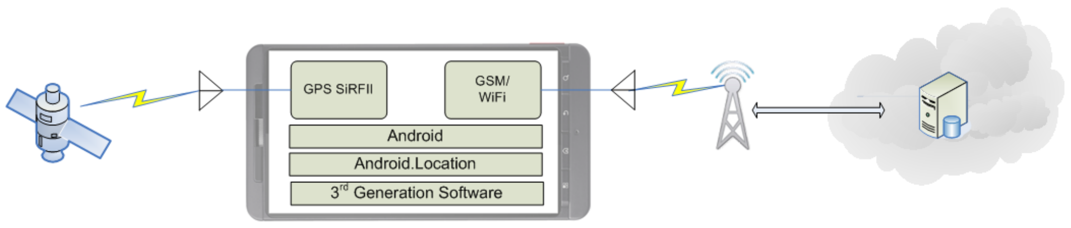
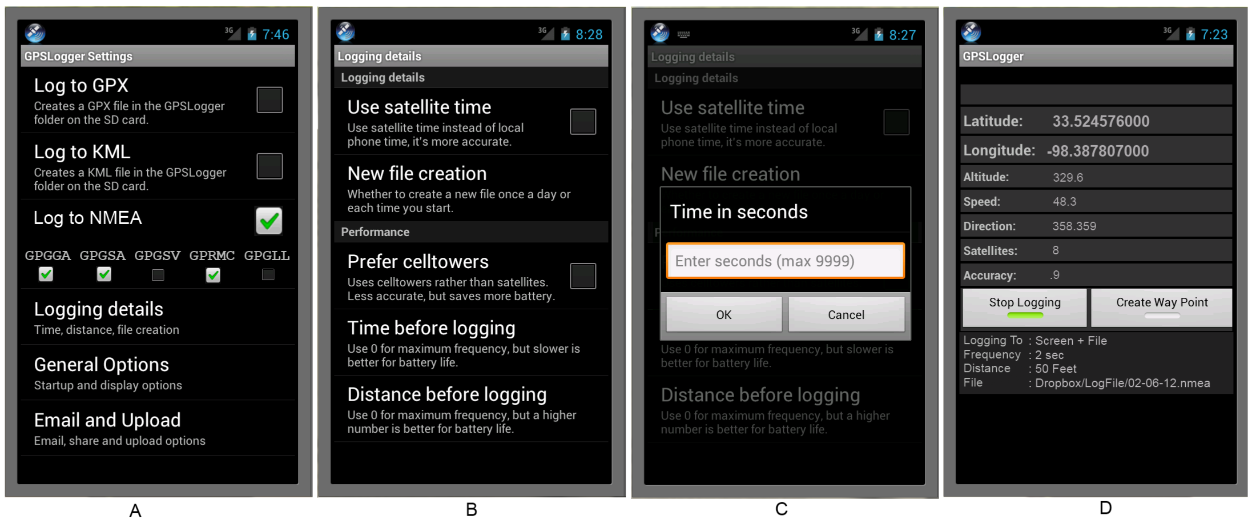


Figure 4.10. 3rd Generation Architecture



GPSLogger Settings

Log to GPX
Creates a GPX file in the GPSLogger folder on the SD card.

Log to KML
Creates a KML file in the GPSLogger folder on the SD card.

Log to NMEA ✓

GPGGA ✓ GPGSA ✓ GPGSV GPRMC ✓ GPGLL

Logging details
Time, distance, file creation

General Options
Startup and display options

Email and Upload
Email, share and upload options

Logging details

Use satellite time
Use satellite time instead of local phone time, it's more accurate.

New file creation
Whether to create a new file once a day or each time you start.

Performance

Prefer celltowers
Uses celltowers rather than satellites. Less accurate, but saves more battery.

Time before logging
Use 0 for maximum frequency, but slower is better for battery life.

Distance before logging
Use 0 for maximum frequency, but a higher number is better for battery life.

Time in seconds
Enter seconds (max 9999)
OK Cancel

GPSLogger

Latitude:	33.524576000
Longitude:	-98.387807000
Altitude:	329.6
Speed:	48.3
Direction:	358.359
Satellites:	8
Accuracy:	.9

Stop Logging Create Way Point

```
Logging To  : Screen + File
Frequency   : 2 sec
Distance    : 50 Feet
File        : Dropbox/LogFile/02-06-12.nmea
```

Figure 4.11. 3*rd* Generation Examples: A) File format choice. B) Adjust logging time, or distance. C) Model to enter custom values. D) The data logger output.

An example of the added simplicity can be compared to one major component implemented in 2nd generation. It was to provide an "AutoPilot", in which a user would be automatically logged as "arrived" to a previously defined *destination*, in turn keeping track of daily activities with very little interaction. This functionality was implemented at the lowest levels, including distance calculations between a users current location and all current saved *destinations* in the devices database, calculated every few seconds and taxing resources. However, in the Android LocationManager class, a simple method call to "*addProximityAlert*" provides the exact same functionality. Even if the addProximityAlert is implemented in much the same manor as CaPPture's 2^{nd} generation, it still provides a layer of abstraction allowing better engineering at a faster pace. Since this method is native to Android, its portability fares much better (although never 100%), along with the many additional native functions replacing the previous non-portable implementation hindered by the .NET framework and many device specific features embedded in the code. Figure 4.11 shows examples of some of the features implemented in the current generation.

4.7.1. Summary of Data Collected

Currently the 3^{rd} generation of the data collection software is being tested by four individuals in the North Texas area. The users all recorded data continuously every 5 seconds as long as the unit was on. The collected data is summarized in appendix B. The summary shows a culmination of data collection efforts over the past several years with many volunteers

The goal behind this data collection effort has always been to gather enough information about user activity to then create robust accurate models that would predict the behavior or location of individuals. What ultimately happened is that the amount of knowledge in the data collection process itself became an entity of its own. Figure 4.12 shows the progression of the quality of data spanning each generation. It is a small data set, but the trend line clearly shows that the amount of data needing filtering from the first generation to the last generation clearly goes down, meaning the latest sets of data are more complete and understandable.

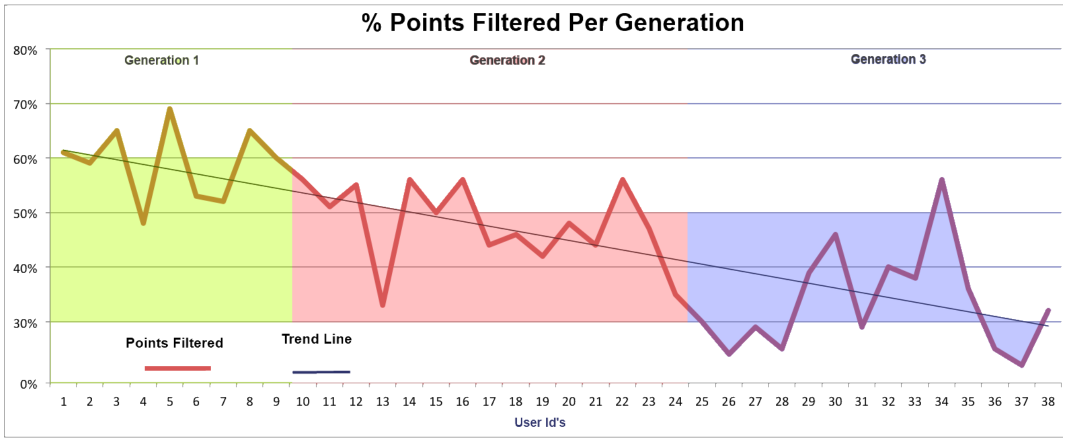


Figure 4.12. Initial quality of each generation of collection.

CHAPTER 5

MAP-MATCHING

5.1. Introduction

This chapter introduces a novel offline routing based map-matching approach that is used as a filter step in the CaPPture system. This approach is used for standardizing previously identified routes within a collected set of GPS trajectories. Map-matching is the process of matching data points of a GPS trajectory to a known road network (described in chapter 3). In general there are two types of map matching: online and offline. Online map matching is performed in real time typically for use in vehicle navigation systems. Offline map-matching is performed *a posteriori* with regard to the collected GPS data. This method is primarily used for the analysis of historic travel data and can also be used as a means of data cleansing, by snapping small errors within the data set to the underlying road network. When trying to identify a repeated activity by a user, variations in the driver's trajectory or additions of GPS error, can make a commonly traveled route seem different on each occasion. Individually, each variation can seem trivial, but the synergism of all the variations together can be problematic when comparing like routes.

The purpose behind the routing based map-matching approach described in this chapter is to normalize each path that is taken between the same starting (V_i) and ending (V_j) points[1]. In [33], Froehlich et al. calculated a similarity score when comparing paths using an algorithm based on the Hausdorff distance between two paths [23]. One of the potential downfalls when comparing two raw paths is the small differences between each path with regard to GPS error. Research during the writing of this thesis has found that each path, even between the same A and B, have enough small differences between them, to create significant doubt in any similarity score (see figure 5.1).

[1]V_i and V_j are also referred to as A and B.

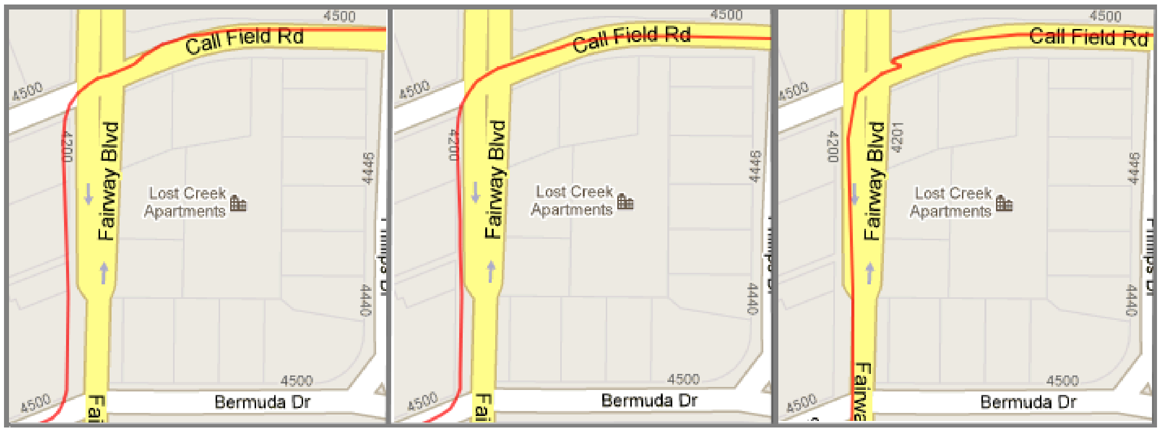


Figure 5.1. The same snapshot of a route driven three separate times connecting the same V_i and V_j.

Realizing the small differences (even between paths connecting the same A, B), the goal of this map-matching method is to take each path and provide a standardized map matched version of the path in order to ensure that every path connecting A, B are identical. Identical paths are one of the necessary cornerstones in the creation of a “clean” GPS data set. Whether algorithms are being developed for location Prediction, route Prediction, or location Labeling there is a necessity for a pristine data set in which each algorithm can be trained and tested against. Correctness is one of the major, if not the main concern, in the development of prediction algorithms, and by eliminating “noise” from the training set, a researcher can add a level of certainty to the algorithm’s results.

5.2. Overview

An abstract description of the map-matching method described in this study can be broken down into three parts. An overview of the method can be seen in figure 5.2.

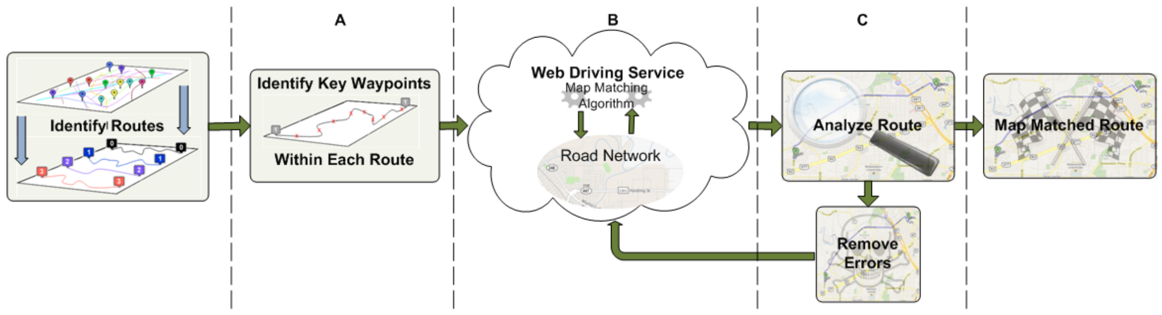


Figure 5.2. Flow of the web based map-matching algorithm.

Given a data set, with routes[2] for individual users identified, the map-matching algorithm uses the following approach:

A. Identify key waypoints

Identify "important waypoints" (*i-point*) in a route using an improved "Peucker–Douglas–Ramer" (IRDP) curve reduction algorithm. The necessity to modify the original RDP is due to it's inability to specifically bound the number of points in a resulting representation, while simultaneously making optimal choices of which points best represent the route. These two improvements are key when using the black box algorithm of the web based service that provides the bulk of the map-matching logic. Intelligently selecting, and bounding the number of points sent in a request that is sent across networks, is necessary for efficient use of resources.

B. Web based driving service

Send the *i-points* identified by IRDP to a web based driving directions service, which returns a map matched route utilizing the set of *i-points*. This returned route is a standardized representation of the original GPS trajectory constructed using the minimum necessary set of points, but

[2]The terms "route" and "curve" are synonymous in this discussion.

some of the initial *i-points* may have been incorrectly matched and must be filtered. This is the motivation for using an intelligent selection algorithm for identifying these points.

C. Filter and refine

This step (if needed) identifies the incorrect portion of the returned route. If an *i-point* is identified as being problematic (meaning it leads to an incorrect matching), the filter will eliminate it. This refined set of *i-points* is submitted the second time, resulting in a correctly map matched version of the original route.

Each of the three parts of the map-matching approach are described in detail in the sections that follow.

5.3. Identify Key Waypoints

The most essential component of this map-matching method (figure 5.2[A]) is an algorithm to identify *i-points* within a given route. The motivation behind identifying key points within the trajectory is to keep web traffic minimized. By using a web based driving directions service, many across-network requests will be performed, and limitations on the size of a request should be imposed.

Each web based driving directions service has a corresponding Application Programming Interface (API) allowing a developer to utilize their services. Each API gives strict specifications (and restrictions) as how to structure a request. Most of the restrictions relate to the number of requests allowed within a set time period and the number of points allowed within a request. The specifications also describe methods for packaging data being sent to the driving service. This is typically done using Javascript Object Notation (JSON) or some variation of an Extensible Markup Language (XML). These two examples are a fraction of the specifications provided by each API.

To fulfill the specifications provided by the driving directions service[3], an algorithm

[3]The map-matching method in this thesis utilized Google Maps® as its primary map-matching engine.

to reduce the number of points in a route needed to be investigated. One such algorithm that performed this reduction is the RDP algorithm described in chapter 2.

5.3.1. Point Selection Methods

Several avenues of point selection were explored. The following section briefly describes a couple of the more successful point selection algorithms that were analyzed. This section ends with the best point selection method. It describes the original RDP and the improvements made by IRDP which outperformed every other point selection algorithm it was tested against.

Equidistant

The first point selection algorithm described here is based on a simple heuristic. The idea is to evenly space selected points throughout the route. In theory this will not allow the Driving Directions (DD) service enough leeway to make its proprietary choices altering the original route, forcing it to use only the original road segments. The point placement was not consistent in regards to representing the route. This method was a very simple implementation, but had no regard for the identifying features of a route. For example, a long straight segment within a route does not need a series of evenly spaced points to ensure the DD service chooses this segment. One strategically placed point will suffice and ensure the segment gets utilized in the subsequent map-matching.

Turn Detection

The turn detection algorithm is based on one created by Husnain et al. [68]. The implementation within the CaPPture system can be seen in the algorithm in definition 16. Generally, by calculating the tangent angle θ for every point, and then taking the derivative of the angle with respect to the distance traveled ($\frac{\Delta\theta}{\Delta s}$, where $\Delta\theta$ = Change in angle, and Δs = change in distance) also for every point, the beginning, peak, and end of a turn can be identified.

Definition 16: Turn Detection Algorithm

```
   Input: $P = \{p_0, \ldots, p_n\}, \epsilon, WindowSize$
   Output: $P' = \{p'_0 \ldots, p'_n\}$ where $P' \subseteq P$
1  HalfWindow = $\lfloor WindowSize/2 \rfloor$;
2  for $i \leftarrow (HalfWindow + 1)$ to $(length(P) - (HalfWindow + 1))$ do
3      for $j \leftarrow (i - HalfWindow)$ to $j \leq (i + HalfWindow)$ do
4          $\Delta\delta \leftarrow Dist(P_{j-1}, P_j)$
5      for $j \leftarrow (i - HalfWindow)$ to $j \leq (i + HalfWindow)$ do
6          $\Delta\alpha \leftarrow BearingChange(P_{j-1}, P_j)$
7      $\tau = \frac{\Delta\alpha}{\Delta\delta}$
8      if $\tau \geq \epsilon$ then
9          $P' = P' \cup P_i$
```

Calculating θ for each GPS point P_t, works by computing the angle between the first and last points within a sliding window, ω, surrounding P_t. The sliding window grows forwards ($\omega = \omega \cup P_{t+n}$) and backwards ($\omega = \omega \cup P_{t-n}$ as long as P_{t+n} and P_{t-n} are within some minimum threshold distance from P_t. When the window surrounding P_t stops growing (no points are within the minimum threshold), a tangent is drawn between the first and last points in ω. This is the angle θ for P_t. This is repeated for each point $P_t \in P$. Calculating the distance, s, for each GPS is done in the exact same fashion by using the distance from the first and last points in ω and assigning that as a distance for point $P_t \in P$.

Once the angle θ and distance s are calculated, a derivative $\frac{\Delta\theta}{\Delta s}$ for each GPS point is calculate and assigned to P_t. Δa is the change in heading or angle and Δs is the change in distance with regard to the first and last points in ω. The further the value of $\frac{\Delta a}{\Delta s}$ is from 0, the sharper the turn on the route, and by watching the increase and decrease in $\frac{\Delta a}{\Delta s}$, we can deduce the beginning and end of a turn.

The turn detection of point select uses the GPS point closest to the peak, or crux of the turn as one of the points submitted to the DD service. As one might notice, there may be very few points utilized to represent a route when using just turn detection, and this did pose a problem.

Centroid Based Turn Detection

This method was also modeled after the method used in Husnain et al. [68] for calculating the centroid for a turn. The centroid is calculated similar to what the center of mass for a body would be calculated as:

$$Centroid = \frac{\sum A_N C_N}{\sum A_N}$$

where $\frac{\delta\alpha}{\delta s}$ is a weight function A_N, and C_N is the index of the N^{th} GPS point between the start and end of a turn.

Points residing within or close to turns caused problems when submitted to a DD service. This is discussed in detail in section 5.5, but in general it was due to the fact that turns (corners) offered many road segments to which they could be matched to. Given the small error always present in GPS signals, it proved to be a poor point selection method.

Discovering that turns tended to be poor choices, parameters were introduced to enhance point selection algorithms based on turns. These parameters allowed points to be chosen either before, after, or on both sides of the turn based on some pre-determined distance. This would allow the submitted points to be within some distance of a turn, but keep such points away from the problematic intersections. By setting the parameters from tens to hundreds of feet, a turn could have points both before and after a significant distance away, theoretically ensuring the road segment before a turn and after a turn both get included in the map-matching.

Inverse Peucker

The RDP and IRDP algorithms tend to function as pseudo corner detectors when used to process route data. That is, the endpoints of the line segments are typically in the location of route turns/corners. This is due to the fact that corners tend to have a larger perpendicular distance from the line segments when evaluated by both algorithms. By having little or no control over how many points were returned, RDP, tended to deteriorate even more so into a corner detection algorithm. The benefit of IRDP is that it ensures a larger portion of the original curve is still represented and does not cluster around corners. This does not however, totally alleviate the problems that corners present.

Random

This was used as a base line comparison function to ensure that all of the design and logic put into each point selection algorithm did not just degrade into something that was equivalent to or out performed by random selection.

5.3.2. Comparison of Point Selection Methods

The performance of each point selection algorithm can be seen in figure 5.3. The metric used was to count how many points were identified as incorrect after the initial submission to the DD service was returned. Although the research started with the assumption that a single submission to a DD service would be all that was necessary in fulfilling the map-matching step, it was soon discovered that a second submission would be necessary to ensure 100% accuracy. Therefore, the point selection algorithms were ranked based on the percentage of initially submitted points that needed filtering before the second submission.

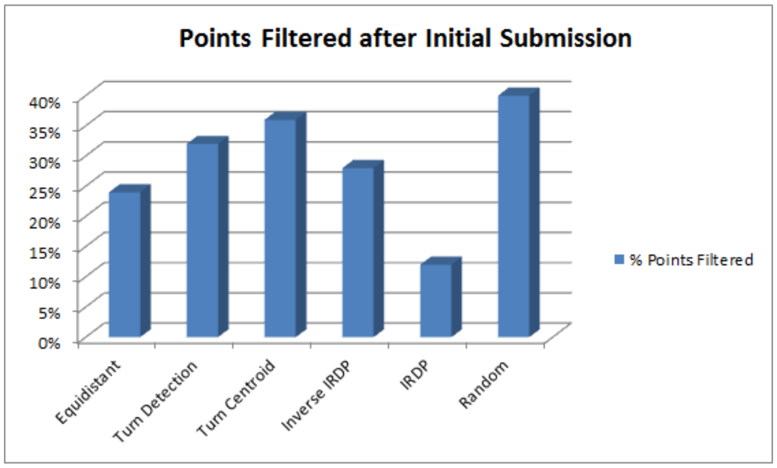


Figure 5.3. Percentage of points removed after initial submission to DD service.

The ability of IRDP to make intelligent selections of which portions of a curve to represent is not just substantial in its performance when compared to the other point selection algorithms, it is mostly substantial because the next component in the map-matching algorithm requires communication with a web based DD service. It would be entirely infeasible to send the entire corpus of points (670 points for the average route) to the DD service for every map-matching request. Therefore, by reducing the number of points submitted, and by outperforming the other point selection algorithms in choosing good points, the turnaround time for matching a route is held to an acceptable level.

5.3.3. Improved Ramer–Douglas–Peucker

One problem with using the RDP algorithm to reduce the number of points in a route, was controlling the number of points in the final representation. This is indeterminable based solely on the adjustment factor, ϵ. By increasing or decreasing the value of ϵ, one could control to some extent the percentage of points returned from the original route, but by no means had any real control over a specific number of points that appropriately represented the route. Additionally, because of the algorithm's recursive implementation, placing an upper bound on the returned points results in points that are not representative of the entire curve.

Recursion is a common method of simplification to divide a problem into subproblems of the same type. This is also called divide and conquer and is key to the design of many algorithms. Divide and conquer serves as a top-down approach to problem solving, where problems are solved by solving smaller and smaller instances. For example, an array of points can be processed recursively by continuously passing indexes that are closer together, and processing (what is perceived to be) a smaller and smaller instance of the array. When the instance of the array gets to a predetermined size, a boolean test can opt to stop the recursion. This boolean test is also known as a base case.

Since the algorithm calls itself (creating a duplicate copy), instructions within the algorithm are executed in the same order. As can be seen on line 9 in algorithm 1 (RDP), the indexes being passed to the next instance of the algorithm range from $p_1 \rightarrow index$, where index is the next point in the array that has the furthest perpendicular distance from the curve. For each subsequent recursive call, the values $p_1 \rightarrow index$ are passed, moving index closer and closer to point p_1, creating a smaller instance of the array to process, but also processing only one side of the curve first.

This is not a problem in typical recursive solutions, in fact eliminating large instances in each subsequent recursive call can be preferred. For example, when searching for a key value, eliminating portions of the search space is desirable, and will speed up the search time. In the case of RDP, however, the entire curve needs to be processed, finding the most representative set of sub points within the curve. By placing an upper bound on the desired result, a possibility of not ever processing points on the opposite side of the index occurs when the upper bound is met quickly (never getting to line 10 in algorithm 1 (RDP)).

The original implementation evaluates each points' perpendicular distance from line segments on the left side of the index or "pivot point". The "Pivot Point" is the point that has the largest perpendicular distance from a line connecting two points A and B. Initially A and B are the starting and ending points of a curve, and the pivot point would be the third point added to the result.

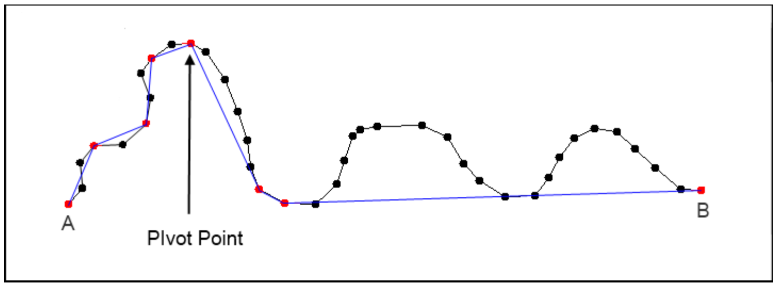


Figure 5.4. Results of the original RDP with an upper bound of eight points.

If a point was within ϵ distance from the segment, it was included. In figure 5.4, the bound is set to eight and based on the chosen value of ϵ, the curve to the right side of the pivot point did not receive an acceptable representation. If the upper bound were six, no points on the right side of the pivot point would have been chosen.

Identifying the circumstances that led to the shortcomings of the original RDP algorithm led to an improved version, IRDP. First, by modifying the RDP algorithm to an iterative solution, the problem of a skewed result on one side of the pivot point could be alleviated. This obviously is not the only fix. An iterative solution, using the same distance metric ϵ, would again result with the same unknown number of points. This result may not necessarily be skewed to one side, but it would not be a distribution that is representative of the original curve either.

The solution to the distribution problem was to have IRDP make intelligent selections of which portions of the curve to next evaluate. In fact, the concept of ϵ is totally removed.

Since the goal is to only return a set of N number of points to represent the curve, the algorithm must ensure it returns the best possible N points. This is achieved using the following series of steps:

1. Add first and last points to result R.
2. Find the longest polygonal chain[4] AB between any pair of points in R .
3. Find the pivot point in relation to AB, and add it to R.
4. Go to 2, and repeat until bound is met.

Definition 17: Improved RamerDouglasPeucker algorithm

```
Input: $P = \{p_0, \ldots, p_n\}, I = \{i_0, i_n\}, MaxPoints$
Output: $P' = \{p'_0 \ldots, p'_n\}$ where $P' \subseteq P$
1  $P' \leftarrow P' \cup P_0$
2  $P' \leftarrow P' \cup P_n$
3  $i = 0$
4  $j = |P| - 1$
5  $PointsSelected = 2$
6  $\lambda = Line(P_i, P_j)$
7  repeat
8      $Dmax = 0$
9      for ($n \leftarrow i + 1$ to $j - 1$) do
10         $d \leftarrow$ PerpendicularDistance($P_n, \lambda$)
11         if ($d > dmax$) then
12             $Dmax \leftarrow d$
13             $m \leftarrow n$
14     $P' \leftarrow P' \cup P_m$
15     $Sort(P')$
16     $PointsSelected \leftarrow PointsSelected + 1$
17     $MaxLength \leftarrow 0$
18     for ($m \leftarrow 0$ to $|P'|$) do
19         $PolyLength = polygonalChainLength(P_{P'_m}, P_{P'_{m+1}})$
20         if ($PolyLength > MaxLength$) then
21             $Maxlength = PolyLength$
22             $i \leftarrow P'_m$
23             $j \leftarrow P'_{m+1}$
24             $\lambda = Line(P_i, P_j)$
25 until ($PointsSelected >= MaxPoints$);
26 return $P'$
```

[4]A polygonal chain P is a curve specified by a sequence of points so that the curve consists of the line segments connecting the consecutive points.

5.3.4. Analysis and Comparison of RDP and IRDP

In figure 5.5, a side by side comparison can be seen showing the choices made by each algorithm. In the original RDP implementation, step 1 adds the pivot point, and steps 2-4 add the first three points to the result, all of which are to the left of the pivot point. In the IRDP implementation, however, already at step four a more distributed selection of points is apparent. Notice that the original algorithm does add points to the result on the right side of the pivot point, but because of the bound, it terminates before a representative subset of points for the left side is reached. The improved version does not fail to represent the entire curve with the limited number of points allowed by the bound.

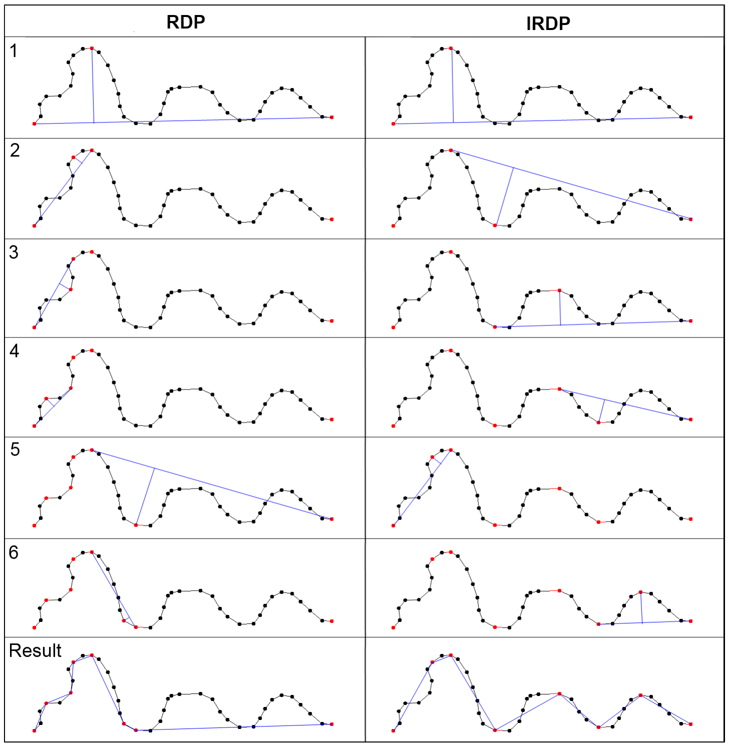


Figure 5.5. Results of both algorithms side by side showing a skewed result for the original RDP and a more distributed result in the IRDP implementation. This example has a bound of eight points.

Even if the bound constraint is removed, the IRDP algorithm results in a representation of the original curve that requires fewer points to obtain the same approximate quality as the original algorithm. This is due to IRDP's ability to find the largest polygonal chain within the curve to next represent. This ensures that in every step the next largest portion of the curve will have a point added to the approximation, thereby guaranteing that a good distribution of points will be used along the entire curve, and avoid clustering.

The only down side of IRDP is the small amount of complexity added to the algorithm. The original RDP can be described by a linear recurrence $T(n) = 2T(\frac{n}{2}) + O(n)$ which when solved is equal to $O(nlogn)$. However, the worst case complexity of RDP is $O(n^2)$. The latter run time is by far not the average case, and only occurs when the chosen threshold is so low that every point is included in the approximation.

IRDP is made up of a series of three $O(n)$ operations that are all repeated C times (see algorithm in definition 17), where C is the upper bound on the number of points needed in the approximation. This still reduces to $O(Cn)$ and by removing the constant IRDP is a $O(n)$ bounded algorithm.

5.4. Web Based Driving Directions

The most novel component of the new map-matching method is the use of a web based (DD) service to actually perform the conversion of raw gps data to map-matched data. Utilizing a web service generally requires concrete knowledge of the number of points being sent due to upper bounds on the number of points allowed in a single request. Most of the highly utilized web based mapping services provide programming API's with various limitations, mostly in relation to the number of requests allowed, and the number of points allowed per request. The IRDP algorithm fulfilled these limitations by representing routes within the prescribed limitation in points. The returned route is a standardized representation of the original GPS trajectory constructed using the minimum necessary set of points. Figure 5.6 shows an example of three separate routes, all connecting the same starting and stopping locations, map-matched to the same normalized path.

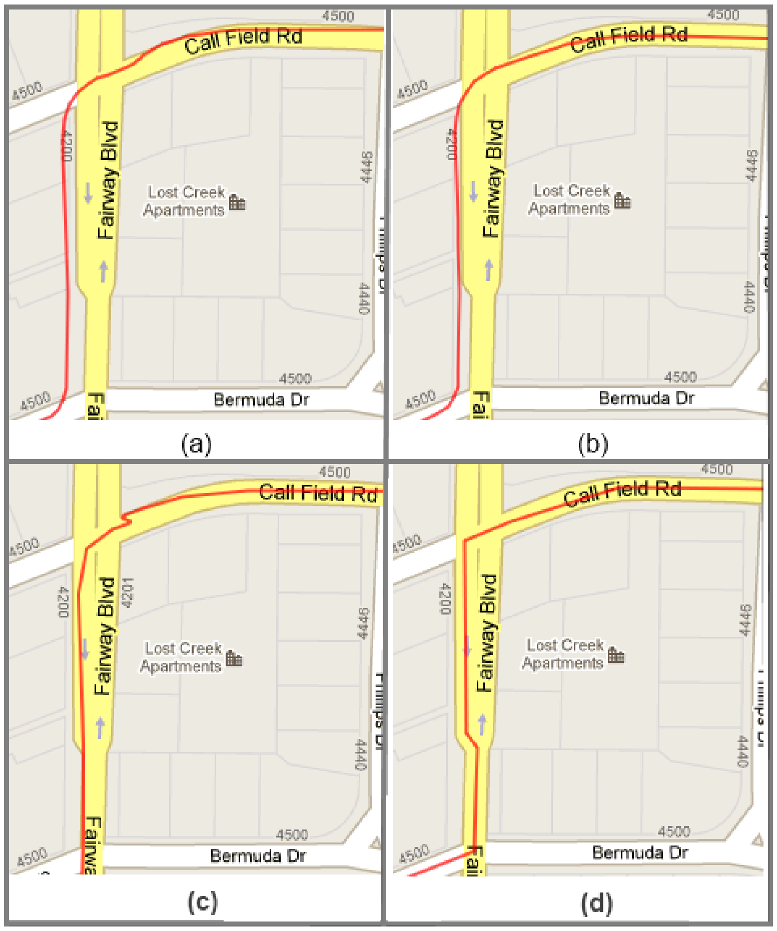


Figure 5.6. Matching routes (a,b,c) to the same normalized path (d).

This standardization method allows individual routes driven by a traveler between the same two destinations on different occasions to be accurately adjusted and portrayed as being the same series of logged points. The benefits of standardizing a user's route can be found when comparing two routes traveling between the same destination on separate occasions. When routes are standardized, the distance metric used for comparing routes will typically have a tighter upper bound, allowing a threshold to be set with more confidence and reduce the number of false positives and negatives.

Without an offline map-matching filtering step, routes that travel the same path on a road network will differ from one another due to the variation and error in the GPS produced data along with differences in a driver's trajectory. Map-matching provides a way to compare, categorize, and analyze similar and dissimilar routes by providing a consistent representation of a route.

Additionally to providing a consistent representation of a route, this map-matching algorithm gives as a means of significantly reducing the number of points representing a route. In figure 5.7, one route with all of its original data points plotted is shown next to the map-matched version that incorporates just the necessary points needed to accurately portray the route on the underlying road network. The original route had 340 data points (1 per second) plotted, and the reduced version has 21, for a reduction of 94%.

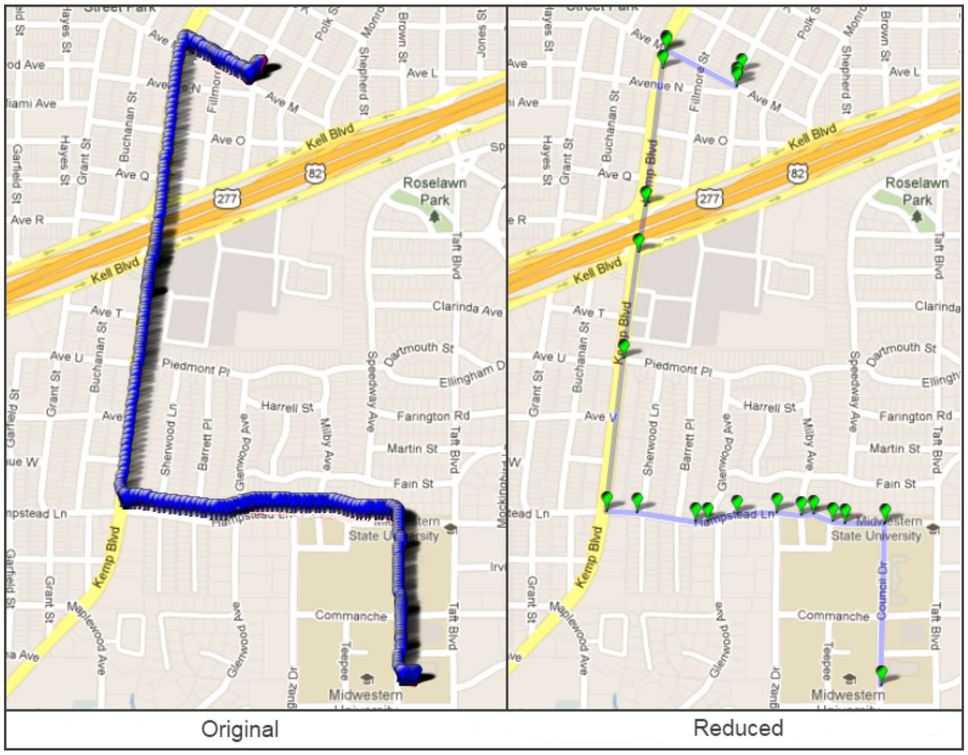


Figure 5.7. Route with set of all original points and route with the reduced interpolated set.

Given the identified *i-points* from step one of the algorithm (figure 5.2[A]), step two can now send these *i-points* to a DD service routing algorithm to obtain an initial matched route (figure 5.2[B]). It then inspects the matched route and classifies each *i-point*. Those (if any) that were probable in causing error are removed (figure 5.2[C]). The classification and removal of error prone points is discussed in the next section.

The DD service routing algorithm can be obtained by using the map based services provided by many popular web service providers including: MapQuest®, Yahoo®, and Google®. The DD services are built to interpolate points between a starting point A and stopping point B, regardless of distance apart. Each interpolated point has the benefit of being a map-matched, regardless of the values of A or B. The CaPPture system takes advantage of the fact that by providing any two points (V_i, V_j) to the DD service, all points necessary to provide adequate representation of the underlying road network connecting V_i and V_j, will be returned. This is always the minimum number of points needed to represent that specific section of the road network.

Figure 5.8 shows the four web services that were provided slightly different lat/lon pairs[5] representing A and B. Even though each web service returned the same normalized interpolated route, this is not normally the case.

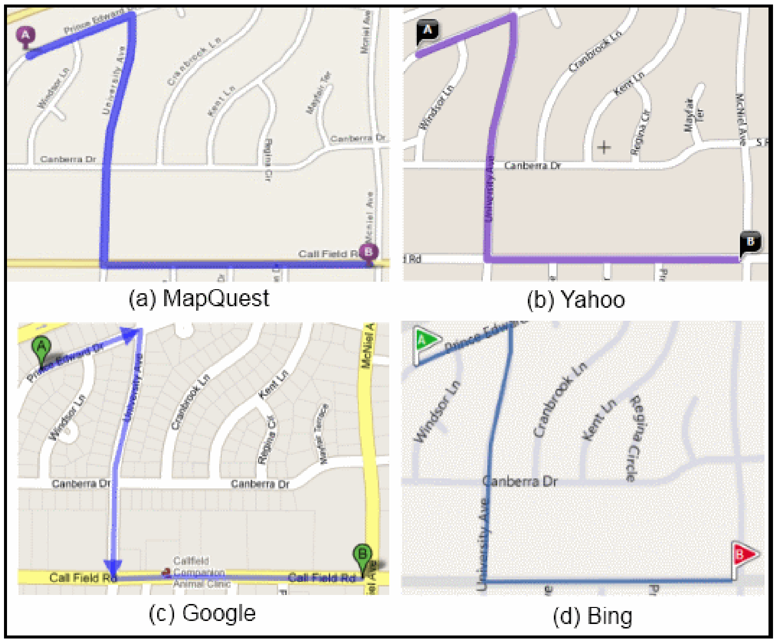


Figure 5.8. Results of DD services that returned the same route.

Each interpolated point between A and B will create a route connecting the start and stop of the route, and be perfectly aligned with the road network that the applicable DD has defined within its system.

[5]The differences fell within the same parameters that typical routes connecting the same starting and stopping destinations would fall into.

To send just the starting point A and the ending point B of a route that has only a single solution (as does the short example in figure 5.8) will guarantee the same result. However, by separating A and B by a distance that allows the DD service to make choices, it cannot be guaranteed that a route matching the original will be returned. Figure 5.9 shows a route that was incorrectly interpolated by a DD service because only the first and last points (A and B) were submitted, and alternative routes were available. Figure 5.9 portrays the original route in which the driver utilized smaller back-roads to get to their destination. The DD service, on the other hand, did not choose the same route basing its choice on proprietary routing algorithms.

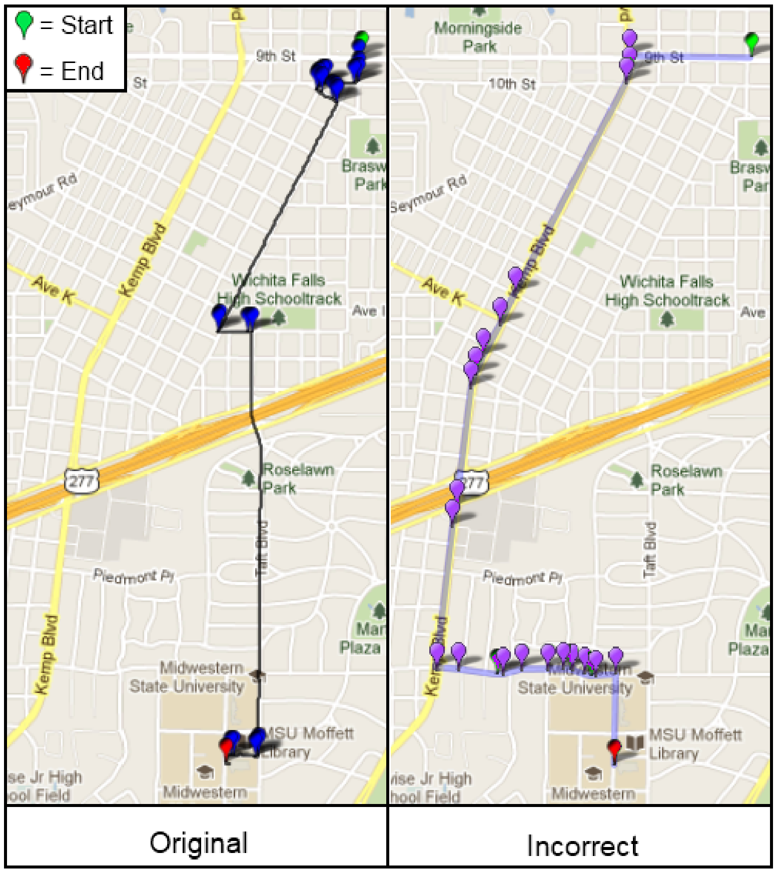


Figure 5.9. Results of DD service returning the wrong route given too much leeway.

The problem arises when the chosen web service is given enough leeway to interpolate every portion of the route indiscriminately relative to the actual route. Figure 5.10 shows two separate DD services that were given the same starting and ending destinations, yet they returned different routes. The decisions made by each DD service are based on the proprietary algorithms used by each service to calculate the “best” (least cost) route between A and B.

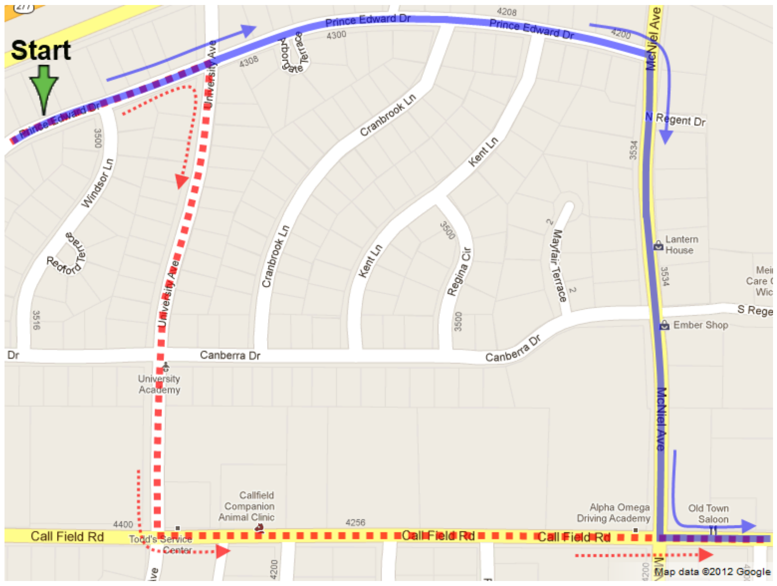


Figure 5.10. Two DD services can easily return different routes connecting the same starting and stopping points (Google Maps® is in red [dashed] and Bing® is in blue [solid]).

To calculate a best cost route, DD services implement variations of Dijkstra[24] and A*[42] shortest path algorithms. These algorithms take into account the connectivity (navigability) and impedance between A and B. Impedance is the cost of traveling across a navigable feature. Some examples of cost are: distance, speed-limits, turn-restrictions, and one-way streets. Each DD service calculates impedance using proprietary versions of the aforementioned algorithms assigning costs in different ways [29]. This gives the possibility that given the same starting and stopping points, the resulting routes can differ. This point is emphasized, because these path algorithms make decisions based on a set of cost parameters, where actual drivers tend to use personal heuristics not known to a mapping service. The results are that the “fastest” or “shortest” or “optimal” route returned by the mapping service will not always match the route chosen by a driver.

It was previously stated that supplying only a starting point A and an ending point B to a DD service will quite possibly result in a route that does not match the actual traveled path. Therefore it is necessary to supply the DD service with a subset of points to ensure the service will produce a correct match. However, it is also not necessary or even desirable to supply the DD service with the entire set of route points. Although DD services are able to correctly match single points to the road network in most instances, they do sometimes produce incorrect matches. Not supplying the entire set of points, minimizes the chance of an incorrect match. In addition, it also significantly reduces the amount of time needed to match the route.

5.5. Filter and Refine

The third component of the new map-matching algorithm (figure 5.2[C]) is a trained expert system to identify and remove problematic *i-points* to ensure accurate map-matching of routes. This filter-and-refine step is necessary due to the possible and somewhat common occurrence of a submitted *i-point* being incorrectly matched by the DD service. Even one mismatched *i-point* will cause the route to be incorrectly map matched. Point mismatches are usually due to, but not limited to, the point being located near an intersection or underpass/overpass (see figure 5.11). Because the point selection algorithm does not have local access to a road network representation, it is unable to foresee and therefore avoid these situations. The novel filter-and-refine algorithm achieves map-matching at an extremely accurate level.

Figure 5.11 shows a typical intersection in which many road segments converge. Part A of the figure shows a driver traveling west and turning north. The road segment that the driver used (shaded in horizontal lines) does not match directly with the GPS trace. Part B of the figure shows a zoomed in version with the same GPS trace seemingly swinging out beyond both available road segments. The original point (depicted by a light colored balloon with no letter), was matched to the closest road segment (red balloon with letter B), and should have been matched to the location depicted by the green balloon with the letter "G". Since the point was matched to the incorrect road segment, the driving service returned a route utilizing the incorrect segment, no matter the cost. The cost in this context means adding routing information (in addition to the original route) ensuring that the mismatched point is traveled through. Part C of the figure shows the additional "cost" added by the DD service ensuring that the mismatched point is utilized. In this case (emphasized by dashed line) the DD service had to add a dangerous u-turn to ensure it could return a path that traveled through the mismatched point. By developing a method of identifying mismatched points, a second submission to the DD service (with mismatched points removed) will return a route that is error free (as depicted in part D of the figure).

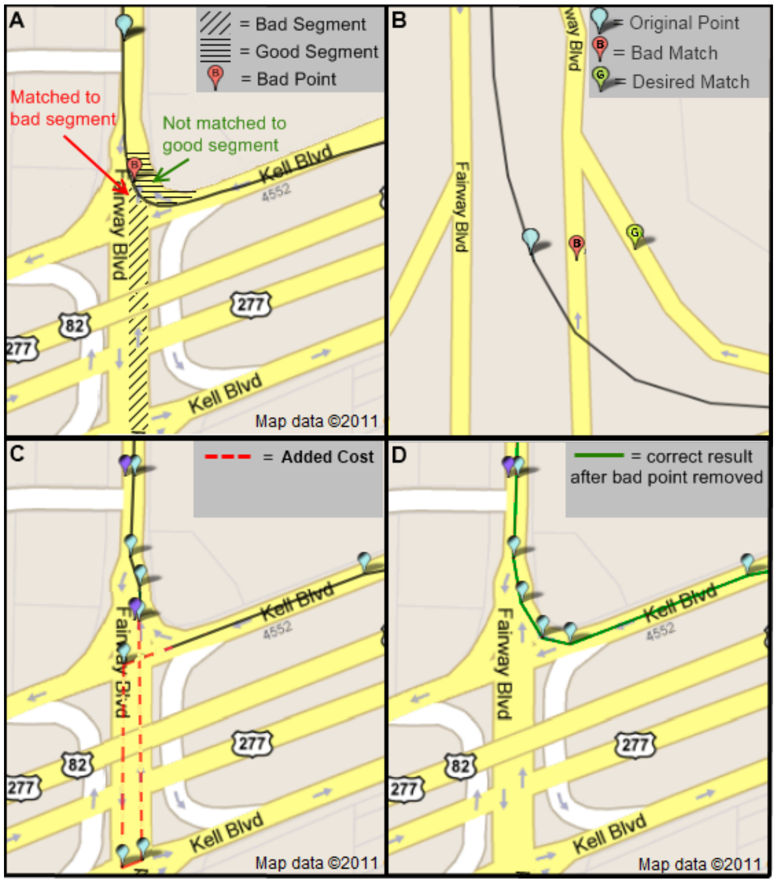


Figure 5.11. A problematic intersection forcing the need for point removal and re-submission to obtain a correctly map-matched route.

The problem with the DD service's initial map-matching attempt possibly containing incorrectly matched points was the motivation for a set of bad point filters to identify and remove any points causing the route to be incorrectly matched. These point removal filters were also used to evaluate other versions of point selection algorithms discussed in a subsequent section.

5.5.1. Bad Point Identification and Removal

In order to “force” the DD service to return a route that accurately follows the actual path traveled, the system also calculates and submits the inverse corner points of the route, in addition to the originally selected corner points. An inverse corner point is a point that lies midway along the path between two corner points. Submitting both the original corner points and the inverse corner points ensures the DD service is not given sufficient latitude to make incorrect assumptions. This reduces the probability of a removed point causing a DD service route error to near zero.

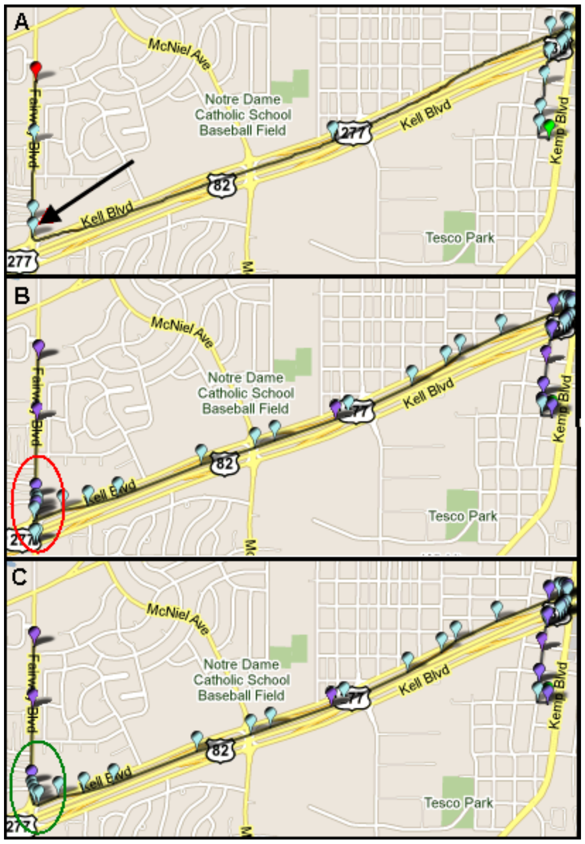


Figure 5.12. (A) Original route with points selected for submittal. Arrow shows bad point needing removal. (B) First result from the DD service before bad point removal. (C)Final correct Map-Matched result.

5.5.2. Identification/Removal of Points

After the Peucker/inverse corner point selection algorithm has been performed, the selected points are then submitted to the DD service and a map matched route is returned. However, the DD service may have mismatched some of the points submitted causing the returned route to be incorrect (discussed previously). Therefore it is necessary to identify the points that were mismatched in order to eliminate them. An example of point identification and removal can be seen in figure 5.12.

In order to identify the mismatched points, the C4.5[86] algorithm was used to develop a set of rules to be applied during the comparison of the original route to the route returned by the DD service. The tree was trained using a set of attributes: *DistPoint*, *DiffBearing*, *PathRatio*, and *DuplicatePoint* explained in the following sections [6].

DistPoint - The value of this attribute is obtained by finding the distance from A to B. The idea behind this attribute is that mismatched points could be identified by having a distance over some discovered threshold. Results showed that GPS error rendered this useless, because the point was always matched to the closest segment (correctly or incorrectly), and therefore no discoverable threshold could be identified.

DiffBearing - is the difference in bearing between A to B. Typically, a mismatched point would have a bearing completely different then the original point due to the DD service having to find an alternate path in order to keep the submitted point included in the route. In figure 5.13, it can be seen that the point along the original path (dotted line) was matched to the road segment to the south, when it should have been matched to the segment north of it, causing the bearing difference between A and B to be close to 180 degrees.

PathRatio - The single most important attribute when identifying mismatched points turned out to be the *PathRatio*. The path ratio is obtained by calculating the sum of the distances between each point from "Start" to "End" in the original path and in the returned path. If the ratio is over some threshold, it then implies that the path returned from the DD

[6]When calculating values for the attributes, any comparisons made between points always refers to a point, A, originally identified by the Peucker/inverse corner point selection algorithm, and B, the closest point (based on distance) returned by the DD service.

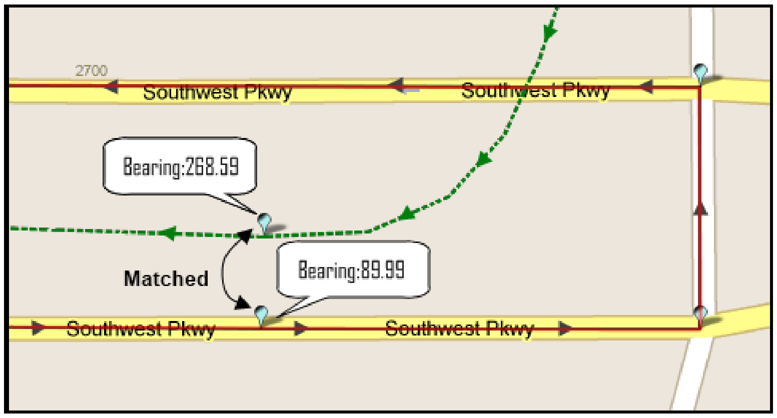


Figure 5.13. The original point (along the dotted path) when compared to its matched point (along the solid path) should have a similar bearing. If the bearing difference is over some threshold, it is most likely mismatched.

service deviated from the original path in some manner. An example of this can be seen in figure 5.14. The *PathRatio* difference was calculated for points before and after a submitted way point.

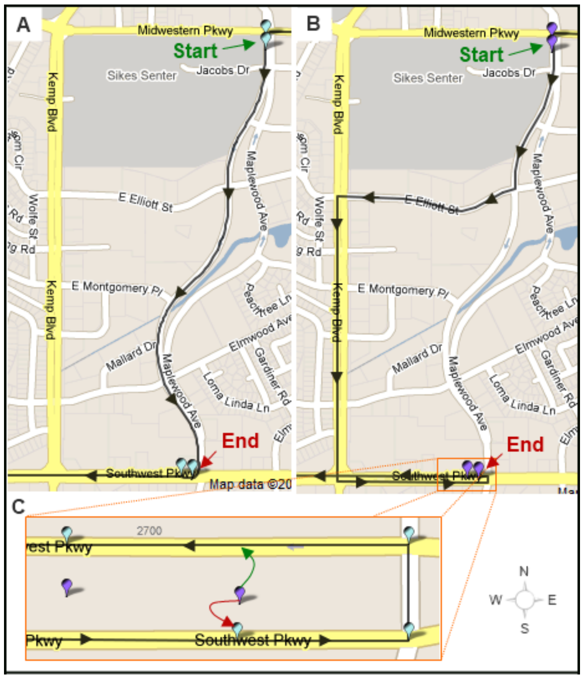


Figure 5.14. (A) Shows the original route with a path between "Start" and "End" and (B) shows an alternate route chosen by the DD service between the two. (C) exhibits a blown up section of the map showing why the DD service had to deviate from the original route to ensure it included the mismatched point.

DuplicatePoint - A duplicate point is returned from the DD service in instances when the service had to alter the route in order to keep a mismatched point within the returned route. In the context of this paper, a duplicate point can be defined as $\{< p_1, t_x > \ldots < p_1, t_y >\}$ where p is the same lat/lon, but the time difference between $< p_1, t_x >$ and $< p_1, t_y >$ is greater than 1. This means that the DD service returned a point that it needed to travel through twice to keep all the points submitted by the Peucker/inverse corner point selection algorithm included in the route. Any submitted point with a duplicate at $t-1$ or $t+1$ is also suspect of being mismatched.

Of the attributes listed above, the two most important were found to be *PathRatio* and *DiffBearing*. Either of these two values being over the discovered thresholds gave much confidence in eliminating the submitted point. Once the mismatched points have been identified, they are removed from the set of selected points. The revised set of selected points is then resubmitted to the DD service and a correctly map matched route is returned.

5.6. Experiment and Results

5.6.1. Gps Data

Experiments collected trace data by placing GPS loggers in the cars of ten volunteers. The loggers are the “Genie BGT-31” GPS data logger which use the SiRF Star III chip set and an optional SD card slot. Each logger was given to the subjects with a car charger and a 2GB SD card. The device was set to log raw NMEA data directly to the SD card and could hold approximately 50 days worth of GPS data. Five subjects collected the bulk of data which contained over 8.9 million individual data points with a time granularity of 1-2 second intervals. Each subject collected data an average of 34.3 days with a total number of 584 stops and 341 routes identified.

5.6.2. Raw GPS Segmentation

As a precursor to testing the map-matching technique, a significant amount of GPS traces were collected. Working with raw GPS traces is undesirable and clarifying the data is a crucial step, as mentioned in [17]. The majority of the data was collected in a 70 square mile geographical area surrounding Wichita Falls, Texas, a city with a population of just over 100,000. The raw GPS data was segmented by sorting it first chronologically and then identifying stops within the recorded points $(P_1, ..., P_n)$. The collected GPS traces were logged using "continuous" mode, which means that data points were being logged continuously regardless of whether the subject was moving or not.

To identify stops with data logged in "continuous" mode, one can not simply look for time gaps between P_i and P_j over some threshold. First, one must identify when a subject is not moving (change in time without change in location), and second determine if the stop in movement is a purposeful stop. Any stop over a threshold of 180 seconds was assumed as being purposeful. This threshold was not an objective choice, but a heuristical one.

After stops were identified, subsequently identifying the routes that connected each of the stops was possible. Analysis of the data identified 584 stops and 341 routes. To ensure that the identified features within the data were reliable, a significant portion of the data that did not fit into our expected parameters for stops or routes was not used. With the data that was usable, the average trip time was 11.17 minutes, the average trip distance was 4.38 miles, and the number of trips per day averaged 3.8. This fits into the expected parameters, as summarized in a 2001 national household travel survey[44].

5.6.3. Results

The experiment utilized a framework written by the authors that identified key features within GPS streams for individual users. The resulting data consisted of a total of 584 stops and 341 routes. Out of the 341 routes, 200 were selected for analysis due to there proximity within the test city and the availability of the subjects which allowed visual verification from the original drivers. Out of the 200 routes analyzed, we had 100% accuracy with the two stage submission process of our algorithm, and achieved a pristine standardized set of driven routes. Although the experimental step seems simple (visual confirmation), a considerable amount of time was spent inspecting possible problematic routes with features such as turn-arounds and overpasses to ensure the algorithm correctly processes difficult routes.

CHAPTER 6

LABELING AND PREDICTION

6.1. Background

6.1.1. Decision Trees

ID3 and C4.5 are algorithms introduced by J. Ross Quinlan to create classification models, also called *decision trees* 6.1. One drawback of the original ID3 algorithm was its inability to classify attributes with continuous values. For example in figure 6.1 you can see that the attribute “outlook” has three possible discrete values: Sunny, Overcast, and Rain. The problem is not all attributes conveniently have non-continuous values. The C4.5 model takes care of this restriction and allows each attribute to have (possibly) a continuous range of values (e.g. temperature or time).

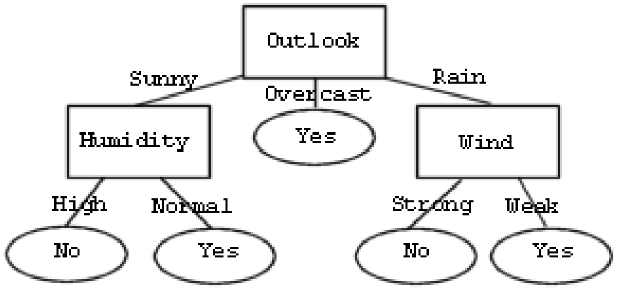


Figure 6.1. Decision tree (choosing to golf).

6.1.2. Dbscan

Density based clustering uses the concepts of *neighborhood* and *density* to determine what constitutes a cluster [30]. There are two parameters used to define these concepts: *Eps* and *MinPts*. *Eps* determines the area of a neighborhood (radius) and *MinPts* determines the minimum number of points that must be present in that radius to deem it a cluster. Density based clustering also uses two concepts: *density-connectivity* and *density-reachability* which allow a cluster to *grow* if nearby points are close enough to current *core* members of the cluster. *Core* members are points that are not on the periphery of a cluster. *Growing* is well suited for finding clusters that have odd shape (as can be seen in figure 6.2).

This is one reason we chose a density based clustering scheme. Typically things that happen in real life do not fit in boxes and circles. Dbscan's ability to find arbitrarily shaped clusters is also what makes it a more optimal choice over algorithms such as the K-means clustering approach, which favors symmetric shaped clusters [40]. As can be seen in figure 6.3, a persons' parking habits might not fit into a simple easily determined pattern.

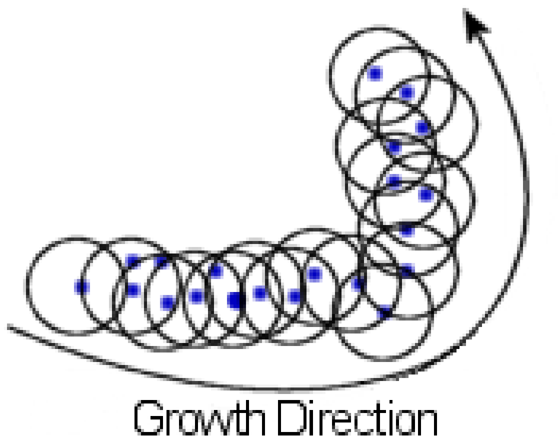


Figure 6.2. Density reachability and connectivity.

Clusters typically have a much higher concentration of points than the surrounding area. The sparse areas not included in a cluster are considered noise. Dbscan fits well in this research for the fact that areas that are visited frequently, become clusters. Areas that are visited infrequently are considered noise and ignored if the *MinPts* is set appropriately. Obviously if *MinPts* is set to 1, everything becomes it's own cluster. K-means has a tendency to pull these outliers in, forcing them into one of the K clusters [40]. In our framework, we use DBSCAN to find clusters of geospatial points and call these individual collections: *points-of-interest* (*POI*). We then aggregate the temporal values of each member in a *POI* to obtain additional information over and above the spatial attributes given with latitude/longitude pairs. This is explained further in 6.2.2.

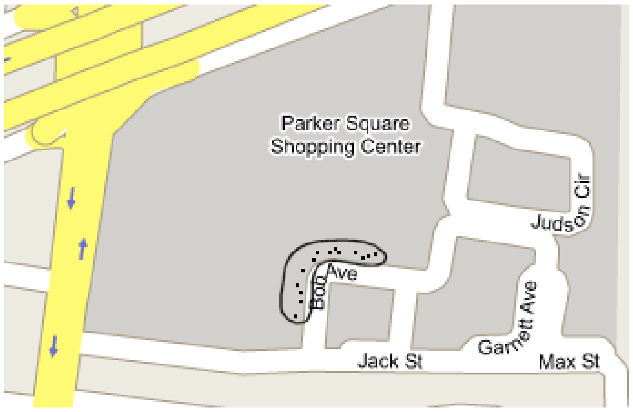


Figure 6.3. Non-typical cluster.

6.2. Trip Purpose Detection

6.2.1. Data Preparation

Finding POI's and classifying them required a couple of preliminary steps. The first step was to compute trip stop lengths from given raw GPS data. This has to be done based on the type of logged data in the set. Typically GPS logs come in two categories: "continuous" and "movement only". If the GPS unit placed an entry in the log only when the unit was moving, then the trip stop length could be computed by the difference between the two timestamps (the one before the stop and the one after the stop). If however, the unit logged data with set time intervals regardless of movement or not, then the trip stop length would have to be computed based on the difference of the first and last records of contiguous entries with a velocity of 0.

Trip lengths in this study were computed using both techniques mentioned above. The logged GPS data can be described as a collection of points $(P_1, P_2 ... P_n)$. Each point is defined by a Latitude (Lat) and Longitude (Lon) pair, accompanied by the Time of Day (ToD). The entire set becomes:

$$(P[Lat, Lon, ToD]_1, P[Lat, Lon, ToD]_2, ..., P[Lat, Lon, ToD]_n)$$

Taken alone, each point in the data set gives very little information about the trip or the purpose of the trip. In fact, any single point taken alone is somewhat useless. For example if point P_i has a latitude of x, a longitude of y, and a ToD of z. The values x, y, z together do not give much information. If, though, $P_{i+1}.z$ has a ToD of $P_i.z + 3960$, we can infer that the individual remained at P_i for 66 minutes. Then if the individual continued visiting that same (approximate) location: (x, y), and staying for roughly 60 minutes, it is possible to infer with some certainty what type of location this was, and possibly the purpose of the visit. Of course, other factors are involved, such as exactly when during the day is the location visited.

6.2.2. Data Aggregation

Ultimately for a single point to become meaningful, it must be related to, or grouped with other similar points. As briefly mentioned in 6.1.2, by clustering these like points' together *POI's* can be discovered. Once a *POI* is found, then the value of each individual point within the *POI* can be aggregated with the other members to obtain very specific information about a spatial location. Some typical aggregate values are: *Time of Day, Length of Stay, Earliest Arrival Time, etc..*

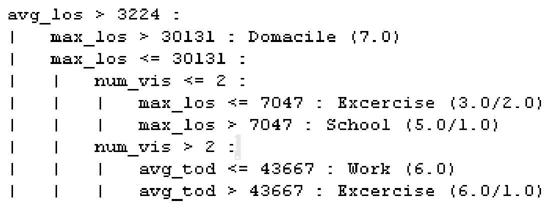
When using a decision tree, each node in the tree is assigned an attribute that best splits the remaining data set. The aggregate values from a training set of *POI's* become these attributes and are used to build the decision tree. After the tree is trained, the remaining *POI's* aggregate values are fed into the tree allowing each to be classified.

Finding reliable values from *POI's* is indispensable because these values are not only train the decision tree, but to also obtain classification for the remaining *POI's.* Therefore, having a reliable method to find areas of interest is imperative if accurate aggregate data is to be found.

6.2.3. Classifying Clusters

Using DBSCAN to identify the *POI's* and subsequently the aggregate values from each, we can then input these into the C4.5 algorithm. To create a decision tree, C4.5

requires a class file and a data file. The class file contains two portions: 1) The training parameters, some of which were mentioned in the previous section, and a list of all possible classifications. 2) The data file, which contains the actual training data used to populate the decision tree. A portion of an example tree can be seen in figure 6.4.



```
avg_los > 3224 :
|   max_los > 30131 : Domacile (7.0)
|   max_los <= 30131 :
|   |   num_vis <= 2 :
|   |   |   max_los <= 7047 : Excercise (3.0/2.0)
|   |   |   max_los > 7047 : School (5.0/1.0)
|   |   num_vis > 2 :
|   |   |   avg_tod <= 43667 : Work (6.0)
|   |   |   avg_tod > 43667 : Excercise (6.0/1.0)
```

Figure 6.4. Portion of Decision Tree.

The values on the left hand side of the operator are the attribute labels of the aggregate data values (from the class file). The values on the right hand side of the operator are the aggregate values obtained from the data file. Finally, the values on the right hand side of the colon are classification labels. To obtain a classification for a cluster (*POI*), you would simply traverse the tree using the aggregate values (from the cluster). At each level of the tree, the appropriate value would be compared and a decision on which direction to go would be made. C4.5 attempts to simplify the tree when possible. It is conceivable that only a small subset of the original attributes are needed to obtain a classification for a *POI*. Figure 6.5 shows an example of two classified points of interest.

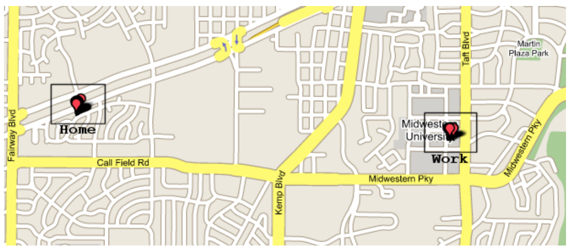


Figure 6.5. Identified clusters.

6.3. Spatial Clustering

As each cluster was classified it was assigned a level of correctness which represented the percentage of correctly classified points within the cluster. The upper two graphs in

Figure 3 summarize the results for spatial clustering only. The left graph represents the results of the experiment for the randomly generated data and the right graph represents the results for the actual GPS data set. The Y-values indicate level of correctness and the X-values indicate spatial cluster size. Clearly the classification framework performed significantly better and more consistently on the randomly generated than on the collected data. This can likely be attributed to at least two factors. One is the difference in size of the data sets, the collected data set being smaller. The second is the cleanliness of the random data set, having little noise. The applicability of the framework to different data set sizes is certainly one area for future research. Both sets of data demonstrated the desirability of larger values for the minimum cluster size. Results were consistently better in both as the minimum number of points increased. Classification of the random data consistently performed between 70% and 97% correctness for all values. Classification of the actual data performed well for cluster sizes of between 25 and 50, but less well at other sizes. It is likely that data set size was a significant factor in the performance.

6.4. Temporal Clustering

This experiment included the first attempt at extending the spatial clustering by reprocessing each cluster temporally. That is, Dbscan was used to process each cluster of the collected data to find sub-clusters based on the time-based values of length or stay and time of day. The results of the experiment are provided in the lower two graphs of Figure 3. The X-values indicate the length of stay and the time of day in seconds, respectively. Clearly, the inclusion of temporal clustering to the classification process improved the performance significantly. The classification percentages were at 80% and above for all values of the minimum number of points. The improved performance, despite the smaller data set size and increased noise, offers promise that the inclusion of time-based factors can significantly improve the performance of the proposed framework.

6.5. Experiment Design and Results

6.5.1. Random Data

Generating Pseudo random trips that somewhat resembled typical behavior, have to be generated within a predefined set of parameters. Trips that are truly random would in no way mimic an individual's behavior. So, trips are simulated according to the following assumptions: 1. All trips are based on a map of an area that had trip stops pre-defined spatially and categorically. 2. Each possible trip stop must have a Minimum Bounding Rectangle (MBR) defined using two Lat/Lon pairs, along with a classification label (work, home, restaurant, etc.). 3. The batch of trips must be generated for a specified number of days, typically 90-120. 4. Each trip is decided without any knowledge of the current location or where the individual has been. This was only to help simplify the trip generation algorithm. Making a decision on where to go based on current location and previously visited locations increased the complexity well beyond what was needed. 5. Trip stops where generated based on a user model. This model was a detailed description of what a user typically did in a week's time. A user was simply a volunteer whom agreed to fill out a detailed questionnaire giving the information needed to define the model. The information could then be used as a guide when generating the random trips.

Another key element in the trip generation process, was having the ability to raise the probability of going on a trip at a specified time. Using a formula specifically created to accurately increase the probability of triggering a trip within a specified time window allowed accurate daily activity logs to be created. The derived log generation formula follows:

$$f(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

where x is the current time of day, μ is a specified time for a location in which the probability of going there should be high, and σ is a time window (standard deviation) around μ.

For example: if an individual goes to work at 8:00 every morning, then the μ for that workplace would be 8. If the individual rarely deviated from this, then σ might be defined as .25, which would create a window of high probability between 7:45 and 8:15 that this individual would go to work. If this time varied between 8 and 10, then the μ could be 9 with an σ of 1 hour.

In order to represent the probability in a more realistic way, the previous function was modified to accept an extra parameter d, where $d = (d_1, d_2) \mid d\epsilon\{(0,1),(-1,0),(-1,1)\}$ such that the final probability becomes:

$$p(x) = \begin{cases} f(x) & for\ \mu + d_1\sigma \leq x \leq \mu + d_2\sigma \\ 0 & \text{for all other } x \end{cases} \tag{1}$$

This was done to allow a user model to be more specific on their typical behavior. For example if a user goes to work at 7:30 every morning, but never before, d could be set to (0,1). This way when applying σ to μ, the window for travel to work would be 7:30 + σ.

6.5.2. Results

Random Data. To test the accuracy of the classification technique, 50 generations of random data were created. For each generation, the clustering algorithm modified two variables to find which offered the best chance for correct classification. The two variables were distance (*Eps*) and density (*MinPts*). Distance is defined as a cluster size in square feet, and density is simply the number of points within the cluster. The starting cluster size was roughly 26 x 26 feet, or 1/200 * 1 mile. For this size, all clusters of density 1 through density 15 were found, aggregated, and classified. This process was repeated for sizes up to 1/20th of a mile (10 distinct cluster sizes).

As each cluster was found, it was classified using a classification tree based on the data generated for that test. Each cluster was assigned a level of correctness, which is simply a percentage of correctly classified points within the cluster. A value of 1 implies all points within the cluster are correctly classified, and a value of 0 implies no points are correctly classified.

Figure 6.6 shows the results for each cluster size. The Y-values indicate level of correctness, and the X-values indicate cluster density. It is obvious there are cluster sizes that perform significantly better in the classification process.

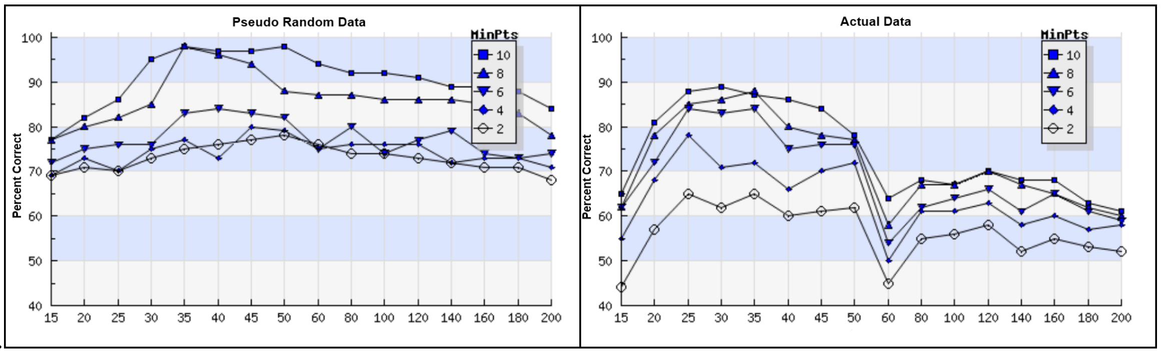


Figure 6.6. Clustering locations based on a spatial eps for random and actual data.

Actual Data. Figure 6.6 shows the results of the actual data alongside the results of the randomly generated data. Again, the x values indicate cluster size in feet, and the y values indicated correctness. It is obvious there are cluster sizes that perform significantly better in the classification process. The smallest cluster size classified all points in the 90th percentile for density over 7.

The remainder of the cluster sizes performed somewhat similarly. Each cluster peaked in correctness around a density of 5, before making a sharp decline and then slowly rising again. This decline can be explained by the effects of outlying values. If a point is added to the cluster and its trip purpose is not similar to those previously in the cluster, it skews the aggregate data. As more similar points are added, the common values among them average in to the classification and slowly increase the correctness.

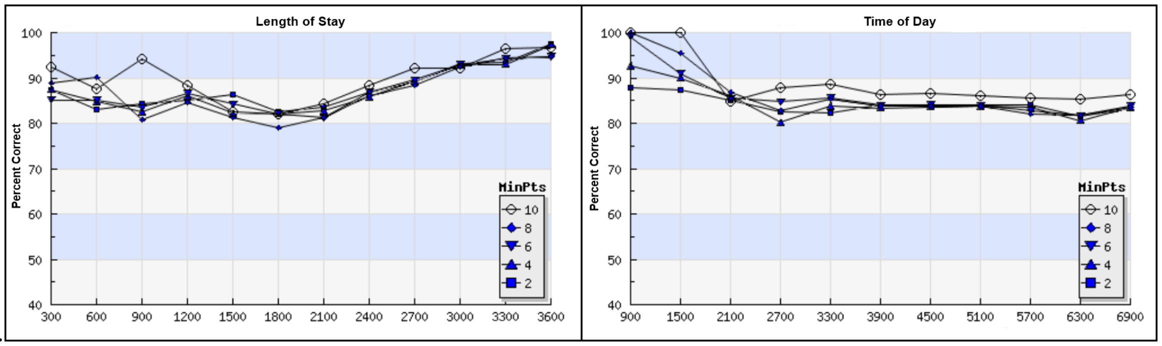


Figure 6.7. Clustering locations based on a temporal eps for random and actual data.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This study set out to answer the research question: "Can the daily activities of a given individual be labeled and/or predicted?" Part of the methodology used to answer this question required the collection of historical GPS data for many individuals. The data was necessary to train and test various labeling and prediction algorithms. However, the initial collection of data was found to be problematic, and the analysis of the data showed that the quality did not reach acceptable levels. This required the addition of tasks within the methodology in order to improve and streamline the collection and processing of collected data. Those additional tasks resulted in the bulk of this study and lead to an extensive and rooted understanding of GPS and GIS concepts.

7.1. Conclusions

Collection - The data collection effort was initially assumed trivial. It was a tool to gather enough information about user activity to then create robust and accurate models that would predict the behavior or location of individuals. What ultimately happened is that the data collection process became an entity in its own right due to the complexities introduced by signal loss. To assure an accurate and clean data set, numerous filtering and signal loss identification algorithms were either incorporated or created. This resulted in three generations of data collection software culminating in a very large and near error free historical GPS database that can be used in the creation of prediction or labeling algorithms.

Map-Matching - As the collection process continued, the size of the GPS database grew into the millions of records and introduced a need to reduce the overall size of the database in order to speed up queries performed on the data. It also made way for an opportunity to improve the overall quality of the stored GPS points. The novel offline map-matching algorithm developed in this study succeeded in reducing the number of points needed to represent a route by up to 94%. In addition to the overwhelming reduction in points, the individually map-matched points made the comparison of routes travelled by the same subject very accurately differentiated. The creation of a successful and accurate offline map-matching algorithm is the second preliminary step toward this studies goal of the creation of prediction algorithms.

Labeling And Prediction - The final goal of this study was to accurately classify a users POI. This study showed a method for labeling visited locations by sending aggregate data from clusters through a decision tree created by the c4.5 algorithm. This method was up to 90% accurate depending on cluster sizes. The downside is that the method used to create the decision tree is dated. There are many viable classification alternatives that can be implemented for improved accuracy and robustness. Some of these alternatives are discussed in the next section dealing with future work.

7.2. Future Work

In a publication titled "Ubiquitous Advertising: The Killer Application for the 21st Century" [55], John Krumm lays out the inevitability of targeted advertising based on such parameters as location and time of day. The article goes on to point out that individuals won't even pay a few cents to avoid advertisements in downloaded apps. Many points in this and other articles show that the need for location aware and location based services is still relatively young. In another recent publication, Jiawei Han states that: "[c]omputing with spatial trajectories is still a fairly young and dynamic research field." [116]." Devices are becoming more ubiquitous and powerful, connection speeds are steadily increasing and more available, and the adoption of location aware software is steadily increasing. These trends corroborate the need for improved location aware and location based services.

To fulfill the need for accurate and improved location aware software, the prediction portion of this study must be amplified and extended to incorporate other prediction and classification algorithms with which to increase the effectiveness and accuracy of predictions. Hidden Markov Models (HMM), a type of dynamic Bayesian network, are currently used in map-matching and route prediction and show mild success. This is one area in which the prediction portion of this study could be extended into.

Another area that shows promise in the creation of prediction models is Symbolic Regression (SR), also known as Genetic Programming (GP). This is the process of finding a function that fits (models) a set of input data. For example, given a set of y and X values, where X is one or more dependent variables, and y is the independent variable. If an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y. Cornell University has created a successful software tool called "Eureqa" that has shown success in detecting equations and hidden mathematical relationships in large and complex sets of data. A recent literature review shows that this has not currently been used as a basis for predicting a users route or location. Therefore, this is the next area to be explored with high expectations of achieving this studies goal in creating a robust and accurate location prediction or activity labeling algorithm.

APPENDIX A

GPS FORMATS

Table A.1. List of acceptable GPS formats

File Format	**WayPoints**		**Tracks**		**Routes**	
	Read	Write	Read	Write	Read	Write
Alan Map500 tracklogs (.trl)			yes	yes		
Alan Map500 waypoints and routes (.wpr)	yes	yes			yes	yes
Brauniger IQ Series Barograph Download			yes			
Bushnell GPS Trail file (.trl)			yes	yes		
Cambridge/Winpilot glider software	yes	yes				
CarteSurTable data file (.cst)	yes		yes		yes	
Cetus for Palm/OS	yes	yes	yes			
CoastalExplorer XML	yes	yes			yes	yes
Columbus/Visiontac V900 files (.csv)	yes		yes			
Comma separated values	yes	yes				
CompeGPS data files (.wpt/.trk/.rte)	yes	yes	yes	yes	yes	yes
CoPilot Flight Planner for Palm/OS	yes	yes				
cotoGPS for Palm/OS	yes	yes	yes			
Data Logger iBlue747 csv			yes	yes		
Dell Axim Navigation System (.gpb)						
DeLorme .an1 (drawing) file	yes	yes		yes	yes	yes
DeLorme GPL			yes	yes		
DeLorme PN-20/PN-30/PN-40 USB protocol	yes	yes	yes	yes	yes	yes
DeLorme Street Atlas Plus	yes	yes				
DeLorme Street Atlas Route			yes			
DeLorme XMap HH Native .WPT	yes	yes				
DeLorme XMap/SAHH 2006 Native .TXT	yes	yes				
DeLorme XMat HH Street Atlas USA .WPT (PPC)	yes	yes				
Destinator Itineraries (.dat)					yes	yes
Destinator Points of Interest (.dat)	yes	yes				
Destinator TrackLogs (.dat)			yes	yes		
EasyGPS binary format	yes	yes				
Embedded Exif-GPS data (.jpg)	yes	yes				
Enigma binary waypoint file (.ert)	yes	yes			yes	yes
FAI/IGC Flight Recorder Data Format			yes	yes	yes	yes
Franson GPSGate Simulation		yes		yes		yes
Fugawi	yes	yes				
G7ToWin data files (.g7t)	yes		yes		yes	
Garmin 301 Custom position and heartrate	yes	yes				
Garmin Logbook XML			yes	yes		
Garmin MapSource - gdb	yes	yes	yes	yes	yes	yes
Garmin MapSource - mps	yes	yes	yes	yes	yes	yes
Garmin MapSource - txt (tab delimited)	yes	yes	yes	yes	yes	yes
Garmin PCX5	yes	yes	yes	yes	yes	yes

Garmin POI database	yes	yes				
Garmin Points of Interest (.gpi)	yes	yes				
Garmin serial/USB protocol	yes	yes	yes	yes	yes	yes
Garmin Training Center (.tcx)	yes		yes	yes		
Geocaching.com .loc	yes	yes				
GeocachingDB for Palm/OS	yes	yes				
Geogrid-Viewer ascii overlay file (.ovl)	yes	yes	yes	yes	yes	yes
Geogrid-Viewer tracklogs (.log)			yes	yes		
GEOnet Names Server (GNS)	yes	yes				
GeoNiche .pdb	yes	yes				
GlobalSat DG-100/BT-335 Download			yes			
Google Earth (kml)	yes	yes	yes	yes	yes	yes
Google Maps XML			yes			
Google Navigator Tracklines (.trl)			yes	yes		
GoPal GPS track log (.trk)			yes	yes		
GpilotS	yes	yes				
GPS TrackMaker	yes	yes	yes	yes	yes	yes
GPSBabel arc filter file	yes	yes				
GPSdrive Format	yes	yes				
GpsDrive Format for Tracks	yes	yes				
GPSman	yes	yes				
GPSPilot Tracker for Palm/OS	yes	yes				
gpsutil	yes	yes				
GPX XML	yes	yes	yes	yes	yes	yes
HikeTech	yes	yes	yes	yes		
Holux (gm-100) .wpo Format	yes	yes				
Holux M-241 (MTK based) Binary File Format			yes			
Holux M-241 (MTK based) download			yes			
HTML Output		yes				
html						
Humminbird tracks (.ht)	yes		yes	yes	yes	
Humminbird waypoints and routes (.hwr)	yes	yes	yes		yes	yes
IGN Rando track files			yes	yes		
iGO2008 points of interest (.upoi)	yes	yes				
IGO8 .trk			yes	yes		
Jelbert GeoTagger data file			yes	yes		
Jogmap.de XML format			yes			
Kartex 5 Track File			yes	yes		
Kartex 5 Waypoint File	yes	yes				
Kompass (DAV) Track (.tk)			yes	yes		
Kompass (DAV) Waypoints (.wp)	yes	yes				
KuDaTa PsiTrex text	yes	yes	yes	yes	yes	yes
Lowrance USR	yes	yes	yes	yes	yes	yes
Magellan Explorist Geocaching	yes	yes				
Magellan Mapsend	yes	yes	yes	yes	yes	yes
Magellan NAV Companion for Palm/OS	yes	yes				
Magellan SD files (as for eXplorist)	yes	yes	yes	yes	yes	yes

Magellan SD files (as for Meridian)	yes	yes	yes	yes	yes	yes
Magellan serial protocol	yes	yes	yes	yes	yes	yes
MagicMaps IK3D project file (.ikt)	yes		yes			
Map&Guide 'TourExchangeFormat' XML					yes	
Map&Guide to Palm/OS exported files (.pdb)	yes				yes	
MapAsia track file (.tr7)			yes	yes		
Mapopolis.com Mapconverter CSV	yes	yes				
MapTech Exchange Format	yes	yes				
Memory-Map Navigator overlay files (.mmo)	yes	yes	yes	yes	yes	yes
Microsoft AutoRoute 2002 (pin/route reader)					yes	
Microsoft Streets and Trips					yes	
Microsoft Streets and Trips 2002-2007	yes	yes				
Motorrad Routenplaner (Map&Guide) .bcr					yes	yes
MS PocketStreets 2002 Pushpin	yes	yes				
MTK Logger (iBlue 747,...) Binary File			yes			
MTK Logger (iBlue 747,Qstarz BT-1000,...)			yes			
National Geographic Topo .tpg (waypoints)	yes	yes				
National Geographic Topo 2.x .tpo			yes			
National Geographic Topo 3.x/4.x .tpo	yes		yes		yes	
Navicache.com XML	yes					
Navigon Mobile Navigator .rte files					yes	yes
Navigon Waypoints	yes	yes				
NaviGPS GT-11/BGT-11 Download	yes	yes	yes	yes	yes	yes
NaviGPS GT-31/BGT-31 datalogger (.sbp)			yes			
NaviGPS GT-31/BGT-31 SiRF binary logfile			yes			
Naviguide binary route file (.twl)	yes	yes				
Navitel binary track (.bin)			yes	yes		
Navitrak DNA marker format	yes	yes				
NetStumbler Summary File (text)	yes					
NIMA/GNIS Geographic Names File	yes	yes				
NMEA 0183 sentences	yes	yes	yes	yes		
Nokia Landmark Exchange	yes	yes				
OpenStreetMap data files	yes	yes		yes	yes	yes
OziExplorer	yes	yes	yes	yes	yes	yes
PalmDoc Output		yes				
PathAway Database for Palm/OS	yes	yes	yes	yes	yes	yes
PocketFMS breadcrumbs			yes	yes		
PocketFMS flightplan (.xml)	yes				yes	
PocketFMS waypoints (.txt)	yes	yes				
Quovadis	yes	yes				
Raymarine Waypoint File (.rwf)	yes	yes			yes	yes
Ricoh GPS Log File			yes	yes		
See You flight analysis data	yes	yes				
Skymap / KMD150 ascii files	yes	yes	yes	yes	yes	yes
SkyTraq Venus based loggers (download)	yes		yes			
SkyTraq Venus based loggers Binary File	yes		yes			
Sportsim track files (part of zipped .ssz			yes	yes		

Suunto Trek Manager (STM) .sdf files			yes	yes	yes	yes
Suunto Trek Manager (STM) WaypointPlus files	yes	yes	yes	yes	yes	yes
Swiss Map 25/50/100 (.xol)	yes	yes	yes	yes		
Tab delimited fields useful for OpenOffice	yes	yes				
Teletype [Get Jonathon Johnson to describe	yes	yes				
Textual Output		yes				
TomTom Itineraries (.itn)					yes	yes
TomTom Places Itineraries (.itn)					yes	yes
TomTom POI file (.asc)	yes	yes				
TomTom POI file (.ov2)	yes	yes				
TopoMapPro Places File	yes	yes				
TrackLogs digital mapping (.trl)	yes	yes	yes	yes		
U.S. Census Bureau Tiger Mapping Service	yes	yes				
Universal csv with field structure in first line	yes	yes	yes	yes	yes	yes
Vcard Output (for iPod)		yes				
VidaOne GPS for Pocket PC (.gpb)			yes	yes		
Vito Navigator II tracks	yes	yes	yes	yes	yes	yes
Vito SmartMap tracks (.vtt)			yes			
WiFiFoFum 2.0 for PocketPC XML	yes					
Wintec TES file	yes		yes			
Wintec WBT-100/200 Binary File Format			yes			
Wintec WBT-100/200 GPS Download	yes		yes			
Wintec WBT-201/G-Rays 2 Binary File Format			yes			
XAiOX iTrackU Logger	yes		yes			
XAiOX iTrackU Logger Binary File Format	yes	yes	yes	yes		
Yahoo Geocode API data	yes					

APPENDIX B

SUMMARY OF COLLECTED DATA

Table B.1. A summary of all collected data to date.

User	Gen	Total Pts	Filtered Pts	% Filtered	Dest	Routes	Stays
0	1	634,331	329,852	52%	5	31	231
1	1	109,357	78,737	72%	12	40	63
2	1	262,431	125,967	48%	9	27	113
3	1	218,409	133,229	61%	9	35	93
4	1	188,789	126,489	67%	4	30	89
5	1	131,359	85,383	65%	7	39	60
6	1	380,914	251,403	66%	9	34	277
7	1	112,227	63,969	57%	8	47	58
8	1	321,311	154,229	48%	5	38	185
9	1	639,277	415,530	65%	6	21	457
10	2	262,194	110,121	42%	9	57	110
11	2	46,393	25,980	56%	8	38	29
12	2	28,751	11,788	41%	4	19	11
13	2	136,447	72,317	53%	9	27	65
14	2	481,291	226,207	47%	11	28	158
15	2	70,954	31,929	45%	11	37	29
16	2	218,443	91,746	42%	7	31	110
17	2	167,976	87,347	52%	10	43	79
18	2	293,401	164,305	56%	4	13	197
19	2	479,737	249,463	52%	4	33	274
20	2	134,132	73,773	55%	7	37	89
21	2	301,309	105,458	35%	4	21	127
22	2	646,877	239,344	37%	9	38	287
23	2	275,123	145,815	53%	9	22	117
24	3	357,085	160,688	45%	5	32	161
25	3	229,089	80,181	35%	4	22	80
26	3	438,330	245,465	56%	6	45	221
27	3	128,926	39,967	31%	9	23	44
28	3	349,477	118,822	34%	6	44	83
29	3	463,764	148,404	32%	7	56	134
30	3	49,819	28,895	58%	5	26	20
31	3	169,415	42,354	25%	5	26	30
32	3	69,844	28,636	41%	5	13	23
33	3	417,339	146,069	35%	7	40	102
34	3	15,014	8,257	55%	9	43	7
35	3	560,496	190,569	34%	6	15	229
36	3	162,904	43,984	27%	5	15	53
37	3	455,343	141,156	31%	9	28	141
		10,408,275	**4,823,830**	**48%**	**268**	**1340**	**8708**

BIBLIOGRAPHY

[1] *Responsible conduct in data management*, Office of Research Integrity, March 2005, [online] urlhttp://www.niu.edu/rcrportal/datamanagement/dmglossary.html.

[2] Berg Insight AB, *Gps and mobile handsets 4th edition*, Tech. report, March 2010.

[3] B. Adams, D. Phung, and S. Venkatesh, *Extraction of social context and application to personal multimedia exploration*, Proceedings of the 14th annual ACM International Conference on Multimedia, ACM, 2006, pp. 987–996.

[4] H. Alt, A. Efrat, G. Rote, and C. Wenk, *Matching planar maps*, Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2003, pp. 589–598.

[5] M. Ankerst, M.M. Breunig, H.P. Kriegel, and J. Sander, *Optics: ordering points to identify the clustering structure*, ACM SIGMOD Record, vol. 28, ACM, 1999, pp. 49–60.

[6] D. Ashbrook and T. Starner, *Learning significant locations and predicting user movement with gps*, Wearable Computers, 2002.(ISWC 2002). Proceedings. Sixth International Symposium on, IEEE, 2002, pp. 101–108.

[7] ———, *Using gps to learn significant locations and predict movement across multiple users*, Personal and Ubiquitous Computing 7 (2003), no. 5, 275–286.

[8] N. Ashby, *Relativity in the global positioning system*, Living Reviews in Relativity 6 (2003), 1.

[9] J. Auld, C. Williams, A.K. Mohammadian, and P. Nelson, *An automated gps-based prompted recall survey with learning algorithms*, Transportation Letters: The International Journal of Transportation Research 1 (2009), no. 1, 59–79.

[10] J. Auld, C. Williams, and K. Mohammadian, *Prompted recall travel surveying with gps*.

[11] P.K. Bachu, T. Dudala, and S.M. Kothuri, *Prompted recall in global positioning sys-*

tem survey: Proof-of-concept study, Transportation Research Record: Journal of the Transportation Research Board 1768 (2001), no. -1, 106–113.

[12] P. Bahl and V.N. Padmanabhan, *Radar: An in-building rf-based user location and tracking system*, INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2, Ieee, 2000, pp. 775–784.

[13] D. Bernstein and A. Kornhauser, *An introduction to map matching for personal navigation assistants*, (1998).

[14] P. Bhawalkar, V. Bigio, A. Davis, K. Narayanaswami, and F. Olumoko, *Schedulenanny: Using gps to learn the user's significant locations, travel times and schedule*, Arxiv preprint cs/0409003 (2004).

[15] D. Birant and A. Kut, *St-dbscan: An algorithm for clustering spatial-temporal data*, Data & Knowledge Engineering 60 (2007), no. 1, 208–221.

[16] S. Burbeck, *Applications programming in smalltalk-80: How to use model-view-controller (mvc).*, Softsmarts, Inc., 1987.

[17] L. Cao and J. Krumm, *From gps traces to a routable road map*, Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2009, pp. 3–12.

[18] L.W. Carstensen, *Gps and gis: enhanced accuracy in map matching through effective filtering of autonomous gps points*, Cartography and Geographic Information Science 25 (1998), no. 1, 51–62.

[19] D. Chalko, *High accuracy speed measurement using gps (global positioning system)*, Scientific Engineering Research P/L (2002).

[20] W. Chen, M. Yu, ZL Li, and YQ Chen, *Integrated vehicle navigation system for urban applications*, Proceedings of the 7th International Conference on Global Navigation Satellite Systems (GNSS), European Space Agency, Graz, Austria, 2003, pp. 15–22.

[21] E.H. Chung and A. Shalaby, *A trip reconstruction tool for gps-based personal travel surveys*, Transportation Planning and Technology 28 (2005), no. 5, 381–401.

[22] A.J. Coutts and R. Duffield, *Validity and reliability of gps devices for measuring movement demands of team sports*, Journal of Science and Medicine in Sport 13 (2010), no. 1, 133–135.

[23] A. Csaszar, *General topology*, vol. 9, Taylor & Francis, 1978.

[24] E.W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische mathematik 1 (1959), no. 1, 269–271.

[25] D.H. Douglas and T.K. Peucker, *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, Cartographica: The International Journal for Geographic Information and Geovisualization 10 (1973), no. 2, 112–122.

[26] J. Dumont, *Trip reporting and gps-based prompted recall: Survey design and preliminary analysis of results*, Ph.D. thesis, University of Toronto, 2009.

[27] M.E. El Najjar and P. Bonnifait, *A road-matching method for precise vehicle localization using belief theory and kalman filtering*, Autonomous Robots 19 (2005), no. 2, 173–191.

[28] S. Elnekave, M. Last, and O. Maimon, *Predicting future locations using clusters' centroids*, Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems, ACM, 2007, pp. 1–4.

[29] S. Erle, R. Gibson, and J. Walsh, *Mapping hacks: tips & tools for electronic cartography*, O'Reilly Media, Inc., 2005.

[30] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining, vol. 1996, Portland: AAAI Press, 1996, pp. 226–231.

[31] A. Fathi and J. Krumm, *Detecting road intersections from gps traces*, Geographic Information Science (2010), 56–69.

[32] Flourishworks, *Succotash*, [Computer Software], 2011.

[33] J. Froehlich and J. Krumm, *Route prediction from trip observations*, SAE SP 2193 (2008), 53.

[34] T. Gaussiran, D. Muncton, B. Harris, and B. Tulman, *An open source toolkit for gps processing, total electron content effects, measurements and modeling*, International Beacon Satellite Symposium, Italy, 2004.

[35] Y. Georgiadou and A. Kleusberg, *On carrier signal multipath effects in relative gps positioning*, Manuscripta Geodaetica 13 (1988), no. 3, 172–179.

[36] S. Greaves, S. Fifer, R. Ellison, and G. Germanos, *Development of a global positioning system web-based prompted recall solution for longitudinal travel surveys*, Transportation Research Record: Journal of the Transportation Research Board 2183 (2010), no. -1, 69–77.

[37] J.S. Greenfeld, *Matching gps observations to locations on a digital map*, 81th Annual Meeting of the Transportation Research Board, 2002.

[38] T. Griffin and Y. Huang, *A decision tree classification model to automate trip purpose derivation*, (2008).

[39] T. Griffin, Y. Huang, and S. Seals, *Routing-based map matching for extracting routes from gps trajectories*, Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, ACM, 2011, p. 24.

[40] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*, Morgan Kaufmann, 2011.

[41] R. Hariharan and K. Toyama, *Project lachesis: parsing and modeling location histories*, Geographic Information Science (2004), 106–124.

[42] P.E. Hart, N.J. Nilsson, and B. Raphael, *Correction to a formal basis for the heuristic determination of minimum cost paths*, ACM SIGART Bulletin (1972), no. 37, 28–29.

[43] J. Hightower, S. Consolvo, A. LaMarca, I. Smith, and J. Hughes, *Learning and recognizing the places we go*, UbiComp 2005: Ubiquitous Computing (2005), 159–176.

[44] P.S. Hu and T.R. Reuscher, *Summary of travel trends 2001 national household travel survey*, Office (2004), no. December 2004, 1–135.

[45] Y. Huang, T. Griffin, and S. Lompo, *Intelligent system for locating, labeling, and*

logging (isl 3), Opportunities and Challenges for Next-Generation Applied Intelligence (2009), 167–173.

[46] GR Jagadeesh, T. Srikanthan, and XD Zhang, *A map matching method for gps based real-time vehicle location*, Journal of Navigation 57 (2004), no. 03, 429–440.

[47] R.R. Joshi, *A new approach to map matching for in-vehicle navigation systems: the rotational variation metric*, Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE, IEEE, 2001, pp. 33–38.

[48] J. Jun, *Smoothing methods designed to minimize the impact of gps random error on travel distance, speed, and acceleration profile estimates*, Ph.D. thesis, Department of Civil Engineering, Clemson University, 2005.

[49] B. Kanagal and A. Deshpande, *Online filtering, smoothing and probabilistic modeling of streaming data*, (2008).

[50] J.H. Kang, W. Welbourne, B. Stewart, and G. Borriello, *Extracting places from traces of locations*, Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots, ACM, 2004, pp. 110–118.

[51] A. Karbassi and M. Barth, *Vehicle route prediction and time of arrival estimation techniques for improved transportation system management*, Intelligent Vehicles Symposium, 2003. Proceedings. IEEE, IEEE, 2003, pp. 511–516.

[52] JS Kim, *Node based map matching algorithm for car navigation system*, International Symposium on Automotive Technology & Automation (29th: 1996: Florence, Italy). Global deployment of advanced transportation telematics/ITS, 1996.

[53] J. Krumm, *Real time destination prediction based on efficient routes*, (2006).

[54] ———, *A markov model for driver turn prediction*, SAE SP 2193 (2008), no. 1.

[55] ———, *Ubiquitous Advertising: The Killer Application for the 21st Century*, Pervasive Computing, IEEE 10 (2011), no. 1, 66–73.

[56] J. Krumm and E. Horvitz, *The microsoft multiperson location survey*, Microsoft ResearchTechnical Report (2005).

[57] ———, *Predestination: Inferring destinations from partial trajectories*, UbiComp 2006: Ubiquitous Computing (2006), 243–260.

[58] K. Laasonen, *Route prediction from cellular data*, Proceedings of the workshop on Context Awareness for Proactive Systems (Helsinki, Finland, June 16-17, 2005). CAPS, Citeseer, 2005, pp. 147–158.

[59] K. Laasonen, M. Raento, and H. Toivonen, *Adaptive on-device location recognition*, Pervasive Computing (2004), 287–304.

[60] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al., *Place lab: Device positioning using radio beacons in the wild*, Pervasive Computing (2005), 301–306.

[61] X. Li, H. Lin, and YB Zhao, *A connectivity based map-matching algorithm*, Asian Journal of Geoinformatics 5 (2005), no. 3, 69–76.

[62] L. Liao, *Location-based activity recognition*, Ph.D. thesis, Citeseer, 2006.

[63] L. Liao, D. Fox, and H. Kautz, *Extracting places and activities from gps traces using hierarchical conditional random fields*, The International Journal of Robotics Research 26 (2007), no. 1, 119.

[64] L. Liao, D.J. Patterson, D. Fox, and H. Kautz, *Learning and inferring transportation routines*, Artificial Intelligence 171 (2007), no. 5-6, 311–331.

[65] ”locosystech”, *”genie gt-31/bgt-31”*, 2009.

[66] S. Lyn, *comscore reports february 2011 u.s. mobile subscriber market share@ONLINE*, April 2011.

[67] J. MacQueen et al., *Some methods for classification and analysis of multivariate observations*, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, California, USA, 1967, p. 14.

[68] H. Malik, A. Rakotonirainy, F.D. Maire, and G. Larue, *Fusion of in-vehicle sensor data to develop intelligent driver training system (idts)*, (2009).

[69] I. Mansfield, *Mobile broadband subscriptions to hit one billion in 2011*, (2011).

[70] F. Marchal, J. Hackney, and K.W. Axhausen, *Efficient map matching of large global*

positioning system data sets: Tests on speed-monitoring experiment in zürich, Transportation Research Record: Journal of the Transportation Research Board 1935 (2005), no. -1, 93–100.

[71] N. Marmasse and C. Schmandt, *A user-centered location model*, Personal and Ubiquitous Computing 6 (2002), no. 5-6, 318–321.

[72] J.D. Martin, J. Krösche, and S. Boll, *Dynamic gps-position correction for mobile pedestrian navigation and orientation*, Proceedings of the 3rd Workshop on Positioning, Navigation and Communication 2006 (WPNC06), 2006.

[73] C. McCarthy, *Astronaut logs one giant check-in for foursquare@ARTICLE*, (2010).

[74] Mendhak, *Gpslogger for android*, http://github.com, 2012, GPL v2 License.

[75] A. Ng, M. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference, 2001, pp. 849–856.

[76] P. Nurmi and J. Koolwaaij, *Identifying meaningful locations*, Mobile and Ubiquitous Systems-Workshops, 2006. 3rd Annual International Conference on, IEEE, 2006, pp. 1–8.

[77] W.Y. Ochieng, M. Quddus, and R.B. Noland, *Map-matching in complex urban road networks*, Revista Brasileira de Cartografia 2 (2009), no. 55.

[78] J. Ogle, R. Guensler, W. Bachman, M. Koutsak, and J. Wolf, *Accuracy of global positioning system for determining driver performance parameters*, Transportation Research Record: Journal of the Transportation Research Board 1818 (2002), no. -1, 12–24.

[79] A. Papliatseyeu and O. Mayora, *Mobile habits: Inferring and predicting user activities with a location-aware smartphone*, 3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008, Springer, 2009, pp. 343–352.

[80] D. Patterson, L. Liao, D. Fox, and H. Kautz, *Inferring high-level behavior from low-level sensors*, UbiComp 2003: Ubiquitous Computing, Springer, 2003, pp. 73–89.

[81] N. Persad-Maharaj, S.J. Barbeau, M.A. Labrador, P.L. Winters, R. Pérez, and N.L.

Georggi, *Real-time travel path prediction using gps-enabled mobile phones*, 15th World Congress on Intelligent Transportation Systems, New York, New York, Citeseer, 2008.

[82] J.S. Pyo, D.H. Shin, and T.K. Sung, *Development of a map matching method using the multiple hypothesis technique*, Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE, IEEE, 2001, pp. 23–27.

[83] M.A. Quddus, R.B. Noland, and W.Y. Ochieng, *A high accuracy fuzzy logic based map matching algorithm for road transport*, (2006).

[84] M.A. Quddus, W.Y. Ochieng, and R.B. Noland, *Current map-matching algorithms for transport applications: State-of-the art and future research directions*, Transportation Research Part C: Emerging Technologies 15 (2007), no. 5, 312–328.

[85] M.A. Quddus, W.Y. Ochieng, L. Zhao, and R.B. Noland, *A general map matching algorithm for transport telematics applications*, GPS solutions 7 (2003), no. 3, 157–167.

[86] J.R. Quinlan, *C4. 5: programs for machine learning*, Morgan kaufmann, 1993.

[87] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, *Using mobile phones to determine transportation modes*, ACM Transactions on Sensor Networks (TOSN) 6 (2010), no. 2, 13.

[88] C. Romero, S. Ventura, P.G. Espejo, and C. Hervás, *Data mining algorithms to classify students*, Proceedings of Educational Data Mining (2008), 20–21.

[89] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, *C-dbscan: Density-based clustering with constraints*, Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (2007), 216–223.

[90] J.H. Schiller and A. Voisard, *Location-based services*, Morgan Kaufmann, 2004.

[91] N. Schüssler and K.W. Axhausen, *Identifying trips and activities and their characteristics from gps raw data without further information*, Transportation Research Record Journal of the Transportation Research Board No 2105 Transportation Research Board of the National Academies (2009), 1–28.

[92] J. Scott, J. Krumm, B. Meyers, AJ Brush, and A. Kapoor, *Home heating using gps-based arrival prediction*, Tech. report, Tech. rep., Microsoft Research, USA, 2010.

[93] R. Simmons, B. Browning, Y. Zhang, and V. Sadekar, *Learning to predict driver route and destination intent*, Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE, IEEE, 2006, pp. 127–132.

[94] T. Sohn, K.A. Li, G. Lee, I. Smith, J. Scott, and W.G. Griswold, *Place-its: A study of location-based reminders on mobile phones*, UbiComp 2005: Ubiquitous Computing (2005), 232–250.

[95] T. Sohn, A. Varshavsky, A. LaMarca, M. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. Griswold, and E. De Lara, *Mobility detection using everyday gsm traces*, UbiComp 2006: Ubiquitous Computing (2006), 212–224.

[96] P.R. Stopher, Q. Jiang, and C. FitzGerald, *Processing gps data from travel surveys*, 2nd International Colloqium on the Behavioural Foundations of Integrated Land-use and Transportation Models: Frameworks, Models and Applications, Toronto (2005).

[97] A. Subramanya, A. Raj, J. Bilmes, and D. Fox, *Recognizing activities and spatial context using wearable sensors*, Proc. of the Conference on Uncertainty in Artificial Intelligence, Citeseer, 2006.

[98] S. Syed and ME Cannon, *Fuzzy logic based-map matching algorithm for vehicle navigation system in urban canyons*, ION National Technical Meeting, San Diego, CA, vol. 1, 2004, pp. 26–28.

[99] N. Toyama, T. Ota, F. Kato, Y. Toyota, T. Hattori, and T. Hagino, *Exploiting multiple radii to learn significant locations*, Location-and Context-Awareness (2005), 24–26.

[100] D. Upadhyay, N. Schüssler, KW Axhausen, M. Flamm, and V. Kaufmann, *Optimal parameter values for mode detection in gps post-processing: An experiment*, Tech. report, Working Paper 506. IVT, ETH Zurich, Zurich, Switzerland, 2008.

[101] D. Verma and M. Meila, *A comparison of spectral clustering algorithms*, University of Washington Tech Rep UWCSE030501 1 (2003), no. 03-05-01, 1–18.

[102] P. Viswanath and R. Pinkesh, *l-dbscan: A fast hybrid density based clustering method*,

Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, vol. 1, IEEE, 2006, pp. 912–915.

[103] A.H. Weilenmann and P. Leuchovius, *I'm waiting where we met last time: exploring everyday positioning practices to inform design*, Proceedings of the third Nordic conference on Human-computer interaction, ACM, 2004, pp. 33–42.

[104] C.E. White, D. Bernstein, and A.L. Kornhauser, *Some map matching algorithms for personal navigation assistants*, Transportation Research Part C: Emerging Technologies 8 (2000), no. 1-6, 91–108.

[105] S.E. Wiehe, A.E. Carroll, G.C. Liu, K.L. Haberkorn, S.C. Hoch, J.S. Wilson, and J.D. Fortenberry, *Using gps-enabled cell phones to track the travel patterns of adolescents*, International Journal of Health Geographics 7 (2008), no. 1, 22.

[106] J. Wolf et al., *Applications of new technologies in travel surveys*, Travel Survey Methods-Quality and Future Directions (2006), 531–544.

[107] J. Wolf, R. Guensler, and W. Bachman, *Elimination of the travel diary: Experiment to derive trip purpose from global positioning system travel data*, Transportation Research Record: Journal of the Transportation Research Board 1768 (2001), no. -1, 125–134.

[108] D. Yang, B. Cai, and Y. Yuan, *An improved map-matching algorithm used in vehicle navigation system*, Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE, vol. 2, IEEE, 2003, pp. 1246–1250.

[109] H. Yin and O. Wolfson, *A weight-based map matching method in moving objects databases*, Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on, IEEE, 2004, pp. 437–438.

[110] X. Yu, *Learning significant user locations with gps and gsm*, Ph.D. thesis, Massachusetts Institute of Technology, 2006.

[111] Y. Zhao and A. House, *Vehicle location and navigation systems: Intelligent transportation systems*, Artech House (1997), 221–224.

[112] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.Y. Ma, *Understanding transportation modes*

based on gps data for web applications, ACM Transactions on the Web (TWEB) 4 (2010), no. 1, 1.

[113] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.Y. Ma, *Understanding mobility based on gps data*, Proceedings of the 10th international conference on Ubiquitous computing, ACM, 2008, pp. 312–321.

[114] Y. Zheng, L. Liu, L. Wang, and X. Xie, *Learning transportation mode from raw gps data for geographic applications on the web*, Proceeding of the 17th international conference on World Wide Web, ACM, 2008, pp. 247–256.

[115] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W.Y. Ma, *Geolife: Managing and understanding your past life over maps*, Mobile Data Management, 2008. MDM'08. 9th International Conference on, IEEE, 2008, pp. 211–212.

[116] Y. Zheng and X. Zhou, *Computing with spatial trajectories*, Springer-Verlag New York Inc, 2011.

[117] L. Zhihua and C. Wu, *A new approach to map-matching and parameter correcting for vehicle navigation system in the area of shadow of gps signal*, Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE, IEEE, 2005, pp. 449–454.

[118] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, *Discovering personal gazetteers: an interactive clustering approach*, Proceedings of the 12th annual ACM international workshop on Geographic information systems, ACM, 2004, pp. 266–273.

[119] ______, *Discovering personally meaningful places: An interactive clustering approach*, ACM Transactions on Information Systems (TOIS) 25 (2007), no. 3, 12–es.

[120] J. Zhou and R. Golledge, *A three-step general map matching method in the gis environment: Travel/transportation study perspective*, Department of Geography, University of California Santa Barbara, Tech. Rep (2004).