


January 2013

# Automatic Identification of Points of Interest in Global Navigation Satellite System Data: A Spatial Temporal Approach

Khoa Anh Tran

University of South Florida, ktran9.lbs@gmail.com

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

 Part of the [Computer Sciences Commons](#), and the [Urban Studies and Planning Commons](#)

## Scholar Commons Citation

Tran, Khoa Anh, "Automatic Identification of Points of Interest in Global Navigation Satellite System Data: A Spatial Temporal Approach" (2013). *Graduate Theses and Dissertations*.  
<http://scholarcommons.usf.edu/etd/4595>

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact [scholarcommons@usf.edu](mailto:scholarcommons@usf.edu).

Automatic Identification of Points of Interest in Global Navigation Satellite System Data:  
A Spatial Temporal Approach

by

Khoa A. Tran

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

Co-Major Professor: Miguel A. Labrador, Ph.D.  
Co-Major Professor: Sean J. Barbeau, Ph.D.  
Rafael Perez, Ph.D.  
Yu Sun, Ph.D.

Date of Approval:  
March 5, 2013

Keywords: POI, Clustering, Mobile, GPS, Tracking

Copyright © 2013, Khoa A. Tran

## DEDICATION

This thesis is dedicated to my family, especially my Mom and Dad for their love, endless support, and encouragement.

## ACKNOWLEDGMENTS

I would like to thank my two major professors, Dr. Miguel Labrador and Dr. Sean Barbeau, for their invaluable guidance, support, and patience throughout the course of this thesis. I am really thankful to them for giving me the great opportunity to work on this project. I would also like to acknowledge the feedback from my committee members, including Dr. Rafael Perez and Dr. Yu Sun, who provided important input to help shape and revise this thesis.

I would like to thank Edward Hillsman, my mentor in the Center for Urban Transportation Research (CUTR), for his guidance and supervision. Thank you to the CUTR crew for their help as I developed my career.

Last but not least, I would like to thank my family and friends for their love, constant motivation, and moral support. Without you, this thesis would have never been completed.

## TABLE OF CONTENTS

LIST OF TABLES	iii
LIST OF FIGURES	iv
ABSTRACT	v
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	1
1.2 Problem Statement	3
1.3 Contributions	5
1.4 Structure of Thesis	6
CHAPTER 2 BACKGROUND AND LITERATURE REVIEW	7
2.1 Background	7
2.1.1 GPS Auto-Sleep	8
2.1.2 Critical Points Algorithm	10
2.2 Related Works	11
2.2.1 Density-Based Spatial Clustering of Applications with Noise and Its Spatial Temporal Variation	12
2.2.2 Clustering-Based Stops and Moves of Trajectories	14
2.2.3 Stay Point Detection	16
2.2.4 Fast Clustering of Global Navigation Satellite System Data	16
CHAPTER 3 ASTIPI - THE AUTOMATIC SPATIAL TEMPORAL IDENTIFICATION OF POINTS OF INTEREST ALGORITHM	18
3.1 Definitions	18
3.1.1 Trajectory Sample	18
3.1.2 Eps-K-Neighborhood of a Point	18
3.1.3 Core Point	19
3.1.4 Directly Density-Reachable	20
3.1.5 Density-Reachable	20
3.1.6 Density-Connected	20
3.1.7 Point of Interest	20
3.2 Algorithm	22
3.2.1 Eps-K-Neighborhood	22
3.2.2 ASTIPI	25
3.2.3 Sample Execution	26

CHAPTER 4	EVALUATION	31
4.1	Experimental Design	31
4.2	Performance on Dataset with GPS Auto-Sleep Module	32
4.3	Performance on Dataset with GPS Auto-Sleep Module and Critical Points Algorithm	34
4.4	Running Time	38
CHAPTER 5	SUMMARY	41
5.1	Summary of Contributions	42
5.2	Future Work	42
REFERENCES		44
APPENDICES		49
Appendix A	Additional Tables	50
Appendix B	Reprint Permissions for Use of Figure 2.2 and Figure 2.3	54

## LIST OF TABLES

Table 2.1	Algorithms Comparison	12
Table 4.1	Values of Parameters of Each Algorithm Used in All Experiments	32
Table 4.2	Summary of Algorithm Accuracy of Modes of Transportation on Full Dataset Using GPS Auto-Sleep Module	34
Table 4.3	Summary of Algorithm Accuracy of Modes of Transportation on Reduced Datasets Using GPS Auto-Sleep Module and Critical Points Algorithm	36
Table A.1	Results of Algorithm Evaluation on Full Datasets with GPS Auto-Sleep Module	50
Table A.2	Results of Algorithm Evaluation on Reduced Datasets with GPS Auto-Sleep Module and Critical Points Algorithm	51
Table A.3	Results of ASTIPI-With-Speed Algorithm Evaluation	52
Table A.4	Results of Algorithm Average Running Time for Twenty Runs of each Test	53

## LIST OF FIGURES

Figure 2.1	Overview of a Tracking Application System	8
Figure 2.2	GPS Auto-Sleep Uses a State Machine with Various Logic Evaluations That Control the Transition Between States, Which Represent Changes to the GPS Sampling Interval Values	9
Figure 2.3	Azimuth Calculations are Used in the Critical Points Algorithm to Determine Change in Direction	11
Figure 2.4	An Example of DBSCAN Algorithm	13
Figure 2.5	An Example of the Returning Trips Problem in CB-SMoT	15
Figure 3.1	An Example of Points of Interest Given a Trajectory Sample	21
Figure 3.2	A Flow Diagram of the Eps-K-Neighborhood Search Function	24
Figure 3.3	A Flow Diagram of the ASTIPI Algorithm	29
Figure 3.4	Sample Execution of a Given Trajectory Sample of Fifteen Points that Results in Two POIs	30
Figure 4.1	Evaluation of ASTIPI, CB-SMoT, SPD, FC on Full Datasets using GPS Auto-Sleep Module	33
Figure 4.2	Evaluation of ASTIPI, CB-SMoT, SPD, FC on Reduced Datasets Using GPS Auto-Sleep Module and Critical Points Algorithm	35
Figure 4.3	Evaluation of ASTIPI-With-Speed, CB-SMoT, SPD, FC on Full Datasets Using GPS Auto-Sleep Module	37
Figure 4.4	Evaluation of ASTIPI-With-Speed, CB-SMoT, SPD, FC on Reduced Datasets Using GPS Auto-Sleep Module and Critical Points Algorithm	38
Figure 4.5	Running Time Evaluation of ASTIPI, CB-SMoT, SPD, FC on Full Datasets Using GPS Auto-Sleep Module	39



## ABSTRACT

In addition to the emergence of smartphones and tablets in recent years, the rise of Global Navigation Satellite Systems (GNSS) has allowed mobile tracking applications to become popular and be put into many uses. Analyzing tracking records to identify points of interest (POIs) is useful for both prediction applications and research such as human behavior analysis, transportation planning, and especially travel surveys. Past research in travel surveys has shown that a GPS mobile phone-based survey is a useful tool for collecting information about individuals. Moreover, a passive travel survey collection is preferred to an active travel survey method by the respondents and the analysts because it is proven to be less error prone. However, passive collection remains a challenge due to a lack of high accuracy algorithms to automatically identify trip starts and trip ends. While travel surveys need a POI identification algorithm to carry out passive information collection, mobile tracking applications must be careful not to affect the user's battery life, which limits the number of GPS coordinates that can be recorded and therefore affects the accuracies of existing POI identification algorithms. This thesis presents Automatic Spatial Temporal Identification of Points of Interest (ASTIPI), an unsupervised spatial temporal algorithm to identify POIs. ASTIPI utilizes the temporal and spatial properties of the dataset to obtain a high accuracy of POI identification, even on a reduced GPS dataset that uses techniques to conserve battery life on mobile devices. While reducing outliers within POIs, ASTIPI also has a linear running time and maintains the temporal orders of the location data so that arrival and departure information can be easily extracted and thus, users' trips can be quickly identified. Using data from real mobile devices, evaluations of ASTIPI and other existing algorithms are performed, showing that ASTIPI obtains the highest accuracy of POI identification with an average accuracy of 88% when performing on full datasets generated using the GPS Auto-Sleep module and an average accuracy of 59% when performing on reduced datasets generated using both the GPS Auto-Sleep module and the Critical Points algorithm.

## CHAPTER 1

### INTRODUCTION

#### 1.1 Motivation

Since the first telephone call from Alexander Graham Bell to his assistant Thomas Watson in 1876 [1], humanity has witnessed the invention of the first mobile phone in 1946 [2] and the birth of Simon, the first smartphone in 1993 [3]. Phones, specifically mobile phones and smartphones, are becoming smaller and cheaper every day, allowing more people to have access to these devices. According to the International Telecommunication Union, in 2011, 87% of the world's population were mobile subscribers [4].

Modern phones not only allow people to communicate over long distances but also have the power of computing. With computing capabilities, phones have become one of the most ubiquitous computing devices that enable humans to collect and process more data. Millions of applications for phones, usually referred to as “apps”, are available for use. As of November 2011, 50% of adult mobile phone owners in the United States have apps on their phones [5]. “Apps” can not only be used in phones, but also in tablets.

One of the key benefits that mobile applications have over desktop and web applications is that mobile phone users usually carry their devices with them all day, while the phones remain on the entire time. With the help of embedded GPS receivers in modern phones, mobile applications can easily collect users' locations and accurately locate users' positions at any given point in time. Location-based services have become a reality. As the number of mobile phone users and apps grows quickly, the number of location-based applications and devices also follows this increasing trend. Recent research from PEW Research Center has found that 74% of smartphone owners use their phones to get location-based information [6]. In addition, Research and Markets [7] projected

that shipment of Global Positioning System (GPS) and Location Based Service (LBS) devices will reach approximately 1,015 million units by 2015.

These numbers are fascinating because they not only show that ubiquitous computing devices are available to more people, but also show that location-based applications are of great interest to the public. Unlike other applications, the knowledge of where and when users run a location based application can be employed to improve the system or create other services. Inrix [8] is a typical example of a location based service application in which users' GPS locations are captured and provide users with real-time traffic information. Another popular use of location-based service is device tracking, as it is easier now than ever before. Mobile phone users now only have to install a tracking application to their phone and the application will run quietly in the background while regularly sending GPS locations to a remote server via cell phone networks. One apparent use of a tracking application is to locate the user in real time. However, an analyst can also pick out the users' visited places throughout a period of time to identify the users' preferences, such as going to the mall on weekends, groceries on Thursdays, etc. Additionally, identifying POIs helps applications to know the users' preferred locations from a tracking history so that applications can suggest users similar places they might find interesting. Above all, extracting POIs that users visited from a tracking dataset can be useful in many other fields, especially in travel surveys for analysis in transportation flows, travel demands, planning, etc.

Traditionally, a travel survey has been conducted by either paper-and-pencil interview (PAPI) method or computer-assisted telephone interview (CATI) method. While PAPI takes a lot of resources for distributing, collecting, and inputting data, CATI is problematic because, similar to PAPI, CATI is a "recall" technique that requires the respondents to remember many of their activities throughout a day with accurate time periods and situations [9]. Because of this problem, Miller stated in 2005 that there was a lack of accurate data for travel surveys [10].

With the advancements in technology, many travel surveys utilize GPS, where respondents' location and time are automatically recorded, and thus improving the spatial and temporal data of the survey [11] [12] [13] [14] [15]. The techniques these surveys have employed were either active GPS surveys, in which users were required to input data in electronic travel diaries prior

to the start of a trip, or passive in-vehicle GPS systems, in which GPS devices are turned on and off automatically when the car engines start or stop. Murakami et al. used the active GPS survey method for travel surveys, asking participants to record trip information before each trip [16]. This active GPS method is troublesome because respondents might forget to enter trip information as they travel during the day. In fact, a similar survey method was performed in [17] and the feedback of most participants was that they remembered to carry the devices with them but forgot to start recording trips. Meanwhile, many researchers have shown that it is possible to identify trip start/end locations, as well as start/end times by post-processing the collected in-vehicle GPS dataset [18] [19]. Although these in-vehicle surveys only focus on a small set of travel surveys, it is proven that identifying POIs from the GPS tracking dataset can re-construct trips without asking users to manually start and stop trip collection as participants travel.

With the rise of mobile phone subscribers, person-based GPS tracking allows travel surveys to be executed on trips with different modes, such as walking, biking, taking bus, etc [20] [21]. Moreover, [22] showed that compared to conventional methods, mobile phone-based travel surveys considerably reduced data handling time for surveyors and improved data quality. In [23] and [24], Asakura et al. performed a tracking survey for individual travel behavior using mobile devices. The author post-processed the collected GPS coordinates with a basic “move or stay” rule and stated that the system could be put into practice, but further improvements were required. Therefore, in addition to providing more helpful services to mobile users, an automatic identification of POIs from a GPS dataset collected via mobile devices is also vital for conducting travel surveys with less time consumption, less financial constraints, and less burden for participants.

## **1.2 Problem Statement**

Technological advancements continue to increase and allow humans to witness the bloom of location-based service applications, as more user data, especially tracking data, can be collected and put to use. For further data mining purposes, the need for extracting locations that users have visited becomes significant. However, identifying user visited locations by looking at their GPS

coordinates throughout a day is not trivial. There are five main characteristics of the collected GPS dataset.

- (1) Property #1: Spatial temporal data. Each collected coordinate is also accompanied with a time when the mobile device records the GPS fix.
- (2) Property #2: Switching between stop and movement. This is the nature of a human travel behavior. Users cannot be at two different stops without movement in-between. Based on this property, trip re-construction can be accomplished after successfully identifying POIs.
- (3) Property #3: Noise (i.e. outliers) and the loss of data. Due to limitations of current GPS receiver technology, mobile devices are often unable to record GPS fixes or else they record inaccurate coordinates, without a clear view of the sky (e.g. indoors, tree cover, mountains).
- (4) Property #4: Full dataset and reduced dataset. In order for a tracking application to be used in real life, mobile devices must dynamically sample and send less GPS fixes over the network, making the POIs identification more problematic.
- (5) Property #5: Duplicated trips. Because the tracking application records users' travel throughout an entire day, users often travel back and forth from one place to another using the same routes.

Among the above properties of the collected GPS dataset, Property #3 is a well-known problem due to satellite visibility, signal blockage, and multipath signal reflection [25] [26] [27]. Thus, identifying POIs is a challenging problem. Additionally, while Mayo [28] and Balasubramanian [29] emphasize the limited energy resources on mobile devices, Aguilar [30] specifically notices that energy consumption on mobile devices is a significant concern for tracking applications because GPS fixes need to be obtained and sent via cellular networks. Using mobile devices to receive and send frequent GPS fixes for an entire day is difficult due to the high cost of energy for GPS receivers to obtain fixes and for network communication. Various methods of collecting and sending GPS coordinates have been implemented to overcome these limitations [31] [32] [33]. The solutions these methods propose are often adjusting the sampling and sending rate of GPS fixes, and thus,

reducing the number of location data sent to the server (Property #4). Since these sampling and sending rates directly contribute to the knowledge of users' location, these rates are significant factors that considerably affect the outcome of any POI identification algorithms, making POIs identification from a reduced dataset even more difficult.

Based on GPS fixes that describe users' past travel behaviors, a few algorithms have been established to identify users' points of interest. Not only are the accuracy and evaluation of these algorithms often overlooked, but these algorithms also did not address GPS outliers, the loss of GPS data, and the dynamic sampling and sending rates of coordinates. Limitations of existing algorithms include:

- (1) Problem #1: Lack of POI identification algorithms that have a high accuracy.
- (2) Problem #2: Lack of POI identification algorithms that have a sufficiently high accuracy on a reduced GPS dataset to take into consideration the dynamic sampling and sending rates of GPS fixes, solving the limited energy resource problem on mobile devices.
- (3) Problem #3: Slow running time of  $O(n^2)$  in worst case scenarios.
- (4) Problem #4: Unable to maintain a temporal order of GPS fixes to support fast trip segmentation (i.e., quickly identifying the GPS coordinates for the trip that connects two POIs)

As a result, there is a demand for a new and efficient algorithm that intelligently identifies users' POIs from a set of geographical coordinates, collected via mobile devices with assisted GPS (A-GPS). This new algorithm needs to have a high accuracy on a full GPS dataset, as well as maintain a sufficiently high accuracy on a reduced dataset that uses different methods of collecting and sending GPS coordinates in order to save battery life on mobile device.

### 1.3 Contributions

This thesis presents a novel algorithm, Automatic Spatial Temporal Identification of Points of Interest (ASTIPI), that identifies users' POIs given Global Navigation Satellite System data, collected from the users' mobile devices. Unlike past POI identification algorithms, ASTIPI achieves

a high accuracy in identifying POIs (Problem #1) and maintains a sufficiently high accuracy on reduced datasets that implement methods of calculating and sending GPS coordinates in order to conserve battery life on mobile devices (Problem #2). In addition, the new algorithm keeps a temporal order of GPS coordinates for fast trip segmentation (Problem #4) and performs efficiently with a linear growth function as the number of GPS data points being analyzed increases (Problem #3). Last but not least, using real GPS dataset collected from mobile users, the algorithm is evaluated and compared with existing POIs identification algorithms on both full datasets and reduced datasets.

#### **1.4 Structure of Thesis**

The remainder of this thesis is organized as follows: Chapter 2 provides a detailed review of known POI identification algorithms in literature. Chapter 3 defines POIs and presents the algorithm to identify POIs from the given geographical GPS coordinates collected from mobile devices. Chapter 4 presents an evaluation of the ASTIPI algorithm and compares this new algorithm with existing literature. Chapter 5 concludes the thesis with a summary of the contributions and future research.

## CHAPTER 2

### BACKGROUND AND LITERATURE REVIEW

#### 2.1 Background

A Global Positioning System (GPS) is a space-based satellite navigation system that provides users with positioning, navigation, and timing services and is owned and operated by the United States. In recent years, other global navigation satellite systems (GNSS) have been either launched or under-developed, such as the Russian GLONASS system, the European Union Galileo satellite navigation system, the Chinese Beidou satellite navigation system, and the Indian satellite navigation system. The importance of independently-owned and operated GNSS has been increasingly recognized by many countries over the last decade because many civilian and military applications have been heavily dependent on GNSS. Among the applications that rely on GNSS, tracking applications are one of the most popular, since they can be used for many purposes such as recording commuters' trips, monitoring convicted felons, monitoring commercial driver, tracking animal migration, etc. Figure 2.1 shows an overview of a typical tracking application system.

Developed by the Center for Urban Transportation Research at the University of South Florida, TRAC-IT is a software architecture that collects travel behavior data for all modes of transportation, such as walking, biking, car, public transportation, etc. [34] In addition, it also provides real-time location-based services for GPS-enabled mobile devices. TRAC-IT uses the Location-Aware Information System Client (LAISYC) framework to efficiently manage and communicate real-time location information [35]. Typically, the TRAC-IT application on mobile devices records GPS fixes and sends them to the TRAC-IT server via a cellular provider network. To account for the limitation of battery life during the day, TRAC-IT tries to collect GPS fixes as infrequently as possible without losing any important travel behavior data. In order to accomplish that goal, the TRAC-IT mobile



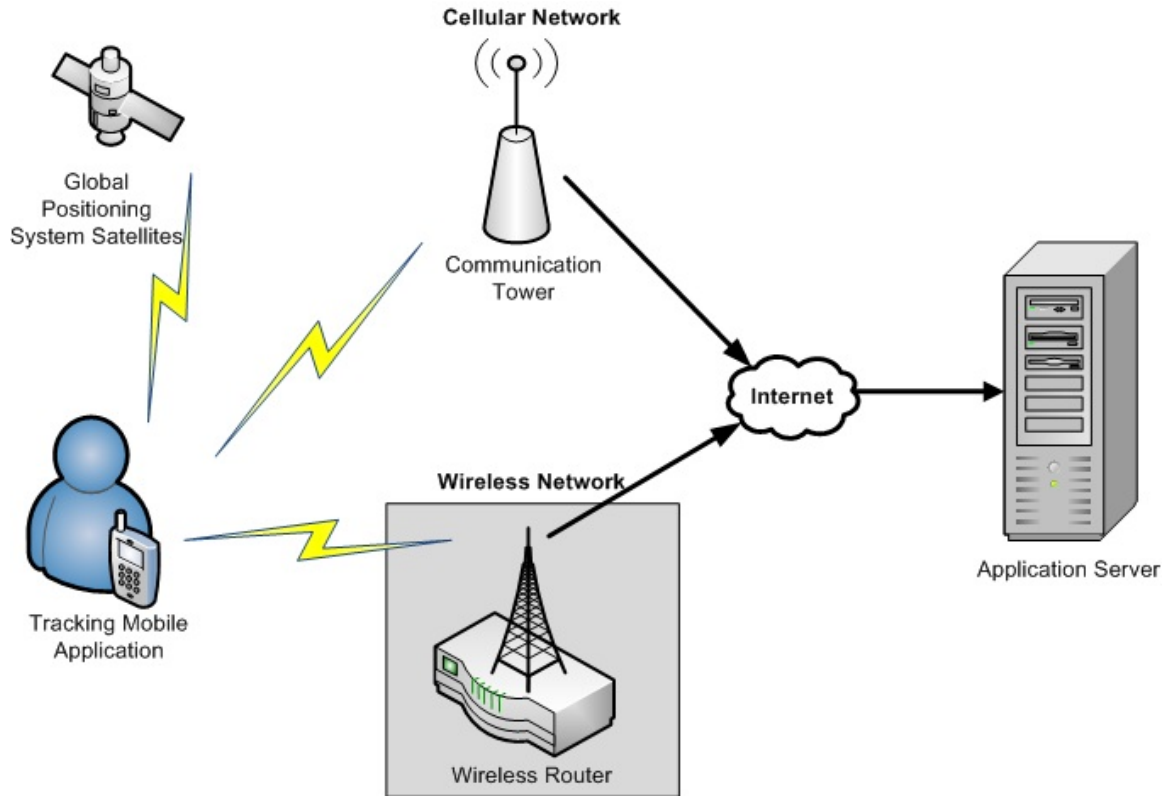


Figure 2.1: Overview of a Tracking Application System

application uses the GPS Auto-Sleep algorithm, which collects fixes at a dynamic sampling rate. In addition, the Critical Point algorithm is also utilized to eliminate non-essential GPS data before sending GPS fixes to the server.

### 2.1.1 GPS Auto-Sleep

In order for the device to be portable, the design of a mobile device is relatively small compared to a desktop or a laptop. Thus, one of the major concerns in mobile devices is its limited energy resources. A good mobile application needs to take into consideration its use of battery energy. In a typical tracking application, one of the largest energy consumption sources is the use of GPS to obtain real-time location information. TRAC-IT attempts to consume less of the battery's energy by collecting location data only when it is useful. The GPS Auto-Sleep module in TRAC-IT is developed to serve this purpose by dynamically adjusting the GPS sampling rate based on user movement [36].

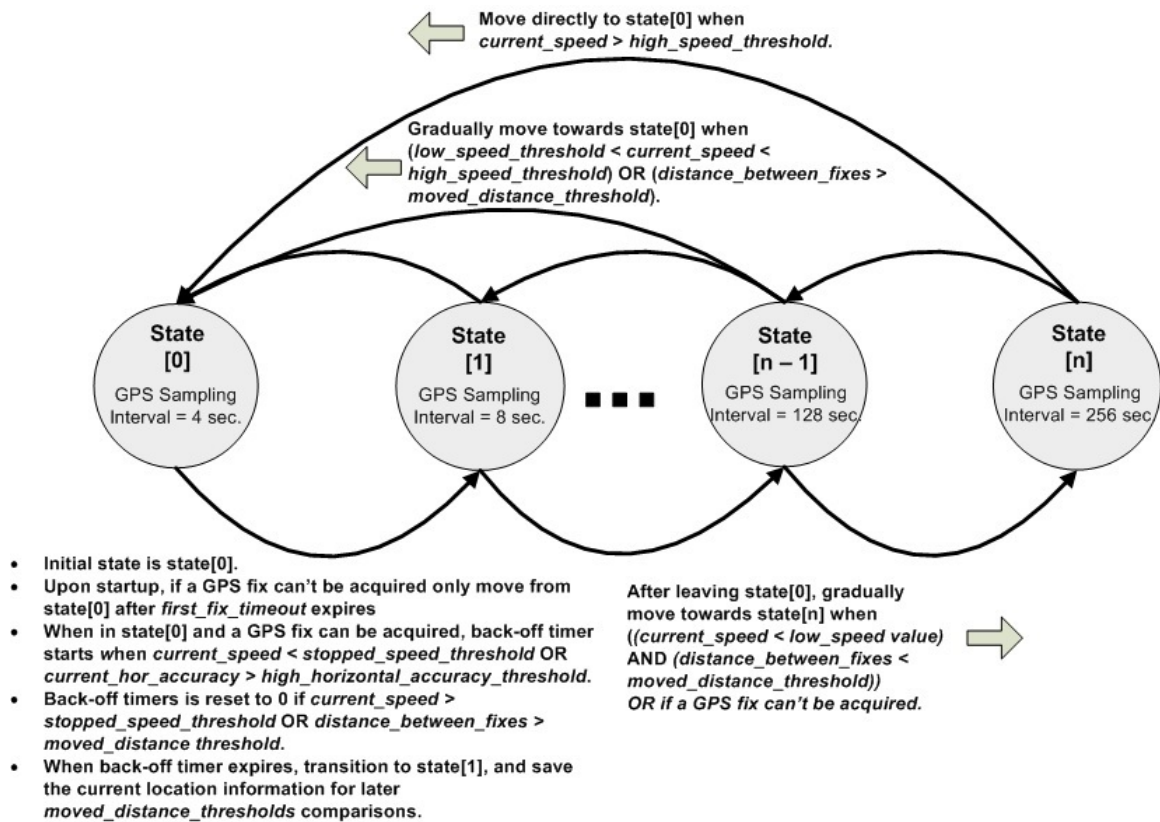


Figure 2.2: GPS Auto-Sleep Uses a State Machine with Various Logic Evaluations That Control the Transition Between States, Which Represent Changes to the GPS Sampling Interval Values. [36] Copyright 2008, IEEE

The idea of the GPS Auto-Sleep module is simple, but efficient. When the user is actively moving, the user's location data is captured at a high resolution so no significant travel behaviors are missed. When the user is stationary, the location data is redundant. Moreover, when the user is stationary, it usually happens indoors. Due to sensor failure, weak signals, and interference indoors, the mobile devices often cannot obtain a GPS fix or the GPS fix is inaccurate (i.e., noisy). At that time, the device continuously attempts to calculate a GPS fix every few seconds and thus, a large amount of energy is wasted. Therefore, location data should not be recorded as frequently if the user is not moving. GPS Auto-Sleep implements this idea by using a Location Aware State Machine. The GPS sampling interval is determined by which state it is currently in. Changes between states in the state machine happen when certain conditions are met or not. Figure 2.2 demonstrates the design of the location aware state machine.

The GPS Auto-Sleep module utilizes speed, time, distance between GPS fixes, and horizontal accuracy to evaluate conditions for state transition. Lower numbered states indicate that the user is moving and thus the sampling interval is small (i.e., more frequent). Higher numbered states indicate that the user slows down and stays at the same location, and thus the mobile device should go into a “sleep” mode to save battery energy. Once the module detects that the user is moving again, it can quickly jump to the first state to capture the user’s location data at a high resolution. More details about the GPS Auto-Sleep module can be found in [37], [36], and [38]. The TRAC-IT mobile application currently has six states with six different GPS sampling intervals, starting from state[0] with 4 seconds, state[1] with 8 seconds, state[2] with 16 seconds, state[3] with 64 seconds, state[4] with 150 seconds, and finally state[5] with 256 seconds. These values are used in the experiments in Chapter 4.

### 2.1.2 Critical Points Algorithm

While the GPS Auto-Sleep module saves battery energy by intelligently reducing the GPS sampling rate, the Critical Points (CP) algorithm module attempts to save energy by sending only GPS fixes that contribute to the knowledge of the travel path. By filtering location data before transmitting them to the server, the CP algorithm module effectively increases the time interval between transmissions, which significantly increases battery life. In addition, sending less data over the cellular network also helps users’ budget on their data plan.

The CP algorithm filters out GPS fixes by looking at the change in direction of the path [36]. If the traveling path is a straight line (i.e., no change in direction), points along the path do not contribute to the knowledge of a path. Only the first point and last point in the path are ‘critical’ and necessary to construct a straight line traveling path. Furthermore, a single GPS fix is ‘critical’ and enough to represent the location where the user is stationary. To implement the idea of CP, at any time of the execution, the CP algorithm keeps references to three points: Last Critical Point, Last Trigger Point, and Current Point. As the name suggests, the Last Critical Point is the last known point that is considered as critical. The Last Trigger Point is the point under consideration as critical. The CP algorithm determines on-the-fly whether a GPS fix (i.e., the Last Trigger Point) is critical.

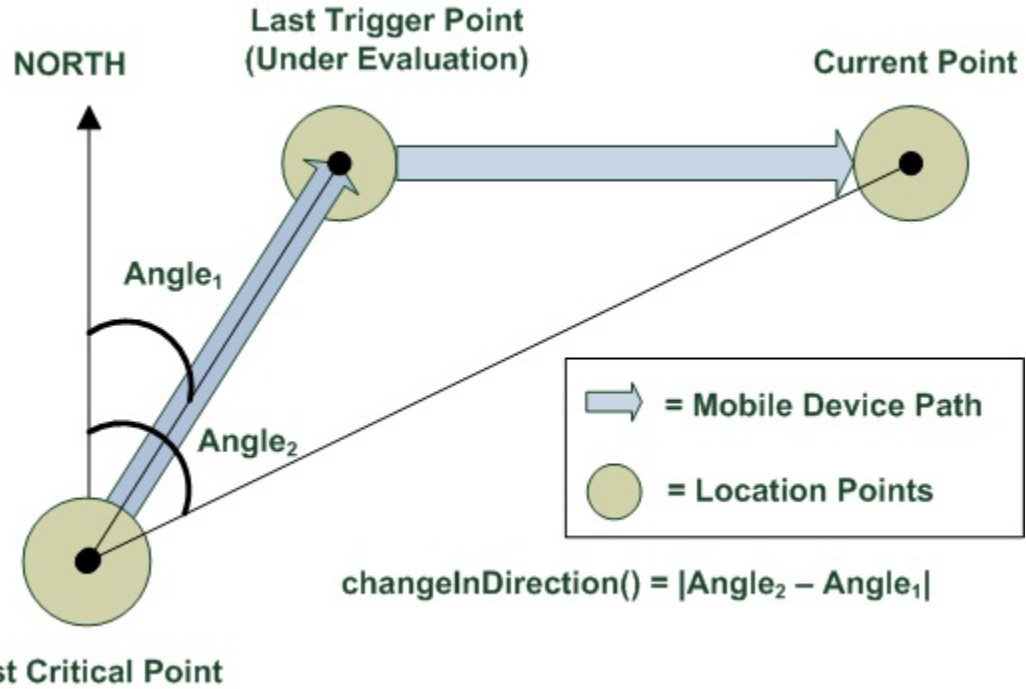


Figure 2.3: Azimuth Calculations are Used in the Critical Points Algorithm to Determine Change in Direction. [36] Copyright 2008, IEEE

When a new location data is retrieved, it will be fed to the algorithm as Current Point. These three points are then evaluated and if certain conditions are met, the Last Trigger Point is determined as critical and the references are updated (i.e., the Last Trigger Point is now the Last Critical Point; the Current Point is now the Last Trigger Point). The time complexity for the CP algorithm is said to be linear and the algorithm is able to run in real time as new GPS fixes are obtained. Figure 2.3 demonstrates the azimuth calculations to determine change in direction. Further information can be found in [37] and [39].

## 2.2 Related Works

At a glance, it may seem that identifying POIs from a set of GPS data would be trivial because analysts can easily determine POIs by looking at the spatial density of coordinates. However, taking into consideration GPS outliers, the loss of location data, and different algorithms (e.g., GPS Auto-Sleep, CP algorithm) for collecting and sending GPS fixes to save battery energy, the answer is

not obvious. Spatial clustering alone, without considering time information, creates errors when identifying POIs. In this section, some of the existing techniques to identify POIs from a collection of GPS fixes are reviewed. Table 2.1 is an overview of the existing works that are discussed in this chapter.

Table 2.1: Algorithms Comparison

	DBSCAN	ST-DBSCAN	CB-SMoT	Stay Point Detection	Fast Clustering
Overview	Uses <i>MinPoints</i> to determine spatial density for clustering	Extension of DBSCAN for spatial and temporal values	Extension of DBSCAN that uses <i>MinTime</i> instead of <i>MinPoints</i>	Uses long distance and long duration for clustering	Uses AVL tree to maintain temporal order for easy trip segmentation
Running Time	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^2 \log(n))$
Spatial Temporal	No	Yes	Yes	Yes	No
Problems	Noise	No	No	Yes	Yes
	Returning Trips	Yes	Yes	Yes	No
	Loss of Location Data	Yes	Yes	No	No

## 2.2.1 Density-Based Spatial Clustering of Applications with Noise and Its Spatial Temporal Variation

In [40], Ester et al. presented Density Based Spatial Clustering of Applications with Noise (DBSCAN), a well-known spatial clustering algorithm that can identify arbitrary-shaped objects and detect noise in a dataset. Using the notion of *Eps-Neighborhood* of a point, DBSCAN decides whether a point  $P$  is a *Core Point* if the number of *Eps-Neighbors* is more than *MinPoints*. Then, a new cluster is formed by points that are density-reachable or density-connected from  $P$ . In Figure 2.4, if *MinPoints* is four, then point  $D$  is directly density-reachable from point  $B$  and point  $E$  is directly density-reachable from point  $C$ . Since point  $D$  and point  $E$  are both density-reachable from point  $A$ , points  $D$  and  $E$  are said to be density-connected to each other by  $A$ . While all points in Figure 2.4 but  $N$  belong to the same cluster, point  $N$  does not belong to any clusters and therefore, it is a *Noise*.

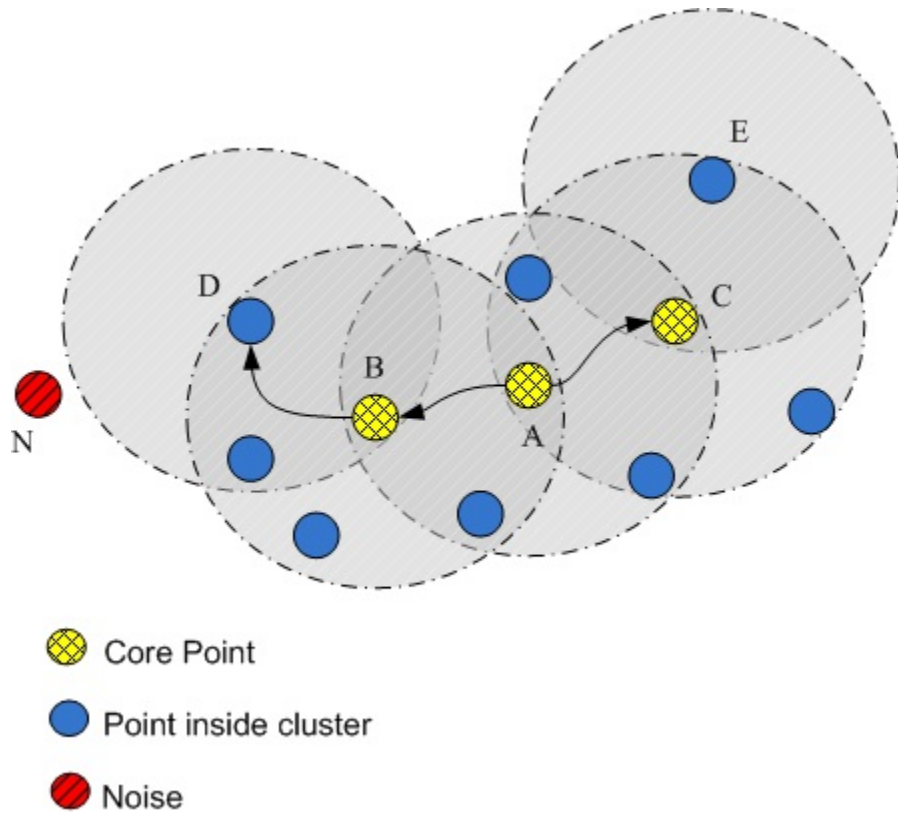


Figure 2.4: An Example of DBSCAN Algorithm

Using the DBSCAN algorithm to identify POIs is not appropriate because *MinPoints* cannot be used with GPS datasets that use intelligent techniques such as GPS Auto-Sleep and the CP algorithm for energy-saving, since noise and loss of location data can occur (i.e., a POI does not necessarily have a large number of GPS fixes). With different methods of collecting and sending location data, it is possible to have one or two GPS fixes at a POI. Furthermore, DBSCAN does not take full advantage of the spatial temporal characteristics of the dataset, resulting in the algorithm's low accuracy (Problem #1).

Introduced by Birant et al., the ST-DBSCAN algorithm [41] is an extension of DBSCAN that clusters spatial temporal data. Unlike the original DBSCAN which only performs clustering on spatial values, ST-DBSCAN has the ability to discover clusters based on the non-spatial, spatial, and temporal values of objects. To include other values besides spatial values, the algorithm modifies the `RetrieveNeighbors()` function such that neighbors of an object must meet both the spatial

condition (i.e.,  $Eps1$ ) and the non-spatial condition (i.e.,  $Eps2$ ). Adding another parameter  $Eps2$  to account for the temporal aspect of the data, ST-DBSCAN still treats them separately in the evaluation function. This is a problem in POI identification as it is difficult to find a value for  $Eps2$  for time. The parameter  $MinPoints$  used for determining *Core Point* remains a problem, while the running time is not improved from  $O(n^2)$  for worst case (Problem #3).

## 2.2.2 Clustering-Based Stops and Moves of Trajectories

Instead of identifying POIs, a slightly different set of problems is computing stops and moves by testing the intersections of trajectories with a given list of geographical locations. Palma et al. [42] proposed the Clustering-Based Stops and Moves of Trajectories (CB-SMoT) algorithm to not only compute stops and moves but also find interesting places that have been left out in the given list of geographical locations. The CB-SMoT algorithm is a two-step algorithm: first, the algorithm identifies potential stops, and second, unknown stops are distinguished from the list of potential stops. Only the first step of the algorithm is considered in this thesis because it focuses on identifying POIs from a sample trajectory. CB-SMoT modified the DBSCAN algorithm to consider the temporal aspect of the dataset. Instead of determining core points by the number of neighboring points, the algorithm utilizes the duration between the first point and the last point in the region to see if it exceeds  $MinTime$ . By extending the DBSCAN algorithm, CB-SMoT is able to detect outliers and at the same time also considers the loss of location data. In addition, CB-SMoT uses the *quantile function* to obtain a relative parameter of  $Eps$  distance related to the mean and the standard deviation of the dataset. The relative parameter allows CB-SMoT to stay away from defining an absolute  $Eps$  value, which is difficult in many cases.

Nonetheless, there are three major drawbacks of the CB-SMoT algorithm. First, the running time complexity of CB-SMoT remains the same as the running time of the DBSCAN algorithm, which is  $O(n^2)$  (Problem #3). Moreover, the *quantile function* requires a priori knowledge of the proportion between points inside potential stops and total points in the dataset. This proportion varies among datasets because users sometimes spend a whole day inside a stop (i.e., the proportion is one), while on different occasions they might be visiting more than ten stops (i.e., the proportion



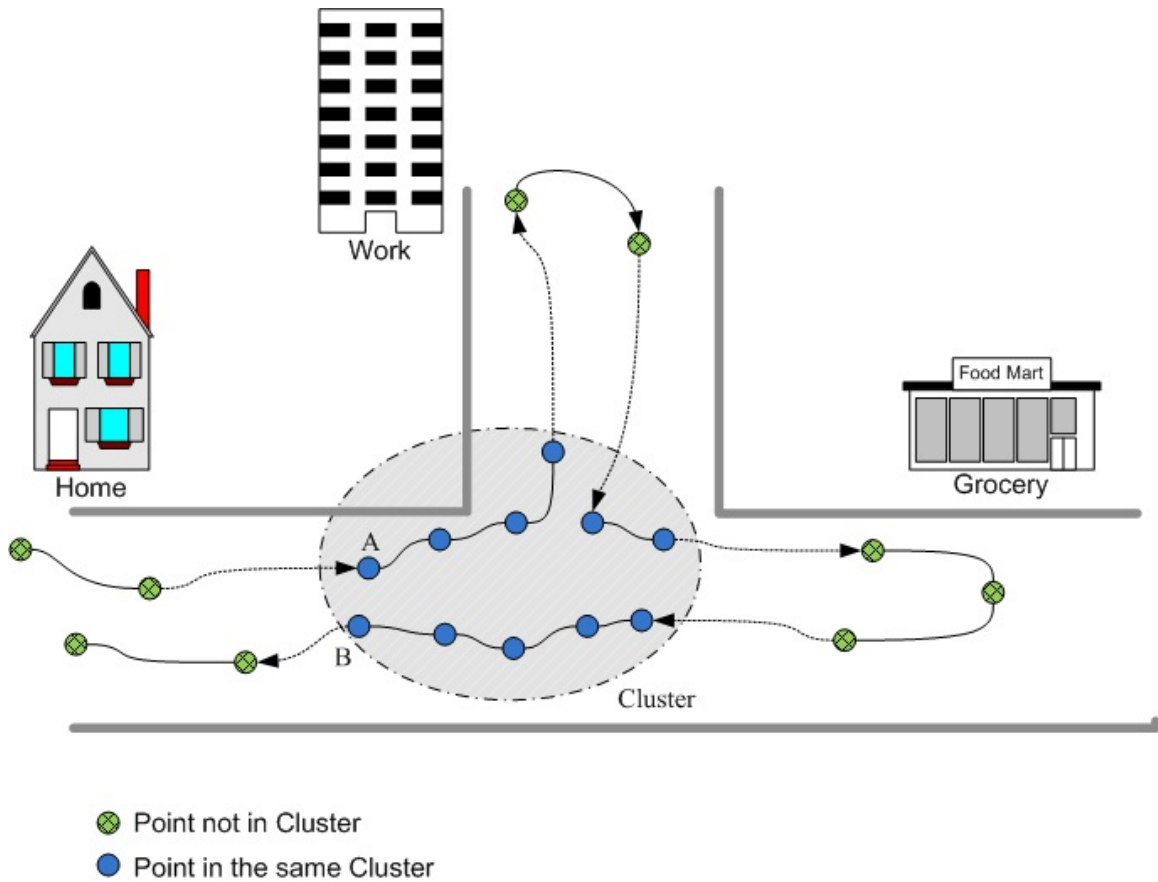


Figure 2.5: An Example of the Returning Trips Problem in CB-SMoT

is much smaller than one). In addition, the mean and the standard deviation of distance among coordinates vary when a dynamic GPS sampling rate is involved. Users traveling the same distance might have different numbers of recorded GPS fixes. Additionally, when searching for neighbors of a point, the algorithm looks at every point in the input trajectory. Returning trips (e.g. home to work in the morning and work to home in the afternoon) are an issue (Figure 2.5), as CB-SMoT will include location data from two different times of the day. With a trajectory that contains the GPS fixes of users' travel throughout a day, the returning trips have a significant impact on the accuracy of the algorithm (Problem #1). Figure 2.5 demonstrates the returning trips problem of the CB-SMoT algorithm.

In Figure 2.5, user Alice travels from home to work at 8 a.m. When she leaves work in the afternoon, she decides to go grocery shopping and then go back home at 6 p.m. As Alice has no



interest at the incorrectly identified Cluster since she does not even stop at Cluster, Cluster is not a POI. However, Cluster is constructed because of the close proximity among points and although CB-SMoT determines density by duration, the time difference between point A and point B is rather long (i.e., about 10 hours). This is incorrect, since although two points are spatially close to each other, they happen at different times of the day.

### 2.2.3 Stay Point Detection

In [43], Li et al. proposed a hierarchical graph-based similarity measurement framework to model human location history and measure similarity among users. In order to extract a user's location information for further data mining and features, Li et al. use the Stay Point Detection (SPD) algorithm. To detect a stay point, the SPD algorithm looks for a region where the user spends a period of time exceeding a certain threshold. The idea behind this algorithm is rather simple. If the distance between two points is far and their duration is short, the user is actively moving. Vice versa, if the distance is short and their duration is long, the user has just spent time in a stay point. Therefore, all the points in between belong to the same stay point.

While the SPD algorithm takes into consideration both distance and time, it is not able to detect GPS outliers. In fact, if loss of data and GPS noise occur consecutively (e.g. indoors), the algorithm will detect GPS noise as a Stay Point. Once the mobile device is able to obtain a correct coordinate again, another Stay Point will be detected. This is a double error, preventing the algorithm from achieving a high accuracy (Problem #1). Another problem with the algorithm happens when all points in the dataset are at the same stay point. The SPD algorithm will be unable to detect a stay point because the distance threshold will never be reached. This is also the worst case scenario for the SPD algorithm with the running time of  $O(n^2)$  (Problem #3).

### 2.2.4 Fast Clustering of Global Navigation Satellite System Data

In [44], Persad identified logical POIs from a large amount of raw GPS data using an agglomerative hierarchical clustering approach. The Fast Clustering (FC) algorithm represents clusters by AVL trees to maintain a temporal order among location data so that after the execution of the FC

algorithm, trips can easily be constructed by retrieving the maximum and minimum nodes in the tree. By taking the agglomerative hierarchical clustering approach, the FC algorithm begins with considering each coordinate as a cluster. Then clusters are combined by an AVL-tree-merge operation if the distance between any two points in associated clusters are within a certain threshold. In addition to maintaining the temporal order within a cluster, another benefit of the FC algorithm is its space-saving property, since the algorithm maintains a  $O(n)$  memory storage throughout the course of the execution ( $n$  is the number of GPS fixes).

Nonetheless, there are several drawbacks of this algorithm. First of all, since FC maintains the temporal order of coordinates within a cluster, it needs to perform an AVL-tree-merge resulting in an additional  $\log(n)$  of the running time. Thus, the time complexity of the FC algorithm is  $O(n^2 \log n)$  (Problem #3). Second, while GPS noise is a known problem of GPS-enabled mobile devices, there is no noise detection and reduction in the FC algorithm. Third, this algorithm introduces “pseudo-POIs”, which are small clusters that do not represent meaningful locations to the users (e.g., traffic light stops). The algorithm cannot differentiate between “pseudo-POIs” and real POIs, and thus detects many incorrect POIs (Problem #1). Last but not least, although the result of the algorithm is ordered by time, the temporal property of the GPS fix is not involved in the clustering process, which does not take full advantage of the spatial temporal properties of the dataset.

## CHAPTER 3

### ASTIPI - THE AUTOMATIC SPATIAL TEMPORAL IDENTIFICATION OF POINTS OF INTEREST ALGORITHM

This chapter presents ASTIPI, a new algorithm for identifying POIs that has a high accuracy (Problem #1) and a linear running time (Problem #3) while maintaining a temporal order of GPS fixes for fast trip segmentation (Problem #4). ASTIPI also takes into account the dynamic sampling and sending rates of GPS coordinates to address battery limitations in mobile devices (Problem #2).

#### 3.1 Definitions

##### 3.1.1 Trajectory Sample

As previously discussed, mobile devices collect GPS fixes and send them to a server. The list of those GPS fixes in time order is called a *Trajectory Sample*. According to Alvares et al. [45], the definition of a *Trajectory Sample* is introduced below.

**Definition 1.** A *trajectory sample*  $T$  is a list of space-time points  $\{p_0 = (x_0, y_0, t_0), p_1 = (x_1, y_1, t_1), \dots, p_N = (x_N, y_N, t_N)\}$ , where  $x_i, y_i \in \mathbb{R}, t_i \in \mathbb{R}^+$  for  $i = 0, 1, \dots, N$  and  $t_0 < t_1 < t_2 < \dots < t_N$

The number of GPS fixes in a *Trajectory Sample* is not limited. A *Trajectory Sample* may contain one or two GPS coordinates to thousands of GPS fixes. In the context of this thesis, a *Trajectory Sample* is a collection of GPS fixes that a server receives from one mobile device (i.e., one user) in a day.

##### 3.1.2 Eps-K-Neighborhood of a Point

**Definition 2.** The *Eps-K-Neighborhood* of a point  $p_a$ , denoted by  $N_{(Eps,K)}(p_a)$ , is the maximal sub-trajectory  $\{p_{s_1} = (x_{s_1}, y_{s_1}, t_{s_1}), \dots, p_{s_m} = (x_{s_m}, y_{s_m}, t_{s_m})\}$  of a trajectory sample  $T \{p_0 =$

$(x_0, y_0, t_0), p_1 = (x_1, y_1, t_1), \dots, p_N = (x_N, y_N, t_N)\}$  such that for any points  $p_{s_i}$  in  $N_{(Eps, K)}(p_a)$ ,  $dist(p_a, p_{s_i}) \leq Eps$  and for any two consecutive points  $p_{s_i}, p_{s_{i+1}}$  in  $N_{(Eps, K)}(p_a)$ ,  $|s_i - s_{i+1}| < K$ .

To construct the *Eps-K-Neighborhood* of a point, two parameters *Eps* and *K* are needed. While *Eps* represents the maximum distance between a point  $p_a$  and its neighbors, *K* represents the maximum difference in index of two consecutive coordinates in the same neighborhood. The parameter *K* serves as a temporal property such that it prevents the algorithm from looking too far ahead in time and allows returning trips to be seen as different trips. This parameter also helps in reducing noise from the dataset, as the algorithm will be slow to react to a change in distance. Lastly, the parameter *K* is a stopping condition so that ASTIPI does not look for its entire dataset, allowing the algorithm to run in linear time. Further analysis of the algorithm's running time will be discussed in later sections.

### 3.1.3 Core Point

**Definition 3.** A point  $p = (x_p, y_p, t_p)$  of a trajectory is called *Core Point* with respect to *Eps*, *MinTime*, and *K* if

- (1)  $|t_{s_m} - t_p| \geq MinTime$  where  $s_m$  is the last point of  $N_{(Eps, K)}(p)$  OR
- (2)  $|t_{s_m} - t_{s_{m+1}}| \geq MinTime$  where  $s_m$  is the last point of  $N_{(Eps, K)}(p)$  and  $s_{m+1}$  is the first next point not in  $N_{(Eps, K)}(p)$

Notice that when  $N_{(Eps, K)}(p)$  (i.e., the *Eps-K-Neighborhood* of point  $p$ ) is empty,  $t_{s_m}$  defaults to  $t_p$ .

Instead of the number of points, defining *Core Point* by *MinTime* is essential to avoid data integrity problems such as noise, absence of coordinates due to limitations in GPS receivers, and dependency on GPS fix sampling rate. Moreover, as of the end of 2012, mobile devices still have problems collecting GPS fixes indoors. In some scenarios, mobile devices only record one GPS fix before coming into a building and record another GPS fix once it gets outside. With the different strategies for calculating and sending GPS coordinates, tracking applications may put the phone into idle mode to save battery life and thus, two consecutive GPS fixes are far apart in both distance and

time. To further address this problem of indoor GPS receivers and the dynamic sampling and sending rates, the algorithm also considers the duration of the last coordinate in the *Eps-K-Neighborhood* and the first next coordinate not in *Eps-K-Neighborhood* of a point. This means that even if  $p$  has no *Eps-K-Neighbors*,  $p$  can still be a *Core Point* as long as the difference in time between  $p$  and the next point in the trajectory sample is greater than *MinTime*.

### 3.1.4 Directly Density-Reachable

**Definition 4.** A point  $q$  is directly density-reachable to a point  $p$  if  $q \in N_{(Eps,K)}(p)$  and  $p$  is a *Core Point* with respect to *Eps*, *MinTime*, and  $K$ .

### 3.1.5 Density-Reachable

**Definition 5.** A point  $q_0$  is density-reachable from a point  $p$  with respect to *Eps*, *MinTime*, and  $K$  if there exists a chain  $q_0, q_1, q_2, \dots, q_N$  where  $q_N = p$  and  $q_k$  is directly density-reachable to  $q_{k+1}$ .

### 3.1.6 Density-Connected

**Definition 6.** Two points  $p$  and  $q$  are density-connected with respect to *Eps*, *MinTime*, and  $K$  if there exists a point  $o$ , and both  $p$  and  $q$  are density-reachable from  $o$ .

### 3.1.7 Point of Interest

**Definition 7.** A point of interest  $S$  of a trajectory sample  $T$  with respect to *Eps*, *MinTime*, and  $K$  is a non-empty subtrajectory of  $T$  satisfying the following conditions:

- (1)  $\forall p, q \in T$ : if  $p \in S$  and  $q$  is density-reachable from  $p$  with respect to *Eps*, *MinTime*, and  $K$ , then  $q \in S$ .
- (2)  $\forall p, q \in S$ :  $p$  is density-connected to  $q$  with respect to *Eps*, *MinTime*, and  $K$ .
- (3)  $\exists p \in S$ :  $p$  is a *Core Point*.

A point of interest (POI) is a collection of GPS fixes ordered by time that describes one common logical geographic location. In many location-based services and applications (e.g., geocoding),

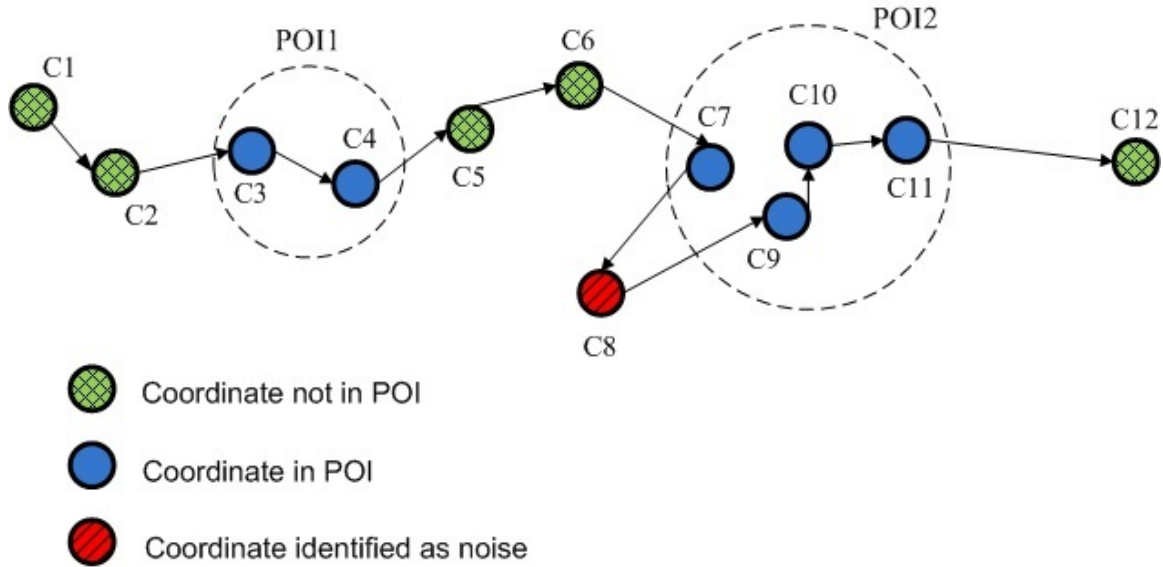


Figure 3.1: An Example of Points of Interest Given a Trajectory Sample

regardless of how large the geographic location is, a location is simplified to one coordinate (i.e., one pair of latitude and longitude), which is often referred to as a POI. However, in this algorithm, for a detailed analysis of the dataset, coordinates are grouped together when these GPS fixes are significant to the user's travel behaviors and depict the same logical geographic location. Each group of coordinates is considered as a POI. The benefit of grouping location data that shares the same meaning to users is that with a set of GPS fixes indicating the same location, analysts can always go back and represent the location with one coordinate (e.g., the center coordinate of a POI). Moreover, from the identified POIs generated by the algorithm, the amount of time users spend at a place is easily extracted. Trip information is also conveniently constructed by the relationship among POIs. Figure 3.1 is an illustration of two POIs, denoted by *POI1* and *POI2*.

In Figure 3.1, the duration that the user spent at *POI1* can be computed by the difference in time between *C4* and *C3*. Similarly, the duration spent at *POI2* is the time difference between *C11* and *C7*. A trip from *POI1* to *POI2* is generated by points between *C4* and *C7*. Notice that coordinates in *POI2* do not have consecutive indexes because *C8* is an outlier. Other alternative names of POI are *Stay Point* [43], *Stop* [45], and *Place* [46].

In general, the *trajectory sample* is the input of the algorithm while the POIs are the output that the algorithm provides.

## 3.2 Algorithm

### 3.2.1 Eps-K-Neighborhood

As discussed in Section 1.2, one of the characteristics of a tracking application dataset is its spatial temporal property (Property #1). However, while the DBSCAN algorithm clusters objects in any shapes and eliminates noise in the dataset, it only considers the spatial properties: the distance among points and the number of points inside a region. Determining density of a region by the number of points is a problem in the collected GPS dataset because in addition to GPS outliers, the loss of location data also occurs. Therefore, the ASTIPI algorithm determines density by the duration that a user spends inside a region. The duration in a region is computed by first finding a neighborhood and then calculating the time difference between the first coordinate and the last coordinate in that neighborhood. To further address the problem of loss of data, if any two coordinates are consecutive in time and their time difference is long, regardless of their proximity, ASTIPI determines the former coordinate is dense (i.e., *Core Point*), even if it has no neighbors. Using time to identify density has two benefits. First, density is not dependent on the number of points, which minimizes the problem of loss of GPS fixes and also accounts for various techniques of collecting and sending GPS fixes for the purpose of saving battery life. Second, the duration value can be adjusted for purposes of data mining and applications.

Another modification to the DBSCAN algorithm in ASTIPI is the  $K$  parameter, mentioned in Definition 2. The parameter  $K$  is used as a stopping condition for the neighborhood search of a point.  $K$  is the limit of the number of consecutive coordinates that are not within a certain distance threshold. In the original DBSCAN algorithm, in order to find neighbors of a point  $P$ , DBSCAN needs to compute distances between  $P$  and all other points in the dataset. Noise detection and elimination are accomplished because the DBSCAN neighbors search does not stop immediately once the proximity condition is not met. In order to be considered as noise in DBSCAN, the point  $P_{noise}$  needs to be compared with all other points, and once DBSCAN guarantees that  $P_{noise}$  does

not belong to any clusters,  $P_{noise}$  is identified as noise. Meanwhile, ASTIPI utilizes the fact that the collection of GPS fixes is spatial temporal and at the same time, is also in time order. Therefore, ASTIPI only needs to perform the search for the next  $K$  coordinates. If those  $K$  more GPS fixes are not within the region of point  $P$ , ASTIPI is confident that the user is moving away from  $P$ . A small  $K$  value prevents the algorithm from noise detection and elimination. On the other hand, the larger the  $K$  parameter is, the more confidence ASTIPI has that the user is moving away, and outlier detection and elimination can be achieved. Nonetheless, a large  $K$  value results in a longer running time of the algorithm. In addition to noise detection and elimination, using parameter  $K$  for stopping neighbors search helps to solve the problem of returning trips, mentioned in Section 2.2.2. The solution to the returning trips problem is to not look too far ahead in time. With a sufficiently small value of  $K$ , the problem is solved because the search process does not look for the entire dataset but stops after  $K$  more processing times.

Like the original DBSCAN, the goal of the *Eps-K-Neighborhood* function is to find coordinates in the trajectories that are close to a point  $P$ . These GPS fixes need not be too far away in time from  $P$ . In order to meet this goal, ASTIPI utilizes the nature of human travel behaviors: travel time and travel distance are highly correlated. Given the same amount of time, the distance covered by a person in motion is farther than that of a person who is stationary. Also, given the same amount of distance, the time it takes for a person while moving is shorter than that of a person who is staying in one place. To find the *Eps-K-Neighborhood* of a coordinate *coord*, ASTIPI begins the search at the position *startIndex*. For each GPS fix *currCoord* starting at *startIndex*, the algorithm sees if *currCoord* is within reach of *coord*. If they are close to together, *currCoord* is a *Eps-K-Neighbor* of *coord* and ASTIPI continues looking at the next GPS fix. If the two coordinates are not within reach, ASTIPI increases the *notInRangeCount* by one until this value exceeds a certain threshold, then ASTIPI stops the search and sees if *coord* has met the criteria to be a *Core Point*. The *Eps-K-Neighborhood* function returns the list of coordinates, ordered by time, and the decision of the point *coord* is a *Core Point*. Figure 3.2 illustrates the flow of the algorithm.

Algorithm 1 is the pseudo-code of the *Eps-K-Neighborhood* function. In addition to the pool of coordinates, the *Eps* parameter, and the  $K$  parameter, the *Eps-K-Neighborhood* function requires



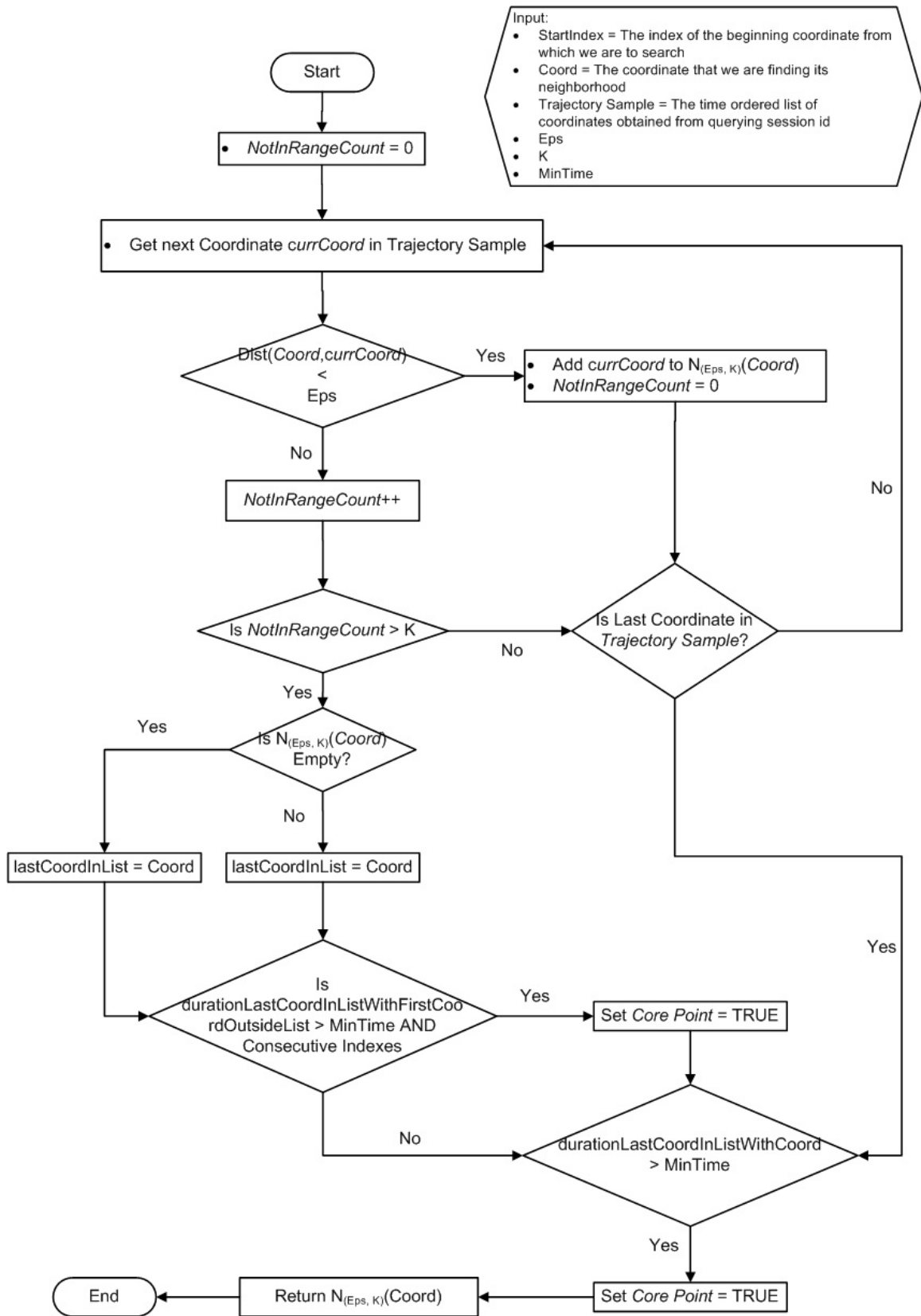


Figure 3.2: A Flow Diagram of the Eps-K-Neighborhood Search Function

---

**Algorithm 1:** Eps-K-Neighborhood(startIndex, coord)

---

**Require: Input:** A trajectory sample  $T$  of size  $n$ , where  $n \geq 0$

**Require: Output:** Eps-K-Neighborhood of coord (i.e.,  $N_{(Eps,K)}(coord)$ )

```
1: coord.setCorePoint(false)
2: notInRangeCount = 0
3: for  $i = startIndex, i < T.size()$  do
4:    $currCoord = T.get(i)$ 
5:   if  $distance(currCoord, coord) < Eps$  then
6:     Add  $currCoord$  to neighborhood list  $N_{(Eps,K)}(coord)$ 
7:     notInRangeCount = 0
8:   else
9:     notInRangeCount = notInRangeCount + 1
10:    if notInRangeCount >  $K$  then
11:       $lastCoordInList = N_{(Eps,K)}(coord).isEmpty ? coord : N_{(Eps,K)}(coord).last$ 
12:      if  $durationLastCoordInListWithFirstCoordOutsideList > MinTime$  AND
13:         $i - K = lastCoordInListIndex + 1$  then
14:          coord.setCorePoint(true)
15:        end if
16:      break
17:    end if
18:  end for
19:  $lastCoordInList = N_{(Eps,K)}(coord).isEmpty ? coord : N_{(Eps,K)}(coord).last$ 
20: if  $durationLastCoordInListWithCoord > MinTime$  then
21:   coord.setCorePoint(true)
22: end if
23: return  $N_{(Eps,K)}(coord)$ 
```

---

the starting index from which the search should begin and the current coordinate on which the neighborhood search is performed. Unlike the neighborhood search in DBSCAN (also referred to as regionQuery), which searches from the beginning to the end of the dataset, the starting index that is passed in from the main program helps to significantly reduce the processing time.

### 3.2.2 ASTIPI

To incorporate the temporal properties of the GPS fixes, ASTIPI re-defines the meaning of a *Core Point* and the meaning of neighbors of a point. ASTIPI utilizes the *Eps-K-Neighborhood* function to implement the new definitions and address the problems of noise detection and reduction, loss of location data, and returning trips. Since the search for neighbors does not look for the

entire collection but stops after few number of coordinates that are outside a pre-defined range, the running time of the algorithm can be significantly improved. Modifications to the main DBSCAN algorithm is necessary in ASTIPI so that the algorithm keeps track of the coordinate indexes, and thus, only performs necessary calculations. In general, in order to find POIs, ASTIPI needs to determine if a point  $P$  is a *Core Point* and find coordinates that are density-reachable from  $P$  or density-connected to  $P$  with respect to  $Eps$ ,  $MinTime$ , and  $K$ . Specifically, given a list of time-ordered coordinates, the ASTIPI begins at the first coordinate  $P_0$  in the trajectory sample. It first finds the *Eps-K-Neighborhood*  $N_{(Eps,K)}(P_0)$  of  $P_0$  and determines if  $P_0$  is a *Core Point*. If  $P_0$  is not a *Core Point*, the algorithm continues with the next GPS fix in the time-ordered trajectory sample. If  $P_0$  is a *Core Point*, ASTIPI will look for the *Eps-K-Neighbors* of each point  $P_i$  in  $N_{(Eps,K)}(P_0)$  and check if  $P_i$  is a *Core Point*. If  $P_i$  is a *Core Point*, all coordinates in *Eps-K-Neighbors* of  $P_i$  are added to  $N_{(Eps,K)}(P_0)$  and the process continues until the end of the list  $N_{(Eps,K)}(P_0)$ . While executing the *Eps-K-Neighborhood* function, the algorithm maintains the index of the last coordinate in the *Eps-K-Neighborhood* of the most recent *Core Point* (i.e., *currIndex*). By keeping track of this index, ASTIPI performs the *Eps-K-Neighborhood* search only on the necessary location data. At the end of this process, ASTIPI will have all points that are either density-reachable from  $P_0$  or density-connected to  $P_0$ . The algorithm continues to look at the next GPS fix in the trajectory sample based on the *currIndex* that ASTIPI has been keeping track of. Figure 3.3 demonstrates the flow of ASTIPI. The pseudo-code of this algorithm is also provided in Algorithm 2.

### 3.2.3 Sample Execution

Figure 3.4 depicts a sample trajectory of a user's travel of fifteen different location data starting from  $C_1$  to  $C_{15}$ . Suppose  $MinTime$  is 300 seconds,  $Eps$  is 100 meters and  $K$  is three. ASTIPI first searches for the *Eps-K-Neighborhood* of  $C_1$  by computing the distance from  $C_1$  to  $C_2$ ,  $C_3$ , and  $C_4$ . Because all distances are less than 100 meters, these three points are added to the neighborhood of  $C_1$ . The *Eps-K-Neighborhood* function continues to calculate the distance from  $C_1$  to  $C_5$ ,  $C_6$ , and  $C_7$ . Since the distances to these three coordinates are not within 100 meters from  $C_1$  and the

---

**Algorithm 2:** ASTIPI(T)

---

**Require: Input:** A trajectory sample  $T$  of size  $n$ , where  $n \geq 0$

**Require: Output:** A list of POIs ordered by time

```
1:  $currIndex = 0$ 
2: while  $currIndex < T.size()$  do
3:    $currCoord = T.get(currIndex)$ 
4:    $EKN = Eps\text{-}K\text{-Neighborhood}(currIndex + 1, currCoord)$ 
5:   if  $currCoord$  is CorePoint then
6:     Add  $currCoord$  and all its neighbors to the new point of interest  $poi$ 
7:      $currIndex = \text{index of last coordinate in EKN}$ 
8:      $i = 0$ 
9:     while  $i < EKN.size()$  do
10:       $Coord_{EKN} = EKN.get(i)$ 
11:       $EKN_{sub} = Eps\text{-}K\text{-Neighborhood}(currIndex + 1, Coord_{EKN})$ 
12:      if  $Coord_{EKN}$  is CorePoint then
13:         $currIndex = \text{index of last coordinate in } EKN_{sub}$ 
14:        Add all coordinates in  $EKN_{sub}$  to  $EKN$ 
15:        Add all coordinates in  $EKN_{sub}$  to  $poi$ 
16:      end if
17:       $i = i + 1$ 
18:    end while
19:  end if
20:   $currIndex = currIndex + 1$ 
21:  Add  $poi$  to  $pointsOfInterest$ 
22: end while
23: return  $pointsOfInterest$ 
```

---

number of coordinates outside 100 meters has reached three, the *Eps-K-Neighborhood* function stops and returns  $N_{(100,3)}(P_1)$ . In addition, since the time difference between  $C_4$  and  $C_1$  is 500 seconds, the algorithm sets  $C_1$  to a *Core Point*. After discovering that  $C_2$ ,  $C_3$ , and  $C_4$  are the *Eps-K-Neighbors* of  $C_1$  and  $C_1$  is a *Core Point*, ASTIPI executes the *Eps-K-Neighborhood* function on  $C_2$  by computing the distances from  $C_2$  to  $C_5$ ,  $C_6$ , and  $C_7$ . Because these three GPS fixes are more than 100 meters from  $C_2$ , ASTIPI stops the search. Next, the algorithm performs an *Eps-K-Neighborhood* search on  $C_3$  and sees that  $C_5$  is not close to  $C_3$  but  $C_6$ ,  $C_7$ , and  $C_8$  are within 100 meters of  $C_3$ . Thus,  $C_6$ ,  $C_7$ , and  $C_8$  are added to the neighborhood of  $C_3$ . The *Eps-K-Neighborhood* search on  $C_3$  stops after computing the distance from  $C_3$  to  $C_{11}$ . Since ASTIPI discovers that  $C_3$  is a *Core Point*,  $C_6$ ,  $C_7$ , and  $C_8$  are added to  $N_{(100,3)}(P_1)$ . The algorithm continues by executing the

*Eps-K-Neighborhood* function on  $C_4$ ,  $C_6$ ,  $C_7$ , and  $C_8$  with a starting search index of nine. Since none of these coordinates is a *Core Point*, no more GPS fixes are added to  $N_{(100,3)}(P_1)$ . Then,  $N_{(100,3)}(P_1)$  is a new POI.

ASTIPI continues with  $C_9$  and  $C_{10}$  and determines that these two points are not *Core Points*. Although the *Eps-K-Neighborhood* function of  $C_{11}$  returns no neighbors, the algorithm determines that  $C_{11}$  is a *Core Point* because the time difference between  $C_{11}$  and  $C_{12}$  is 680 seconds. Thus, a new POI is constructed with only one coordinate  $C_{11}$  in that POI. The ASTIPI algorithm ends at  $C_{15}$ . Notice that although  $C_{15}$  is within 100 meters of  $C_1$ ,  $C_{15}$  is not a *Eps-K-Neighbor* of  $C_1$  because the *Eps-K-Neighborhood* search for  $C_1$  ends after computing the distance to  $C_7$ . This is the intended behavior of ASTIPI, as  $C_{15}$  is not close in time with  $C_1$  although the two points are close in distance. At the end of the execution, two POIs are correctly identified.

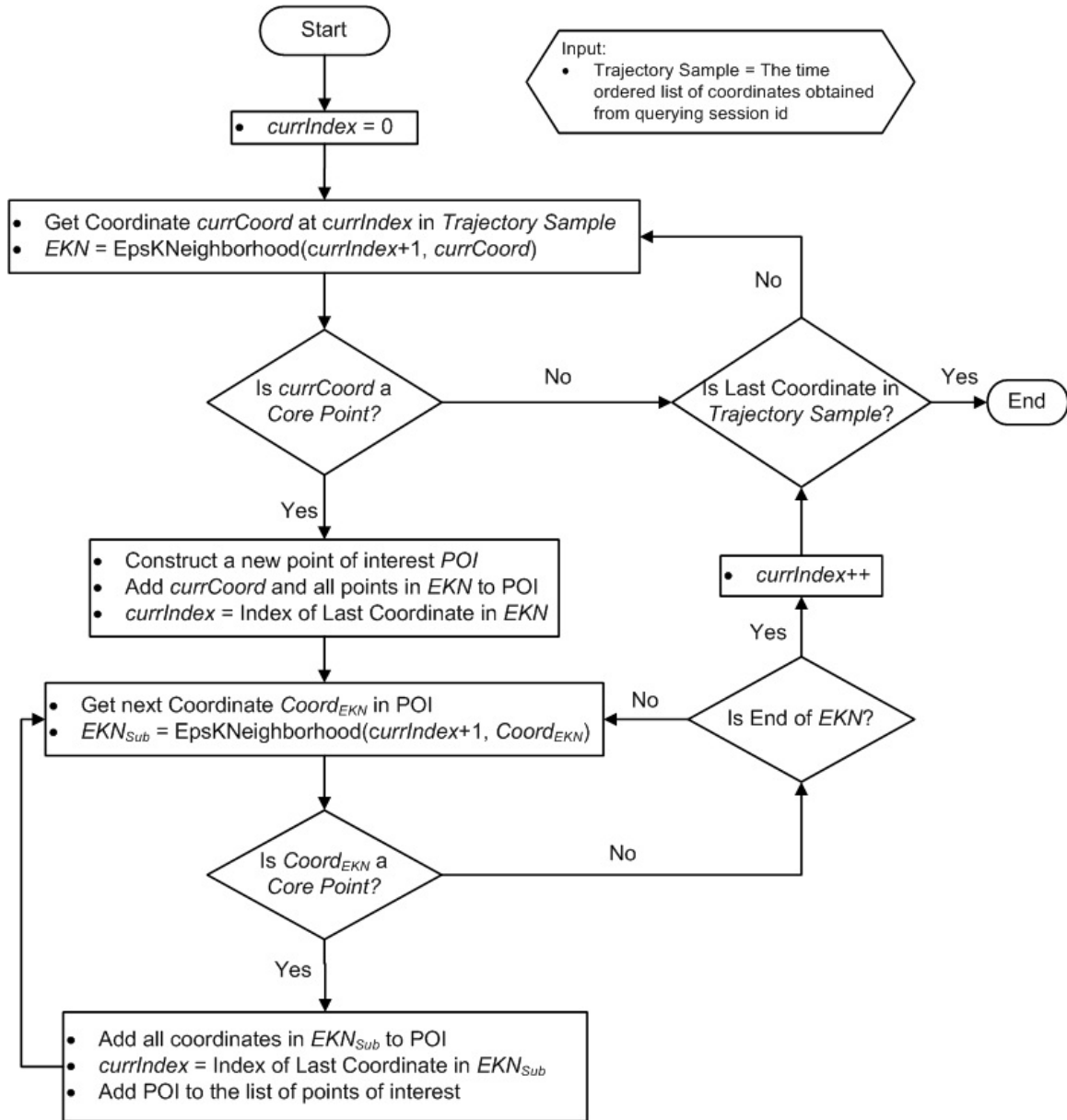


Figure 3.3: A Flow Diagram of the ASTIPI Algorithm

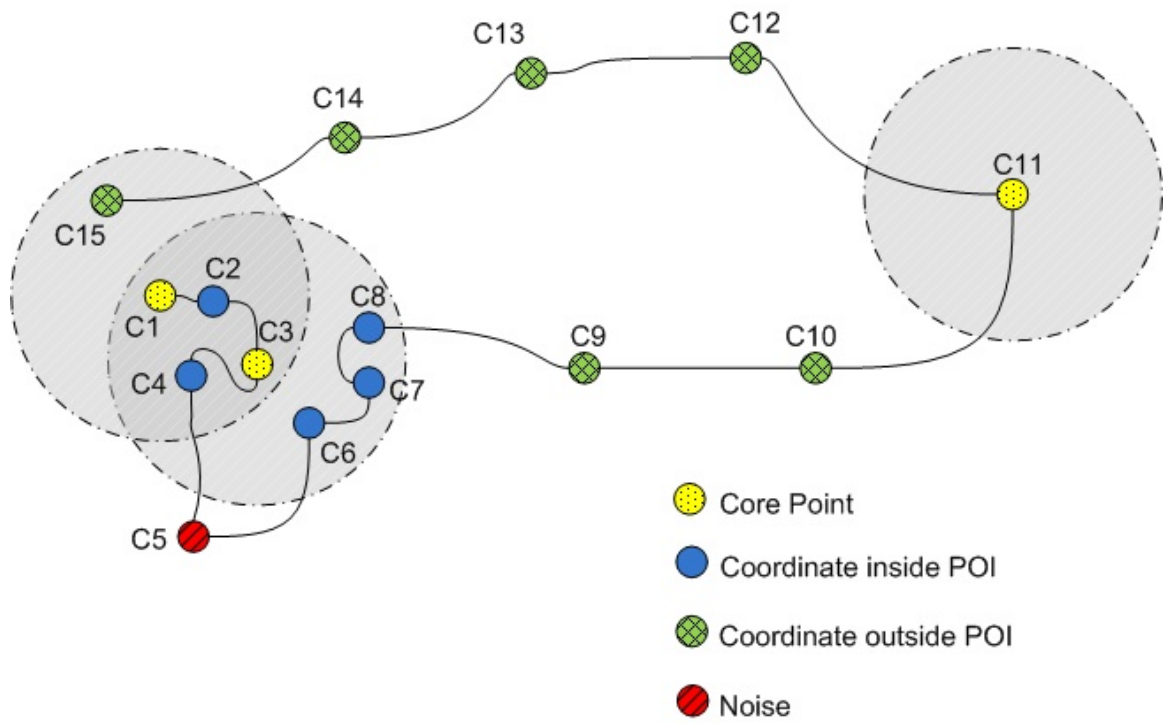


Figure 3.4: Sample Execution of a Given Trajectory Sample of Fifteen Points that Results in Two POIs

## CHAPTER 4

### EVALUATION

As indicated in Chapter 2, CB-SMoT and SPD use the spatial temporal property of the dataset (Property #1) to identify POIs. Furthermore, FC is also an algorithm specifically designed for automatic identification of POIs. Therefore, this section compares the ASTIPI algorithm with the CB-SMoT, SPD, and FC algorithms by evaluating their accuracies on datasets using the GPS Auto-Sleep module (full datasets) and on datasets using a combination of the GPS Auto-Sleep module and the CP algorithm (reduced datasets). This section also compares the running time of ASTIPI against other algorithms' running times.

#### 4.1 Experimental Design

To evaluate the three algorithms, each algorithm is run using a set of previously recorded GPS coordinates. All 27 datasets are collected by TRAC-IT, the mobile application mentioned in Section 2.1, running on a Sanyo Pro 200 with A-GPS. While walking, driving, and riding bus, a user carries the phone throughout a day, reporting the places he visited at the end of each day. These data are recorded and considered as the ground truth data for evaluation. Experiments are performed on the dataset with the GPS Auto-Sleep module, and with a combination of the GPS Auto-Sleep module and the Critical Points (CP) algorithm for conserving battery life on mobile devices. These two datasets are essentially the same, with the only difference that the second one has a significantly fewer number of GPS fixes due to the CP algorithm.

In order to compute the accuracy of the algorithm, definitions of *True Positive*, *False Positive*, and *False Negative* are presented below.



- True Positives (TP): The number of correctly identified POIs that describe users' visited locations.
- False Positives (FP): The number of POIs users did not visit, but the algorithm identifies as POIs.
- False Negatives (FN): The number of POIs users visited, but the algorithm does not identify as POIs.

Then, the accuracy of each algorithm is computed as follows:

$$Accuracy = \frac{TP}{TP + FP + FN} \quad (4.1)$$

The values of duration threshold to be considered an activity vary in literature such as 45 seconds and 120 seconds in [47], 300 seconds in [48], or even 900 seconds in [21]. For this thesis, a duration threshold of five minutes (i.e., 300 seconds) is selected. Table 4.1 describes the values of algorithm parameters used in all experiments. These values are the ones that each corresponding algorithm achieves the highest accuracy when running on all datasets.

Table 4.1: Values of Parameters of Each Algorithm Used in All Experiments

Parameters	ASTIPI	CB-SMoT	SPD	FC
MinTime	5 minutes	5 minutes	5 minutes	N/A
Eps	100 meters	100 meters	180 meters	40 meters
K	3	N/A	N/A	N/A

## 4.2 Performance on Dataset with GPS Auto-Sleep Module

The GPS Auto-Sleep module is a part of the TRAC-IT mobile application that controls the GPS sampling rate. As previously mentioned in Section 2.1.1 the GPS Auto-Sleep module in TRAC-IT has six states with six different GPS sampling interval starting from state[0] with 4 seconds, state[1] with 8 seconds, state[2] with 16 seconds, state[3] with 64 seconds, state[4] with 150 seconds, and finally, state[5] with 256 seconds.

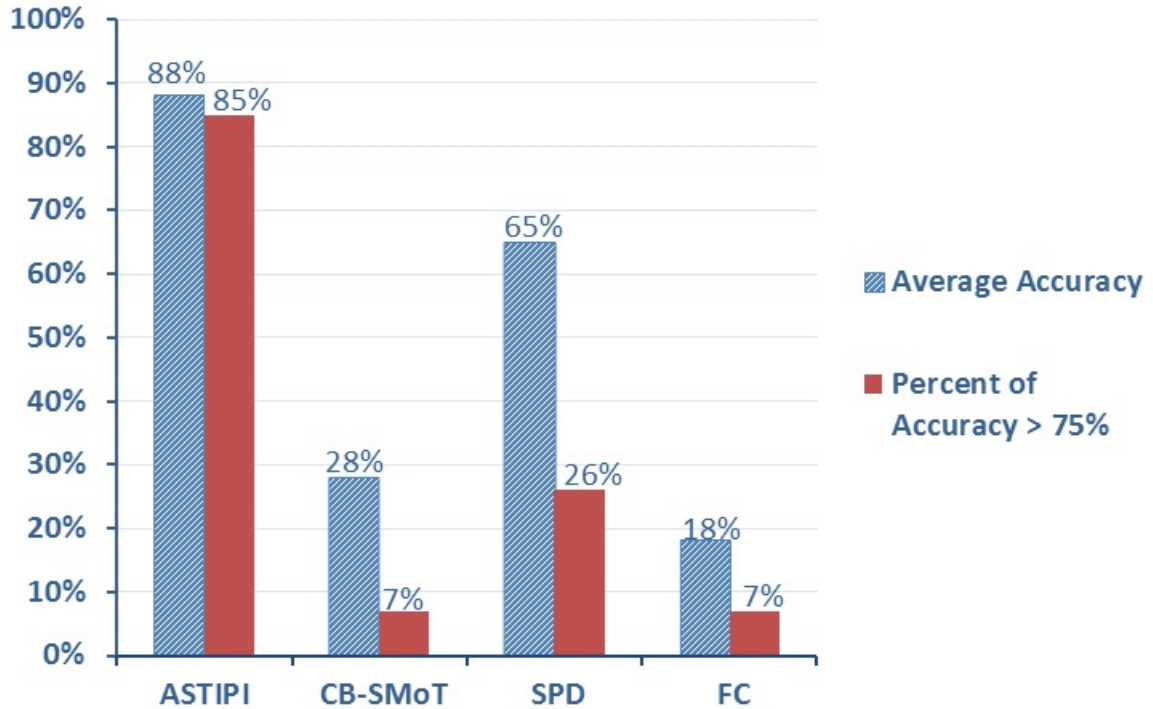


Figure 4.1: Evaluation of ASTIPI, CB-SMoT, SPD, FC on Full Datasets using GPS Auto-Sleep Module

Figure 4.1 shows that among the four algorithms analyzed using full datasets generated by the GPS Auto-Sleep module, ASTIPI yields the highest accuracy with an average of 88% of accuracy, and 23 of the 27 tests (i.e., 85% of all test cases) have an accuracy greater than 75%. ASTIPI performs well on all test cases but one (i.e. Test #4 in Table A.1) when the user visits places that are close to each other (e.g. classes on campus within walking distance). This scenario is expected when using the GPS Auto-Sleep module because while the phone was sleeping, the user moved between classes. By the time the phone woke up, the user was already indoors and GPS Auto-Sleep module could not calculate a new GPS coordinate, and thus, the module thinks the user was stationary and did not attempt to collect GPS fixes that contribute to the knowledge of the user's location. On the contrary, while the SPD algorithm yields an average of 65% accuracy, only 26% of the test cases have an accuracy greater than 75%. This large percentage gap between the average accuracy and percent of test cases suggests that SPD performs well in certain scenarios, but poorly on different datasets. In fact, as previously mentioned in Section 2.2.3, SPD cannot detect any POIs

Table 4.2: Summary of Algorithm Accuracy of Modes of Transportation on Full Dataset Using GPS Auto-Sleep Module

Mode of Transportation	Number of Tests	ASTIPI (%)	CB-SMoT (%)	SPD (%)	FC (%)
Stationary	2	100	100	0	100
Walk	3	69	39	46	37
Walk & Bus	3	67	23	60	22
Walk & Bus & Drive	1	100	13	63	30
Walk & Drive	14	93	22	75	5
Drive	4	94	15	82	3

when the user stays in one place and the 180 meters threshold is never exceeded (e.g. Test #1 and #2 in Table A.1). Meanwhile, CB-SMoT has the third highest average accuracy and FC has the lowest average accuracy among the four algorithms. When splitting up the datasets based on the mode of transportation, ASTIPI remains the algorithm with the highest accuracy on every mode of transportation. However, while ASTIPI performs well when the user is stationary, when using a combination of walking, riding bus, and driving, when using a combination of walking and driving, or when only driving, the algorithm has an average accuracy of lower than 70% for walking, and for walking and riding bus. Again, this is an expected problem of the GPS Auto-Sleep module when the user travels a short distance in a short amount of time while the phone is still asleep. The results of the experiments are provided in Table A.1.

#### 4.3 Performance on Dataset with GPS Auto-Sleep Module and Critical Points Algorithm

Typically, the CP algorithm would be implemented in the TRAC-IT mobile application to filter the GPS dataset before it reaches the server, therefore saving battery energy used in wireless transmissions. However, in this thesis the GPS dataset is post-processed using the CP algorithm so ASTIPI can be analyzed both on the original dataset (Section 4.2) as well as the subset of data generated by the CP algorithm. According to [37], the following thresholds for the CP algorithm are used:

- min\_speed\_threshold = 0.1 meters per second
- max\_walk\_speed = 0.6 meters per second

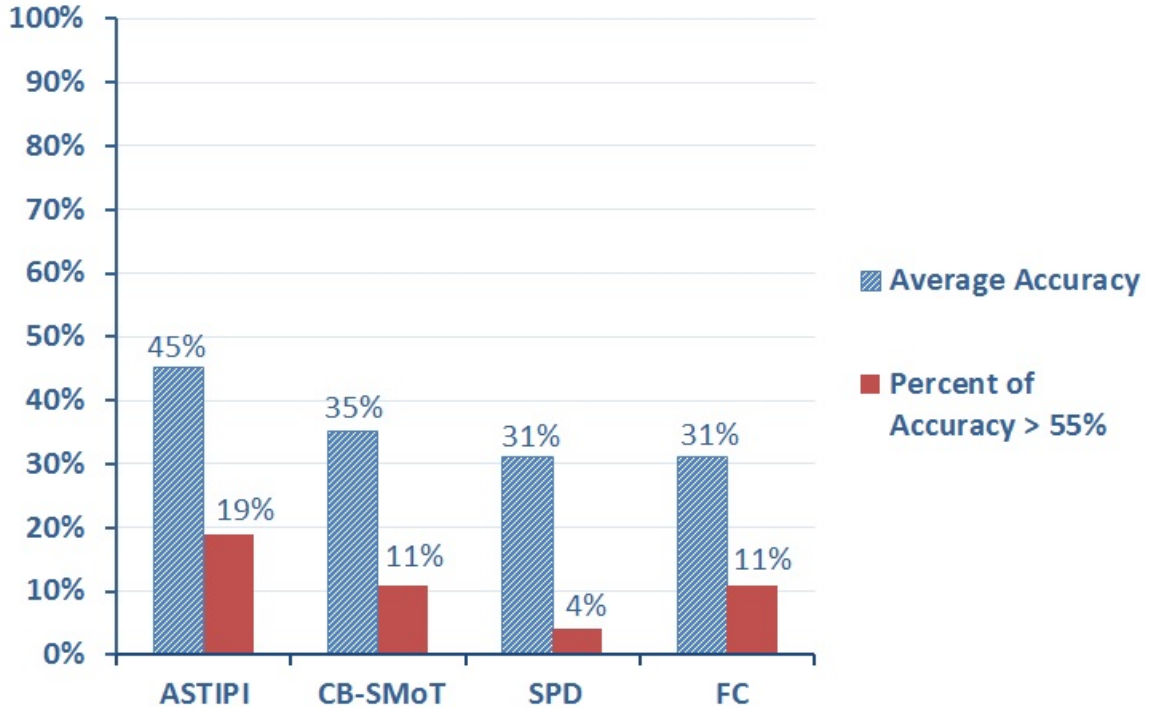


Figure 4.2: Evaluation of ASTIPI, CB-SMoT, SPD, FC on Reduced Datasets Using GPS Auto-Sleep Module and Critical Points Algorithm

- angle\_threshold = 4.5 degrees for walk trips and 3 degrees for car trips.

Figure 4.2 shows that among the four algorithms, ASTIPI is still the most accurate algorithm with an average accuracy of 45%, and 19% of the 27 tests have an accuracy greater than 55%. CB-SMoT comes in second place with an average accuracy of 35%, and 11% of the test cases have an accuracy greater than 55%. Similar to the performance on the datasets with only the GPS Auto-Sleep module on, the gap between the average accuracy and the percentage of test cases having an accuracy greater than 55% of the SPD algorithm is large because of its inability to discover POIs when all the location data represents only one place in the entire dataset. Moreover, for each mode of transportation, ASTIPI remains the most accurate algorithm to identify POIs on reduced datasets. Nonetheless, unlike its performance on full datasets, ASTIPI's accuracy on a combination of walking and driving modes and driving mode only drop significantly compared to other modes of transportation within the same algorithm.

Table 4.3: Summary of Algorithm Accuracy of Modes of Transportation on Reduced Datasets Using GPS Auto-Sleep Module and Critical Points Algorithm

Mode of Transportation	Number of Tests	ASTIPI (%)	CB-SMoT (%)	SPD (%)	FC (%)
Stationary	2	75	100	0	100
Walk	3	34	24	31	29
Walk & Bus	3	35	30	32	22
Walk & Bus & Drive	1	71	43	63	31
Walk & Drive	14	47	31	40	27
Drive	4	34	24	36	21

Although ASTIPI yields the highest accuracy among the four algorithms, its average accuracy is still less than 50%. One reason for the low accuracy is the lack of knowledge about the user's location. The CP algorithm reduces the number of GPS coordinates 70% or more; in certain cases, that number is more than 90%. And the trade-off for sending less number of GPS fixes to conserve battery life is the loss of knowledge about user's location. Another reason for the low accuracy is that ASTIPI is sensitive to determining POIs when the duration between two consecutive coordinates is more than *MinTime* (i.e. 5 minutes in the experiments). This sensitivity is appropriate in a high resolution dataset, as it indicates the user has just gone outside from a building (i.e. because the loss of GPS signal indoors) or the user starts moving again after the phone has been sleeping for some time to save battery life. However, with the involvement of the CP algorithm, a GPS fix is recorded only if there is a significant change in direction and thus, the duration between two consecutive points exceeding *MinTime* does not indicate a POI, resulting in a large number of False Positives in the ASTIPI algorithm. Details of the experiments are described in Table A.2

To improve the performance of ASTIPI on the CP algorithm dataset by reducing the number of False Positives, besides duration, speed needs to be included when determining a *Core Point*. Definition 3 was modified to describe the *Core Point With Speed*.

**Definition 8.** A point  $p = (x_p, y_p, t_p)$  of a trajectory is called core point with speed with respect to  $Eps$ ,  $MinTime$ , and  $K$  if

- (1)  $|t_{s_m} - t_p| \geq MinTime$  where  $s_m$  is the last point of  $N_{(Eps,K)}(p)$  OR

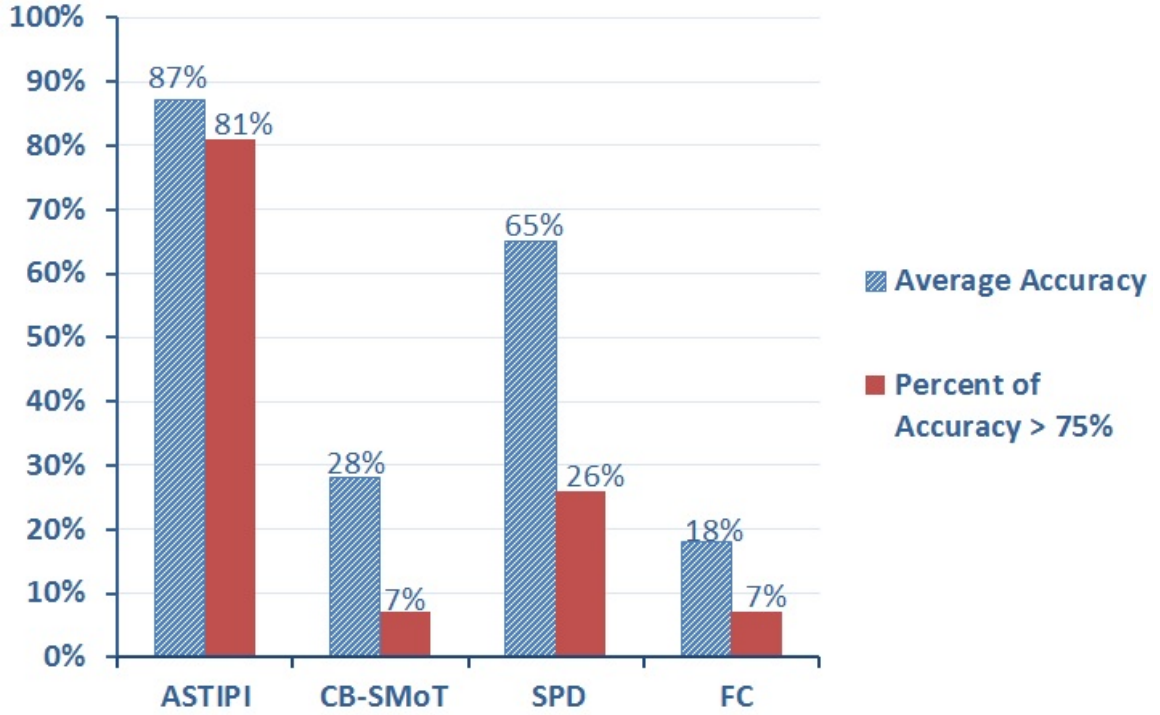


Figure 4.3: Evaluation of ASTIPI-With-Speed, CB-SMoT, SPD, FC on Full Datasets Using GPS Auto-Sleep Module

(2)  $|t_{s_m} - t_{s_{m+1}}| \geq MinTime$  where  $s_m$  is the last point of  $N_{(Eps,K)}(p)$  and  $s_{m+1}$  is the first next point not in  $N_{(Eps,K)}(p)$

(3) The speed at point  $p$  is less than  $MaxSpeed$

Notice that if  $N_{(Eps,K)}(p)$  (i.e. the *Eps-K-Neighborhood* of point  $p$ ) is empty,  $t_{s_m}$  defaults to  $t_p$ .

To reduce False Positives when running CP algorithm, a point  $p$  is a *Core Point* only if  $p$  has a slow speed, indicating the user is not moving. Thus,  $MaxSpeed$  parameter is set to 10 meters per second in the following experiment. The ASTIPI algorithm is modified according to the new definition of *Core Point With Speed*. This ASTIPI-With-Speed algorithm is evaluated by using the same *Eps* value of 100 meters and a  $K$  value of three.

Figure 4.3 and Figure 4.4 illustrates the evaluations of ASTIPI-With-Speed on the full datasets and reduced datasets compared to CB-SMoT, SPD, and FC algorithms. The performance on the full datasets with GPS Auto-Sleep slightly decreases from an average accuracy of 88% to an average

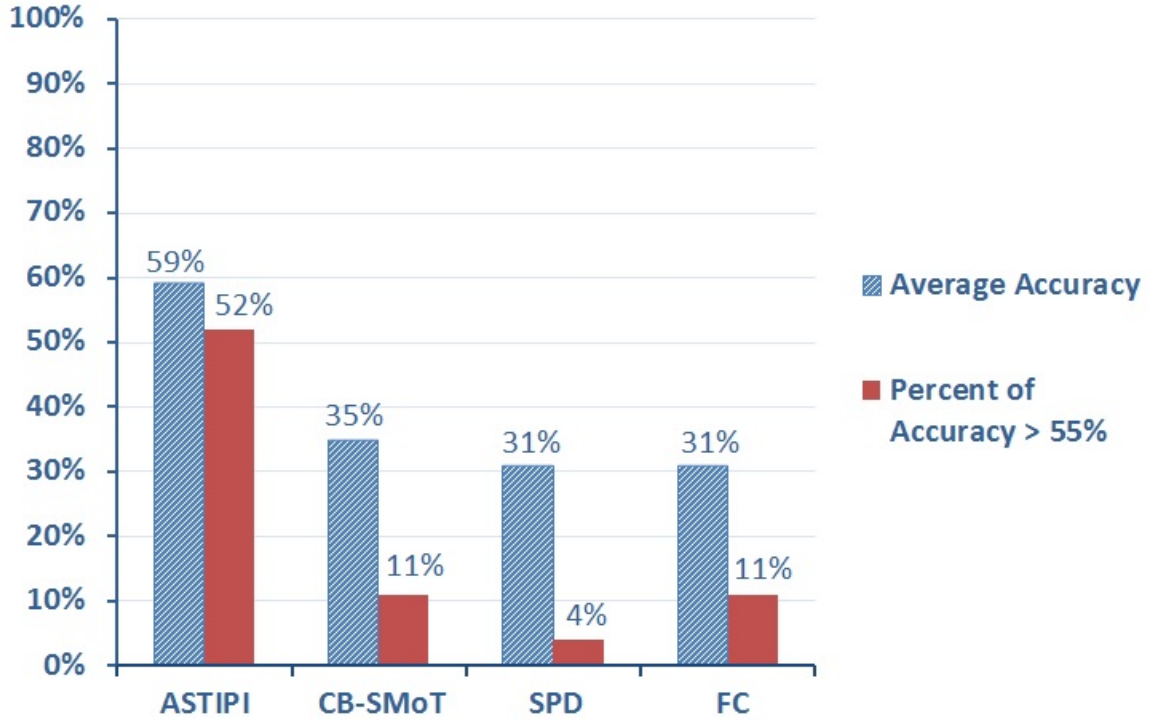


Figure 4.4: Evaluation of ASTIPI-With-Speed, CB-SMoT, SPD, FC on Reduced Datasets Using GPS Auto-Sleep Module and Critical Points Algorithm

accuracy of 87%. However, the performance on the reduced datasets combining the two strategies significantly increases from an average accuracy of 45% to an average accuracy of 59%. Notice that the accuracy increase is primarily due to the reduction in the number of False Positives. Table A.3 presents the results of the ASTIPI-With-Speed evaluation on both full datasets and reduced datasets.

#### 4.4 Running Time

This section gives a detailed analysis of ASTIPI running time. Suppose  $n$  is the size of the trajectory sample  $T$ . The two while-loops in the ASTIPI algorithm ensure that all coordinates but noise in  $T$ , regardless of being a *Core Point* or not, run the *Eps-K-Neighborhood* function. Since ASTIPI uses the *Eps-K-Neighborhood* search, it is imperative to evaluate the running time of the *Eps-K-Neighborhood* function. There is one for-loop in *Eps-K-Neighborhood* running from a given starting index to the end of the trajectory. This loop is halted after a constant  $K$  times failing to meet the close proximity condition between two coordinates. Thus, the worst case scenario of this



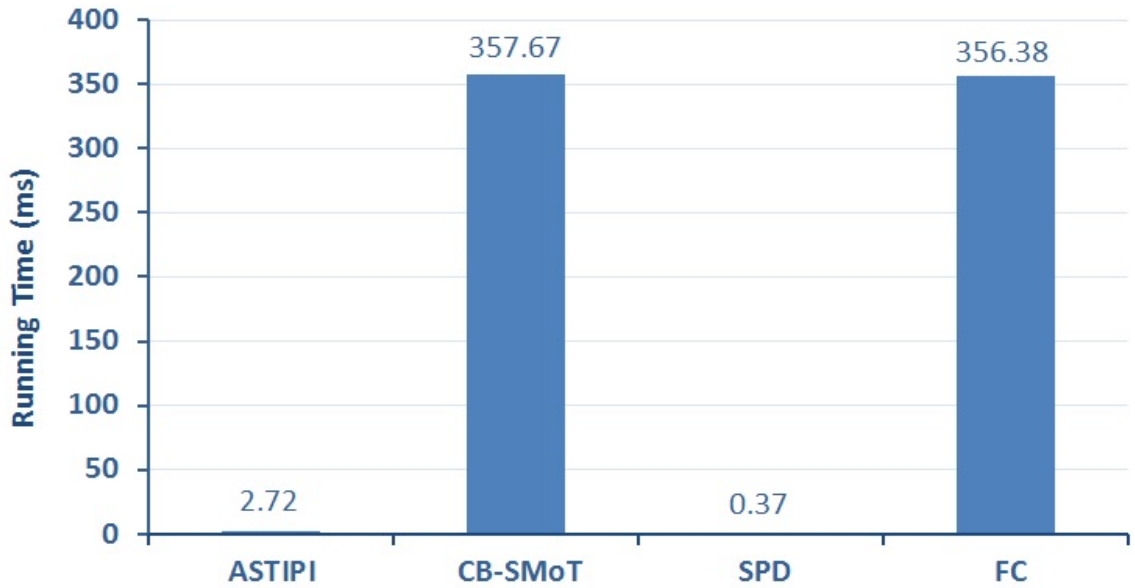


Figure 4.5: Running Time Evaluation of ASTIPI, CB-SMoT, SPD, FC on Full Datasets Using GPS Auto-Sleep Module

function happens when most coordinates in consideration are within  $Eps$  meters, which gives us  $O(n)$ . However, when the worst case scenario of  $Eps$ - $K$ - $Neighborhood$  occurs, the main ASTIPI algorithm also reaches the end of  $T$ , making the running time of ASTIPI a linear running time. Notice that the number of GPS fixes in the trajectory is proportional to its duration. Therefore, as  $n$  goes to infinity (i.e., the size of  $T$  gets larger), the total duration of the trajectory sample also gets longer while  $MinTime$  is relatively small (i.e., hours vs. minutes), making the worst case scenario of  $Eps$ - $K$ - $Neighborhood$  search always returns a *Core Point*. When  $Eps$ - $K$ - $Neighborhood$  function's worst case scenario does not occur, the search will stop after a constant number of comparisons and if the *Core Point* condition is met, the main ASTIPI algorithm will continue with the coordinate where the search left off. In total, the number of comparisons that ASTIPI processes is  $f(n) + C$ , showing that ASTIPI runs in linear time.

To evaluate the running time of ASTIPI and the other three algorithms, the processing time of each algorithm is calculated starting after the trajectory sample is retrieved from the database and ending after the list of POIs is returned from each algorithm. Each test is run twenty times and the average of all recorded running times for each test is used as the measured value for the associated test. Figure 4.5 is the result of this running time evaluation. Out of the four algorithms, SPD has



the smallest average running time of 0.37ms while ASTIPI has the second smallest average running time of 2.72ms and FC, CB-SMoT are in the third and fourth place with 356.38ms and 357.67ms, respectively. The results demonstrate that the CB-SMoT and FC algorithms are significantly slow compared to ASTIPI and SPD. Additionally, although SPD is  $O(n^2)$  in worst case scenario running time, its average running time shown in Figure 4.5 is a little faster than ASTIPI's. Notice that the worst case scenario of SPD happens when the user stays in one place for the entire dataset. For the collected datasets, this scenario contains only a small number of coordinates because once the user stays in one place, the GPS Auto-Sleep Module prevents TRAC-IT from sending duplicate location data, as it does not contribute to the knowledge of user's location. Therefore, the difference in running time is not noticeable between SPD and ASTIPI. In other cases, SPD and ASTIPI utilize the same idea that once a POI is detected, both algorithms process the next coordinate from the last one in the identified POI. Because ASTIPI takes into account noisy data, it needs to perform additional computations before continuing to the next coordinate, which explains why ASTIPI is slightly slower than SPD for average case scenarios. More information about ASTIPI running time evaluation can be found in Table A.4.

## CHAPTER 5

### SUMMARY

The popularity of mobile devices with embedded GPS has allowed more people to get access to location aware applications, especially tracking applications. While more data, especially location data, is being collected, automatic identification of POIs is needed to identify places users have been, so that other applications can suggest and predict places that users might find interesting. In addition, travel surveys can utilize the automatic algorithm to reduce time, cost, and participants' burden to carry out more accurate surveys. Nonetheless, there are still issues with the collected GPS coordinates via mobile devices including outliers, loss of location data, and duplicated trips throughout a day. In addition, limited energy resource is a major limitation in mobile devices that require tracking applications to dynamically calculate and send GPS fixes. All these problems make it difficult to obtain a high accuracy algorithm for automatic identification of POIs. A new algorithm is required to address the following existing problems:

- (1) Problem #1: Lack of POI identification algorithms that have a high accuracy.
- (2) Problem #2: Lack of POI identification algorithms that have a sufficiently high accuracy on a reduced GPS dataset for addressing the energy consumption problem on mobile devices.
- (3) Problem #3: Slow running time of  $O(n^2)$  in worst case scenarios.
- (4) Problem #4: Unable to maintain a temporal order of GPS fixes to support fast trip segmentation.

## 5.1 Summary of Contributions

This thesis presented ASTIPI, a spatial temporal algorithm for automatic identification of POIs given a set of GPS coordinates collected by mobile devices with A-GPS. The new algorithm was evaluated and compared with existing works, demonstrating that ASTIPI significantly improves the accuracy of POI identification to 88% with full datasets that use the GPS Auto-Sleep module and 59% with reduced datasets that use a combination of the GPS Auto-Sleep module and the CP algorithm, addressing Problem #1 and Problem #2. Moreover, ASTIPI also addresses Problem #3 with its linear running time while maintaining a temporal order of GPS coordinates for fast trip segmentation (Problem #4). Overall, ASTIPI is capable of identifying POIs from users' GPS tracking datasets, which can be utilized in many fields, particularly travel surveys, for cost and time savings.

## 5.2 Future Work

ASTIPI currently yields a high average accuracy of 88% with the full datasets using the GPS Auto-Sleep module for battery efficiency. However, ASTIPI only has an average accuracy of 45% (or 59% for ASTIPI-With-Speed) on the reduced datasets using the combination of the GPS Auto-Sleep module and the CP algorithm. These accuracies of 45% and 59% are low and cannot be used in practice. Although part of the reason for the low accuracies is the significant reduction in location data for some visited places that do not have any GPS fixes, it is useful to improve the ASTIPI algorithm so that the number of True Positives can be increased. One possible solution to increase the number of correctly identified POIs is to use GPS from multiple sources of the same person (e.g. combining car and person-based GPS). This approach allows tracking with high resolution while addressing the limited energy's resources of mobile devices. Another approach is to utilize map data to increase the accuracy of the algorithm.

Furthermore, all tests were performed in the Tampa, Florida area with an  $Eps$  parameter of 100 meters and a  $K$  value of three. While the  $MinTime$  duration varies depending on the type of analysis,  $Eps$  and  $K$  would be different in other cities and neighborhoods where POIs are closer together

and skyscrapers would affect the quality of computed GPS coordinates. Therefore, evaluations of ASTIPI on datasets performed in those neighborhoods are necessary. Additionally, instead of using the absolute distance *Eps* to define proximity between points, ASTIPI can be further improved by employing a relative parameter related to the dataset.

## REFERENCES

- [1] “Inventing the telephone,” accessed: 11/21/2012. [Online]. Available: <http://www.corp.att.com/history/inventing.html>
- [2] “1946: First mobile telephone call,” accessed: 11/21/2012. [Online]. Available: <http://www.corp.att.com/atllabs/reputation/timeline/46mobile.html>
- [3] “Bellsouth, IBM unveil personal communicator phone,” *Mobile Phone News*, vol. 11, no. 43, Nov. 1993. [Online]. Available: <http://go.galegroup.com/ps/i.do?id=GALE%7CA14297997&v=2.1&u=tamp44898&it=r&p=ITOF&sw=w>
- [4] “The World in 2011, ICT facts and figures,” accessed: 11/21/2012. [Online]. Available: <http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf>
- [5] K. Purcell. (2011, Nov.) Half of adult cell phone owners have apps on their phones. [Online]. Available: [http://pewinternet.org/~media/Files/Reports/2011/PIP\\_Apps-Update-2011.pdf](http://pewinternet.org/~media/Files/Reports/2011/PIP_Apps-Update-2011.pdf)
- [6] K. Zickuhr. (2011) Three-quarters of smartphone owners use location-based services. [Online]. Available: [http://pewinternet.org/~media/Files/Reports/2012/PIP\\_Location\\_based\\_services\\_2012\\_Report.pdf](http://pewinternet.org/~media/Files/Reports/2012/PIP_Location_based_services_2012_Report.pdf)
- [7] “Gps market forecast to 2015,” 2012, accessed: 11/21/2012. [Online]. Available: <http://www.businesswire.com/news/home/20120725005838/en/Research-Markets-GPS-Market-Forecast-2015>
- [8] “Inrix traffic never be late again,” 2012, accessed: 11/21/2012. [Online]. Available: <http://www.inrixtraffic.com/>
- [9] R. G. Golledge and J. Zhou, “Gps-based tracking of daily activities,” University of California Transportation Center, Tech. Rep., 2001. [Online]. Available: <http://EconPapers.repec.org/RePEc:cdl:uctcwp:qt9jb438r2>
- [10] H. Miller, “Necessary space^ time conditions for human interaction,” *Environment and Planning B: Planning and Design*, vol. 32, no. 3, pp. 381–401, 2005.
- [11] P. Stopher, P. Bullock, and F. Horst, “Exploring the use of passive gps devices to measure travel,” in *Applications of Advanced Technologies in Transportation (2002)*, 2002, pp. 959–967.
- [12] P. Stopher and P. Bullock, “Using passive gps as a means to improve spatial travel data: further findings,” in *Conference of Australian Institutes of Transport Research, 23rd, 2001, Clayton, Victoria, Australia*, 2002.

- [13] J. Wolf, R. Guensler, S. Washington, and L. Frank, "Use of electronic travel diaries and vehicle instrumentation packages in the year 2000: Atlanta regional household travel survey," *Transportation Research E-Circular*, pp. 413–429, 2001.
- [14] G. Draijer, N. Kalfs, and J. Perdok, "Global positioning system as data collection method for travel research," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1719, pp. 147–153, 2001.
- [15] R. Guensler and J. Wolf, "Development of a handheld electronic travel diary for monitoring individual tripmaking behavior," in *Transportation Research Board Annual Meeting*, 1999.
- [16] E. Murakami and D. Wagner, "Can using global positioning system (GPS) improve trip reporting?" *Transportation Research Part C: Emerging Technologies*, vol. 7, no. 23, pp. 149–165, 1999. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0968090X99000170>
- [17] P. L. Winters, S. J. Barbeau, and N. L. Georggi, "TRAC-IT field testing," in *Smart Phone Application to Influence Travel Behavior (TRAC-IT Phase 3)*, 2008, pp. 90–102. [Online]. Available: <http://www.nctr.usf.edu/pdf/77709.pdf>
- [18] E.-H. Chung and A. Shalaby, "A trip reconstruction tool for gps-based personal travel surveys," *Transportation Planning and Technology*, vol. 28, no. 5, pp. 381–401, 2005. [Online]. Available: <http://EconPapers.repec.org/RePEc:taf:transp:v:28:y:2005:i:5:p:381-401>
- [19] S. Doherty, N. Noël, M. Gosselin, C. Sirois, M. Ueno, and F. Theberge, "Moving beyond observed outcomes: integrating global positioning systems and interactive computer-based travel behavior surveys," 2001.
- [20] M. Flamm and V. Kaufmann, "Combining person based GPS tracking and prompted recall interviews for a comprehensive investigation of travel behaviour adaptation processes during life course transitions," in *11th World Conference on Transport Research*, 2007.
- [21] N. Schuessler and K. W. Axhausen, "Processing raw data from global positioning systems without additional information," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2105, pp. 28–36, 2009.
- [22] N. Ohmori, M. Nakazato, and N. Harata, "GPS mobile phone-based activity diary survey," in *Proceedings of the Eastern Asia Society for Transportation Studies*, vol. 5, 2005, pp. 1104–1115.
- [23] Y. Asakura, A. Okamoto, A. Suzuki, Y. H. Lee, and J. Tanabe, "Monitoring individual travel behaviour using PEAMON: a cellular phone based location positioning instrument combined with acceleration sensor," in *8th World Congress on Intelligent Transport Systems*, 2001.
- [24] Y. Asakura and E. Hato, "Tracking survey for individual travel behaviour using mobile communication instruments," *Transportation Research Part C: Emerging Technologies*, vol. 12, no. 34, pp. 273–291, 2004, intelligent Transport Systems: Emerging Technologies and Methods in Transportation and Traffic. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0968090X04000130>

- [25] J. Wolf, S. Hallmark, M. Oliveira, R. Guensler, and W. Sarasua, "Accuracy issues with route choice data collection by using global positioning system," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1660, pp. 66–74, 1999. [Online]. Available: <http://dx.doi.org/10.3141/1660-09>
- [26] 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*, vol. 1818, no. -1, pp. 12–24, 2002. [Online]. Available: <http://dx.doi.org/10.3141/1818-03>
- [27] P. R. Stopher, Q. Jiang, and C. FitzGerald, "Processing gps data from travel surveys," *2nd International Colloquium on the Behavioural Foundations of Integrated Land-use and Transportation Models: Frameworks, Models and Applications*, Toronto, 2005.
- [28] R. N. Mayo and P. Ranganathan, "Energy consumption in mobile devices: why future systems need requirements-aware energy scale-down," in *Proceedings of the Third International Conference on Power - Aware Computer Systems*, ser. PACS'03. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 26–40. [Online]. Available: [http://dx.doi.org/10.1007/978-3-540-28641-7\\_3](http://dx.doi.org/10.1007/978-3-540-28641-7_3)
- [29] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 280–293. [Online]. Available: <http://doi.acm.org/10.1145/1644893.1644927>
- [30] D. P. Aguilar, "A framework for evaluating the computational aspects of mobile phones," Ph.D. dissertation, Tampa, FL, USA, 2008. [Online]. Available: <http://scholarcommons.usf.edu/etd/111>
- [31] A. Leonhardi, C. Nicu, and K. Rothermel, "A map-based dead-reckoning protocol for updating location information," in *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, ser. IPDPS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 15–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645610.662039>
- [32] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and querying databases that track mobile units," *Distrib. Parallel Databases*, vol. 7, no. 3, pp. 257–387, July 1999. [Online]. Available: <http://dx.doi.org/10.1023/A:1008782710752>
- [33] G. Treu, A. Küpper, and T. Wilder, "Extending the lbs-framework trax: Efficient proximity detection with dead reckoning," *Comput. Commun.*, vol. 31, no. 5, pp. 1040–1051, Mar. 2008. [Online]. Available: <http://dx.doi.org/10.1016/j.comcom.2007.12.013>
- [34] S. Barbeau, M. Labrador, N. Georggi, P. Winters, and R. Perez, "TRAC-IT: A software architecture supporting simultaneous travel behavior data collection and real-time location-based services for gps-enabled mobile phones," in *Transportation Research Board 88th Annual Meeting*, no. 09-3175, 2009.

- [35] S. Barbeau, R. Perez, M. Labrador, A. Perez, P. Winters, and N. Georggi, “A location-aware framework for intelligent real-time mobile applications,” *Pervasive Computing, IEEE*, vol. 10, no. 3, pp. 58–67, Jul-Sep 2011.
- [36] S. Barbeau, M. Labrador, A. Perez, P. Winters, N. Georggi, D. Aguilar, and R. Perez, “Dynamic management of real-time location data on gps-enabled mobile phones,” in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBIComm '08. The Second International Conference on*, Oct 2008, pp. 343–348.
- [37] S. Barbeau, “A location-aware architecture supporting intelligent real-time mobile applications,” Ph.D. dissertation, Tampa, FL, USA, 2012. [Online]. Available: <http://scholarcommons.usf.edu/etd/3968>
- [38] S. Barbeau, P. Winters, R. Perez, M. Labrador, and N. Georggi, “Optimizing performance of location-aware applications using state machines,” Patent US 8036679, Oct 11, 2011. [Online]. Available: <http://www.google.com/patents/US8036679>
- [39] S. Barbeau, P. Winters, R. Perez, M. Labrador, and N. Georggi, “Method for determining critical points in location data generated by location-based applications,” Patent US 8249807, Aug 21, 2012. [Online]. Available: <http://www.google.com/patents/US8249807>
- [40] M. Ester, H. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, vol. 1996. AAAI Press, 1996, pp. 226–231.
- [41] D. Birant and A. Kut, “ST-DBSCAN: An algorithm for clustering spatial-temporal data,” *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, Jan. 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.datak.2006.01.013>
- [42] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares, “A clustering-based approach for discovering interesting places in trajectories,” in *Proceedings of the 2008 ACM Symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 863–868. [Online]. Available: <http://doi.acm.org/10.1145/1363686.1363886>
- [43] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma, “Mining user similarity based on location history,” in *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '08. New York, NY, USA: ACM, 2008, pp. 34:1–34:10. [Online]. Available: <http://doi.acm.org/10.1145/1463434.1463477>
- [44] N. Persad-Maharaj and S. Barbeau, “Fast clustering of global navigation satellite system data and trip segmentation,” in *Dynamic Travel Information Personalized and Delivered to Your Cell Phone*, 2011, pp. 155–190. [Online]. Available: <http://www.nctr.usf.edu/wp-content/uploads/2011/03/77804.pdf>
- [45] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macedo, B. Moelans, and A. Vaisman, “A model for enriching trajectories with semantic geographical information,” in *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, ser. GIS '07. New York, NY, USA: ACM, 2007, pp. 22:1–22:8. [Online]. Available: <http://doi.acm.org/10.1145/1341012.1341041>



- [46] D. Ashbrook and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, Oct. 2003. [Online]. Available: <http://dx.doi.org/10.1007/s00779-003-0240-0>
- [47] S. Bricka and C. R. Bhat, "Comparative analysis of global positioning system-based and travel survey-based data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1972, pp. 9–20, 2006. [Online]. Available: <http://trb.metapress.com/content/x6643h3856243663/>
- [48] J. Wolf, S. SchöUnfelder, U. Samaga, M. Oliveira, and K. Axhausen, "Eighty weeks of global positioning system traces: approaches to enriching trip information," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1870, pp. 46–54, 2004. [Online]. Available: <http://trb.metapress.com/content/4h45701736m78053/>

## APPENDICES

## Appendix A: Additional Tables

Table A.1: Results of Algorithm Evaluation on Full Datasets with GPS Auto-Sleep Module

Test	Mode of Transportation	ASTIPI				CB-SMoT				SPD				FC							
		TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)				
1	Stationary	1	0	0	100	1	0	0	100	0	1	1	0	1	0	0	100				
2	Stationary	1	0	0	100	1	0	0	100	0	1	1	0	1	0	0	100				
3	Walk	3	0	2	60	3	1	2	50	2	1	3	33	4	5	1	40				
4	Walk	4	0	1	80	1	1	4	17	4	1	1	67	2	4	3	22				
5	Walk	4	1	1	67	3	1	2	50	3	3	2	38	3	1	2	50				
6	Walk & Bus	2	0	1	67	1	1	2	25	2	1	1	50	2	5	1	25				
7	Walk & Bus	3	2	4	33	1	2	6	11	3	3	4	30	6	12	1	32				
8	Walk & Bus	4	0	0	100	2	2	2	33	4	0	0	100	4	46	0	8				
9	Walk & Bus & Drive	6	0	0	100	1	2	5	13	5	2	1	63	6	14	0	30				
10	Walk & Drive	4	0	0	100	3	3	1	43	3	0	1	75	3	28	1	9				
11	Walk & Drive	3	0	0	100	1	1	2	25	2	0	1	67	2	146	1	1				
12	Walk & Drive	4	0	1	80	1	2	4	14	5	0	0	100	5	96	0	5				
13	Walk & Drive	6	1	0	86	2	2	4	25	5	1	1	71	6	46	0	12				
14	Walk & Drive	5	0	1	83	2	5	4	18	5	1	1	71	6	95	0	6				
15	Walk & Drive	4	0	0	100	2	5	2	22	3	0	1	75	3	192	1	2				
16	Walk & Drive	5	0	0	100	2	1	3	33	4	1	1	67	5	114	0	4				
17	Walk & Drive	3	0	0	100	0	1	3	0	3	2	0	60	3	150	0	2				
18	Walk & Drive	7	1	0	88	1	5	6	8	6	2	1	67	7	118	0	6				
19	Walk & Drive	6	0	1	86	1	6	6	8	5	1	2	63	7	163	0	4				
20	Walk & Drive	6	0	0	100	4	8	2	29	6	1	0	86	6	130	0	4				
21	Walk & Drive	12	1	0	92	3	4	9	19	12	1	0	92	12	190	0	6				
22	Walk & Drive	6	0	0	100	3	2	3	38	6	2	0	75	6	92	0	6				
23	Walk & Drive	9	0	2	82	3	4	8	20	9	1	2	75	9	288	2	3				
24	Drive	5	1	0	83	3	56	2	5	4	1	1	67	5	335	0	1				
25	Drive	5	0	0	100	2	3	3	25	4	0	1	80	5	182	0	3				
26	Drive	8	0	0	100	8	88	0	8	8	0	0	100	8	142	0	5				
27	Drive	10	0	1	91	3	3	8	21	9	0	2	82	10	278	1	3				
<b>Average Accuracy</b>					88%						28%						65%				18%
<b>Percent of Accuracy &gt; 75%</b>					85%						7%						26%				7%

## Appendix A (Continued)

Table A.2: Results of Algorithm Evaluation on Reduced Datasets with GPS Auto-Sleep Module and Critical Points Algorithm

Test	Mode of Transportation	ASTIPI				CB-SMoT				SPD				FC									
		TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)						
1	Stationary	1	0	0	100	1	0	0	100	0	0	1	0	1	0	0	100						
2	Stationary	1	1	0	50	1	0	0	100	0	0	1	0	1	0	0	100						
3	Walk	1	0	4	20	1	0	4	20	1	1	4	17	4	5	4	17						
4	Walk	3	1	2	50	2	1	3	33	2	3	3	25	2	4	3	29						
5	Walk	2	1	3	33	1	0	4	20	3	1	2	50	3	1	3	40						
6	Walk & Bus	2	1	1	50	2	1	1	50	1	2	2	20	2	5	1	25						
7	Walk & Bus	2	0	5	29	3	5	4	25	3	1	4	38	6	12	4	18						
8	Walk & Bus	2	4	2	25	1	3	3	14	3	4	1	38	4	46	1	23						
9	Walk & Bus & Drive	5	1	1	71	3	1	3	43	5	2	1	63	6	14	1	31						
10	Walk & Drive	3	1	1	60	2	2	2	33	2	2	1	40	3	28	1	27						
11	Walk & Drive	2	1	1	50	2	1	1	50	2	2	1	40	2	146	1	29						
12	Walk & Drive	2	1	3	33	1	2	4	14	3	2	2	43	5	96	3	29						
13	Walk & Drive	5	3	1	56	1	1	5	14	5	4	1	50	6	46	3	25						
14	Walk & Drive	3	1	3	43	2	3	4	22	4	2	2	50	6	95	2	36						
15	Walk & Drive	3	2	1	50	2	1	2	40	3	3	1	43	3	192	1	25						
16	Walk & Drive	4	4	1	44	3	6	2	27	3	6	2	27	5	114	2	15						
17	Walk & Drive	2	2	1	40	2	1	1	50	2	2	1	40	3	150	0	25						
18	Walk & Drive	5	4	2	45	2	6	5	15	5	4	2	45	7	118	4	21						
19	Walk & Drive	6	4	1	55	3	4	4	27	5	4	2	45	7	163	3	33						
20	Walk & Drive	5	7	1	38	4	8	2	29	5	7	1	38	6	130	2	19						
21	Walk & Drive	10	2	2	71	8	1	4	62	9	5	3	53	12	190	2	56						
22	Walk & Drive	4	3	2	44	4	9	2	27	3	7	3	23	6	92	3	17						
23	Walk & Drive	6	11	5	27	5	7	6	28	6	13	5	25	9	288	6	16						
24	Drive	4	6	1	36	5	14	0	26	5	6	0	45	5	335	1	10						
25	Drive	3	3	2	38	1	3	4	13	3	3	2	38	5	182	2	43						
26	Drive	5	9	3	29	5	4	3	42	5	8	3	31	8	142	2	21						
27	Drive	6	8	5	32	5	18	6	17	6	9	5	30	10	278	5	11						
<b>Average Accuracy</b>					45%						35%						31%						31%
<b>Percent of Accuracy &gt; 55%</b>					19%						11%						4%						11%

Appendix A (Continued)

Table A.3: Results of ASTIPI-With-Speed Algorithm Evaluation

Test	Mode of Transportation	On Full Dataset with GPS Auto-Sleep Module				On Reduced Dataset with GPS Auto-Sleep Module and Critical Points Algorithm Strategy			
		TP	FP	FN	Acc (%)	TP	FP	FN	Acc (%)
1	Stationary	1	0	0	100	1	0	0	100
2	Stationary	1	0	0	100	1	0	0	100
3	Walk	3	0	2	60	1	0	4	20
4	Walk	4	0	1	80	3	1	2	50
5	Walk	4	1	1	67	2	1	3	33
6	Walk & Bus	2	0	1	67	2	1	1	50
7	Walk & Bus	3	2	4	33	2	0	5	29
8	Walk & Bus	4	0	0	100	2	2	2	33
9	Walk & Bus & Drive	6	0	0	100	5	1	1	71
10	Walk & Drive	4	0	0	100	3	0	1	75
11	Walk & Drive	3	0	0	100	2	0	1	67
12	Walk & Drive	4	0	1	80	2	0	3	40
13	Walk & Drive	6	1	0	86	5	1	1	71
14	Walk & Drive	5	0	1	83	3	1	3	43
15	Walk & Drive	3	0	1	75	3	0	1	75
16	Walk & Drive	5	0	0	100	4	1	1	67
17	Walk & Drive	3	0	0	100	2	1	1	50
18	Walk & Drive	7	1	0	88	4	1	3	50
19	Walk & Drive	6	0	1	86	4	2	3	44
20	Walk & Drive	6	0	0	100	6	0	0	100
21	Walk & Drive	12	1	0	92	9	1	3	69
22	Walk & Drive	6	0	0	100	4	1	2	57
23	Walk & Drive	9	0	2	82	7	2	4	54
24	Drive	5	1	0	83	4	0	1	80
25	Drive	5	0	0	100	3	0	2	60
26	Drive	8	0	0	100	6	1	2	67
27	Drive	10	0	1	91	6	1	5	50
<b>Average Accuracy</b>					87%	<b>Average Accuracy</b>			59%
<b>Percent of Accuracy &gt; 75%</b>					81%	<b>Percent of Accuracy &gt; 55%</b>			52%

## Appendix A (Continued)

Table A.4: Results of Algorithm Average Running Time for Twenty Runs of each Test

Test	Mode of Transportation	Size (Points)	ASTIPI (ms)	CB-SMoT (ms)	SPD (ms)	FC (ms)
1	Stationary	75	0.05	0.75	0.05	0.70
2	Stationary	79	0.05	0.75	0.05	0.90
3	Walk	76	0.90	1.60	0.05	1.90
4	Walk	332	2.15	15.20	0.25	16.70
5	Walk	122	0.10	2.25	0.05	3.05
6	Walk & Bus	430	1.55	21.95	0.10	21.60
7	Walk & Bus	615	2.00	44.10	0.20	44.70
8	Walk & Bus	1489	5.00	258.70	0.40	263.55
9	Walk & Bus & Drive	1000	2.35	113.90	0.15	119.65
10	Walk & Drive	730	2.55	63.50	0.35	77.70
11	Walk & Drive	791	1.40	73.10	0.20	77.90
12	Walk & Drive	988	1.60	114.20	0.30	117.45
13	Walk & Drive	1086	1.70	136.40	0.25	140.90
14	Walk & Drive	1137	2.05	151.00	0.25	155.30
15	Walk & Drive	1186	2.05	162.90	0.35	168.65
16	Walk & Drive	1201	2.30	166.90	0.30	177.70
17	Walk & Drive	1222	2.25	171.95	0.35	179.20
18	Walk & Drive	1372	2.60	217.35	0.35	230.05
19	Walk & Drive	1848	3.80	409.80	0.50	418.90
20	Walk & Drive	1898	3.25	429.95	0.55	443.80
21	Walk & Drive	1964	3.40	450.00	0.50	471.35
22	Walk & Drive	2492	4.40	726.85	0.70	738.25
23	Walk & Drive	3122	6.40	1119.85	0.85	1154.65
24	Drive	2625	4.45	830.90	0.60	817.40
25	Drive	1199	1.75	167.25	0.25	174.05
26	Drive	3498	5.75	1508.35	0.95	1411.10
27	Drive	4331	7.55	2297.60	1.05	2195.05
<b>Average Running Time</b>			2.72	357.67	0.37	356.38

## Appendix B: Reprint Permissions for Use of Figure 2.2 and Figure 2.3



RightsLink®

Home

Create Account

Help



**Title:** Dynamic Management of Real-Time Location Data on GPS-Enabled Mobile Phones  
**Conference Proceedings:** Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM '08. The Second International Conference on  
**Author:** Barbeau, S.; Labrador, M.A.; Perez, A.; Winters, P.; Georggi, N.; Aguilar, D.; Perez, R.  
**Publisher:** IEEE  
**Date:** Sept. 29 2008-Oct. 4 2008  
Copyright © 2008, IEEE

User ID
Password
<input type="checkbox"/> Enable Auto Login
<input type="button" value="LOGIN"/>
<a href="#">Forgot Password/User ID?</a>
<b>If you're a copyright.com user,</b> you can login to RightsLink using your copyright.com credentials. Already a RightsLink user or want to <a href="#">learn more?</a>

### Thesis / Dissertation Reuse

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to [http://www.ieee.org/publications\\_standards/publications/rights/rights\\_link.html](http://www.ieee.org/publications_standards/publications/rights/rights_link.html) to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK

CLOSE WINDOW

Copyright © 2013 [Copyright Clearance Center, Inc.](#) All Rights Reserved. [Privacy statement.](#)